Automatic Keyphrase Extraction: A Comparison of Methods

Richard Hussey, Shirley Williams, Richard Mitchell
School of Systems Engineering
University of Reading
Reading, United Kingdom
{r.j.hussey, shirley.williams, r.j.mitchell}@reading.ac.uk

***Abstract*—There are many published methods available for creating keyphrases for documents. Previous work in the field has shown that in a significant proportion of cases author selected keyphrases are not appropriate for the document they accompany. This requires the use of such automated methods to improve the use of keyphrases. Often the keyphrases are not updated when the focus of a paper changes or include keyphrases that are more classificatory than explanatory. The published methods are all evaluated using different corpora, typically one relevant to their field of study. This not only makes it difficult to incorporate the useful elements of algorithms in future work but also makes comparing the results of each method inefficient and ineffective. This paper describes the work undertaken to compare five methods across a common baseline of six corpora. The methods chosen were term frequency, inverse document frequency, the C-Value, the NC-Value, and a synonym based approach. These methods were compared to evaluate performance and quality of results, and to provide a future benchmark. It is shown that, with the comparison metric used for this study Term Frequency and Inverse Document Frequency were the best algorithms, with the synonym based approach following them. Further work in the area is required to determine an appropriate (or more appropriate) comparison metric.**

Keywords- Term Frequency, Inverse Document Frequency, C-Value, NC-Value, Synonyms, Comparisons, Automated Keyphrase Extraction, Document Classification.

I. INTRODUCTION

The field of natural language processing contains many algorithms devoted to the process of automatic keyphrase extraction (AKE) but the systems lack a common baseline of having been tested on the same corpora.

The initial decision to make this comparison study stemmed from earlier work [1] that had shown a tendency on the part of authors to use corpora from their discipline area. For example, those of a medical background used medical corpora such as the PubMed Central database, while those in literature might use the Journal on Applied Linguistics. This made the task of comparing the effectiveness of one method to another more complex, as there was no common thread or baseline.

This comparison study builds on a pilot study [1] that showed for a small number of algorithms and corpora, the Term Frequency method was the best. Therefore, this paper sets out to compare the outputs of five systems on the same six corpora. The methods chosen were all in the field of AKE. The C-Value [2] (and its follow-on the NC-Value [2]) demonstrated a series of linguistic filters for determining what phrases should be considered, and uses a ranking metric based on sub-strings. Hussey et al [3] showed that using a thesaurus to group synonyms into keyphrases could be used to improve the results from analysing the document for common themes.

A. Background

A topic, theme, or subject of a document can be identified by keywords: a collection of words that classify a document. Academic papers make use of them to outline the topics of the paper (such as papers about "metaphor" or "leadership"), books in libraries can be searched by keyword (such as all books on "Stalin" or "romance"), and there are numerous other similar uses. The keywords for a document indicate the major areas of interest within it.

A broader way of capturing a concept is to use a short phrase, typically of one to five words, known as a keyphrase. A short phrase of a few linked words can be inferred to contain more meaning than a single word alone, e.g., the phrase "natural language processing" is more useful than just the word "language".

Sood et al. showed [4] (using the Technorati blog [5] as their source document) that a small number of keywords and keyphrases tend to be used (or reused) frequently, while a much larger number are idiosyncratic and demonstrate a low frequency as they are too specific to be reused even by the same author (examples include). Examples of reused phrases included "politics" and "shopping" [5], while "insomnia due to quail wailing" and "streetball china" [5] were among the examples of the idiosyncratic phrases. Additionally Sood et al. showed that in half of cases the keyphrases chosen by an author were not suited to the document to which they were attached.

The task faced by automatic keyphrase extraction (AKE) is to select the small collection of relevant words that can be used to describe or categorise the document. The process of AKE is discussed by Frank et al. [6]. AKE is characterised by using phrases from the source document (or a reference document).

The rest of the paper comprises a review of the algorithms (Section II), the implementation (Section III), and results gained (Section IV), a discussion of the outcomes (Section V), conclusions, and future work (Section VI).

 ISBN: 978-1-61208-181-6

II. REVIEW

In this section, relevant methods and the associated results are discussed. The term frequency and term frequency inverse document frequency methods are pure statistical methods, and their generic use is discussed first.

A. Term Frequency and Inverse Document Frequency

The term frequency is simply the number of times a given term (single word) appears in the given document, normalised to prevent bias toward longer documents (longer documents may have higher term counts regardless of importance of the term) as shown in Equation 1. The higher the term frequency, the more likely the term is to be important.

$$tf(t,d) = \frac{f(t)}{n} \quad (1)$$

Where:

- $tf(t,d)$ is the term frequency for term 't' in document 'd'.
- $f(t)$ is the frequency of the occurrence of the term 't' in the corpus.
- n is the number of terms in the document 'd'.

The inverse document frequency is a measure of the importance of the term to the corpus in general terms. This is achieved by dividing the number of documents in the corpus by the number of documents that contain that term, and then taking the logarithm of the result. This is shown in Equation 2.

$$idf(t) = \log \frac{|D|}{|\{d: t \in d\}|} \quad (2)$$

Where:

- $idf(t)$ is the Inverse Document Frequency for term 't'
- $|D|$ is the total number of documents
- $|\{d: t \in d\}|$ is the number of documents including 't'

Given that if the term 't' does not occur in the corpus, the current denominator can lead to a division-by-zero, it is common to alter Equation 2 as shown in Equation 3

$$idf(t) = \log \frac{|D|}{1 + |\{d: t \in d\}|} \quad (3)$$

The inverse document frequency is then used as a modifying value upon the term frequency, to reduce the value of those terms that are common across all documents. To achieve this Equation 1 and Equation 3 are combined to form Equation 4.

$$tfidf(t,d) = tf(t,d) \times idf(t) \quad (4)$$

A high weight (indicating importance) is achieved by having a high TF in the given document and a low occurrence in the remaining documents in the corpus – hence filtering out common terms (including stop words such as "the" or "and").

B. C-Value

The C-Value algorithm [2] creates a ranking for potential keyphrases (Frantziy et al. refer to them as "term words") by using the length of the phrase, and the frequency with which it occurs as a sub-string of other phrases.

To start the process, the system tags the corpus with part-of-speech data and extracts strings that pass a linguistic filter (see below) and a frequency threshold. Frantziy et al. used three different linguistic filters (expressed as regular expressions) in the first stage of the algorithm, and tested the system against each of them. The broader the filter, the more phrases it lets through. Filter 1 is the strictest, where as Filter 3 the broadest. The filters were:

1. Noun + Noun
2. (Adj | Noun) + Noun
3. ((Adj | Noun) + | ((Adj | Noun) * (NounPrep)?) (Adj | Noun)*) Noun

Assuming that a phrase a gets through the filter, then its C-Value is calculated as shown in Equation 5. Its value is dependent on whether or not a is a sub-string nested inside another valid phrase.

$$Cvalue(a) = \begin{cases} \log_2|a| \cdot f(a) & a \text{ is not } a \text{ sub} - string \\ \log_2|a| \cdot \left(f(a) - \frac{1}{P(T_a)} \sum_{b \in T_a} f(b) \right) & else \end{cases} \quad (5)$$

Where:

- a is the candidate phrase
- $|a|$ is the length of the phrase a in words
- $f(x)$ is the frequency of the occurrence of 'x'
- T_a is the set of phrases that contain a
- $P(T_a)$ is the number of those phrases

Once the C-Value has been calculated, it is used to rank the phrases and the best phrases are selected for use as keyphrases.

Frantziy et al. used two metrics to compare the results: Recall and Precision. Recall was the percentage of the keyphrases in the baseline frequency list that were extracted by the C-value algorithm. Precision was the percentage of the keyphrase in the total list that the domain-subject expert agreed with. For Precision, the broader the filter the lower the increase – although all filters showed an improvement of between 1 and 2%. For Recall, the results were broadly similar in tone and dropped the broader the filter from between 2.5% and 2%.

C. NC-Value

The NC-Value [2] extends the C-Value algorithm by using the words adjacent to of the keyphrase to add a weighting context to the phrase itself. The weighting is a percentage chance that the word is a context word for a phrase rather than just an adjacent word.

To calculate the NC-Value, the C-Value algorithm is modified by a "*context weighting factor*" which is determined by the nouns, verbs, and adjectives adjoining the keyphrase (these are known as context words). The weight is calculated as shown in Equation 6.

$$weight(w) = \frac{t(w)}{n} \quad (6)$$

Where:

- w is the context word (noun, verb, or adjective)
- $t(w)$ is the number of words 'w' occurs with

- n is the total number of phrases

This is then fed into Equation 7, the NC-Value. (7)

$$NCvalue(a) = 0.8Cvalue(a) + 0.2 \sum_{b \in C_a} f_a(b) weight(b)$$

The values of 0.8 and 0.2 used were arrived at following experimentation by Frantziy et al. [2], and therefore may only be applicable to the medical corpora they used.

Frantziy et al. compared the C-Value and NC-Value using the previous defined the Recall and Precision metrics (see Section III.D). The Recall remained the same, as did the average Precision. However the exact Precision varied by section of the output list. The Precision increased in the top section of the list (the top 40 items), and it was reduced in the remainder of the list. This was the expected behaviour, as the aim of the NC-Value was to reorganise the output list to move the better phrases toward the top.

D. Roget and Synoynms

The synonym algorithm [3] takes words from the source document, and groups them together with words that are considered synonyms. It uses a resource document in the form of a thesaurus to aid this. The basic formula for this is shown in Equation 8.

$$KE_N(p_i) = \frac{f(\{w_j : w_j \in S_{p_i}\}) \cdot |w_j|}{|\{S_{p_i}\}|} \quad (8)$$

Where:

- p_i is the candidate phrase
- $f(x)$ is the frequency of occurrence of 'x'
- S_{p_i} is the set of synonyms which p_i belongs to
- $\{w_j : w_j \in S_{p_i}\}$ is all the phrases in set S_{p_i}
- $|w_j|$ is the length of the phrase in words
- $|\{S_{p_i}\}|$ is the number of synonyms in the set

In addition, the unigram list was enhanced by adding the stemmed forms of the unigrams.

However, this method has a tendency to produce a set of keyphrases that are all, almost by definition, synonyms of each other. For example, the words "acquisition" or "taking" can both mean "recovery" [7] and therefore both may have been present as separate keyphrases. Therefore, an additional clustering element is added to group these keyphrases into their synonym groups to prevent a single, popular, concept from dominating. This simply involves applying the original Equation 8 to the keyphrases.

III. IMPLEMENTATION

The basis of the work presented here is the examination of a document with the intention of generating keyphrases from it. The implementation details that follow explain the algorithms of the methods compared. Where more than one method or set of configurations was presented, the chosen settings implemented were those that the authors of those papers found produced the best output.

The different algorithms were tested against six corpora. For this study of the algorithms, it was decided that the corpora would be restricted to academic papers, which for the majority case are submitted with keywords against which the results can be tested.

Five corpora were taken from the Academics Conferences International (ACI) e-journals [8], each corpus on a different subject area: *Business Research Methods* (EJBRM), *E-Government* (EJEG), *E-Learning* (EJEL), *Information Systems Evaluation* (EJISE), and *Knowledge Management* (EJKM). The sixth corpus was taken from PubMed Central (PMC) [9], an archive of biomedical and life science journal papers. The thesaurus used was Roget's "*Thesaurus of English Words and Phrases*" [7] and the Porter Stemming algorithm [10] was used to stem the unigrams.

A. Chance Study

For the chance study, the words from the source document were split into a list of individual words. From this list, a start point was chosen at random and a number of contiguous words were strung together to form a keyphrase. The maximum length of the keyphrase was set at $n = 7$ as this was the longest phrase in the reference document, Roget's Thesaurus [7]. The algorithm used was:

- Randomly select a word in the source document to act as a starting point.
- After each word is added, generate a random number less than or equal to n. If this number is greater than the number of words already in the phrase, add another word.
- Repeat until r keyphrases have been produced (in this study, r was chosen to be 5).

B. Term Frequency

For the term frequency (TF) study, the source document was split into a list of single words and a count of the number of times each appeared in the source document occurred, and then normalised to prevent bias towards longer document. The results were then ranked in size order, and the top five taken as the results. The algorithm for this was:

- Count the occurrence of each word in the document
- Divide the count by the total number of words
- Sort the list by frequency and output the top r ranked items (in this study, r was chosen to be 5)

C. Term Frequency – Inverse Document Frequency

The inverse document frequency (IDF) extends the TF algorithm in an attempt to automatically remove from consideration phrases that are not important as keyphrases because they are common to the whole corpus.

- Count the occurrence of each word in the document
- Take the logarithm of the number of documents in the corpus divided by the number of documents containing that word
- Multiple the two values
- Sort the list by frequency and output the top r ranked items (in this study, r was chosen to be 5)

D. The C-Value

For the C-Value, the document was first filtered through a series of linguistic filters as set out in Section II.B. Phrases

that pass the filter are then categorised as either being stand-alone or as being a sub-phrase of another. Those that are stand-alone are assigned a C-Value based on the base-2 log of their number of words. Those that are sub-phrases are given a more complex C-Value as explained in Section II.B and in the algorithm given below:

- Sift phrases through linguistic filter
- For each valid phrase, take the log (base 2) of the number of words in it, and check to see if it is nested inside another valid phrase
    - If it is not then multiply by the number of times it occurs
    - If it is nested, then multiply by the number of time it occurs minus the sum of the number of times all the longer phrases occur divided by the number of those phrases
- Sort the list by frequency and output the top *r* ranked items (in this study, *r* was chosen to be 5).

E. The NC-Value

The NC-Value extends the C-Value by taking the output from that system and adding on a contextual weight. The weight takes into account the words that surround the candidate phrase. The algorithm for this is:

- Sift phrases through linguistic filter
- For each valid phrase, take the log (base 2) of the number of words in it, and check to see if it is nested inside another valid phrase
    - If it is not then multiply by the number of times it occurs
    - If it is nested, then multiply by the number of time it occurs minus the sum of the number of times all the longer phrases occur divided by the number of those phrases
- Multiply this by 0.8 and then add 0.2 multiplied by the context weight for the phrase
    - The weight is the number of phrases that the context word occurs adjacent to divided by the total number of phrases
- Sort the list by frequency and output the top *r* ranked items (in this study, *r* was chosen to be 5).

F. Roget and Synonyms

The synonym study grouped words in the document together by also including counts of their synonyms in the process. This, however, produced a set of keyphrases that tended to be synonyms of each other, so the synonym study further extended the method to involve a second round of clustering to try to prevent a single, popular concept from dominating. For example, the words "acquisition" or "taking" can both mean "recovery" [7] and therefore both may have been present as separate keyphrases.

- For each *n*-gram in the thesaurus, compare the *n*-gram to the associated synonyms
- For each synonym that matches, add the word to a list, and increase its frequency value by the value of the *n*-gram divided by the number of associated synonyms
- Then, for each Key entry in the thesaurus check to see if the frequency is equal to the highest frequency value in the found in the preceding step.
- For each synonym entry associated with the Key, add the synonym to a second list of words and increase its value by one.
- Sort the list by frequency and output the top *r* ranked items (in this study, *r* was chosen to be 5).

IV. RESULTS

The following section sets out the results of the five algorithms studied. For each of the papers analysed in the corpora, the authors (for the most part) supplied an accompanying list of keyphrases to summarise the content. The results of the algorithms were automatically evaluated by comparing them to the author keyphrases. Where a paper did not have author-supplied keywords, it was automatically excluded from the study and the results.

A match was recorded for a paper if at least one of the output keyphrases matched one of the author keyphrases. However, a naïve text matching approach was used for this initial study. The approach would match any two strings if they were either equivalent or a sub-string of the other, e.g. "know" and "knowledge" would be considered a match.

The following tables are all formatted in the same way. They list the 'Corpus' used in the first column and the number of 'Papers' with keyphrases in that corpus. The number 'Matched' is the number of papers that met the above matching criteria as a raw figure and as a 'Percentage'. The 'Increase' column, where it occurs, is the numerical value by which the percentage differs from the chance results – i.e. if the match percentage was 1% in the chance study and 10% in the TF study, then that would be an increase of nine.

The results are also summarised in Figure 1 on page 5.

A. Chance Study

The chance results showed almost no keyphrases being produced that matched the authors'. The results can be seen in Table I.

TABLE I. CHANCE RESULTS

Corpus	Papers	Matched	Percentage
EJBRM	65	0	0.00%
EJEG	101	2	1.98%
EJEL	111	0	0.00%
EJISE	90	1	1.11%
EJKM	104	5	4.81%
PMC	137	1	0.73%
Average			1.44%

B. Term Frequency

Table II shows the results from the term frequency study, and that it performed very well matching on average over 80% of the keyphrases against the authors'.

 ISBN: 978-1-61208-181-6

TABLE II. TF RESULTS

Corpus	Papers	Matched	Percentage	*Increase*
EJBRM	65	58	89.23%	*89.23*
EJEG	101	93	92.08%	*90.10*
EJEL	111	89	80.18%	*80.18*
EJISE	90	80	88.89%	*87.78*
EJKM	104	101	97.12%	*92.31*
PMC	137	105	76.64%	*75.91*
Average			87.36%	*85.92*

C. Term Frequency – Inverse Document Frequency

The inverse document algorithm showed a drop in performance compared to the simple term frequency results, as shown in Table III.

TABLE III. TF*IDF RESULTS

Corpus	Papers	Matched	Percentage	*Increase*
EJBRM	65	43	66.15%	*66.15*
EJEG	101	66	65.35%	*63.37*
EJEL	111	69	62.16%	*62.16*
EJISE	90	69	76.67%	*75.56*
EJKM	104	71	68.27%	*63.46*
PMC	137	107	78.10%	*77.37*
Average			69.45%	*68.01*

D. The C-Value

As there were three linguistic filters for the C-Value, the results in Table IV show the range of the matched values and then an averaged percentage. Contrary to expectations based on the original paper [2], the broader the filter the better the results.

TABLE IV. C-VALUE RESULTS

Corpus	Papers	Matched	Percentage	*Increase*
EJBRM	65	10-19	~23.08%	*~23.08*
EJEG	101	16-30	~23.76%	*~21.78*
EJEL	111	1-5	~1.80%	*~1.80*
EJISE	90	11-12	~12.22%	*~11.11*
EJKM	104	3-7	~4.81%	*~0.00*
PMC	137	25-31	~21.17%	*~20.44*
Average			~14.47%	*~13.03*

E. The NC-Value

Similar to the C-Value, the results for the NC-Value are displayed as ranges for the matches and as an average percentage.

TABLE V. NC-VALUE RESULTS

Corpus	Papers	Matched	Percentage	*Increase*
EJBRM	65	1-4	~3.08%	*~3.08*
EJEG	101	0	0.00%	*-1.98*
EJEL	111	0	0.00%	*0.00*
EJISE	90	0	0.00%	*-1.11*
EJKM	104	0	0.00%	*-4.81*
PMC	137	0	0.00%	*-0.73*
Average			~0.51%	*~-0.93*

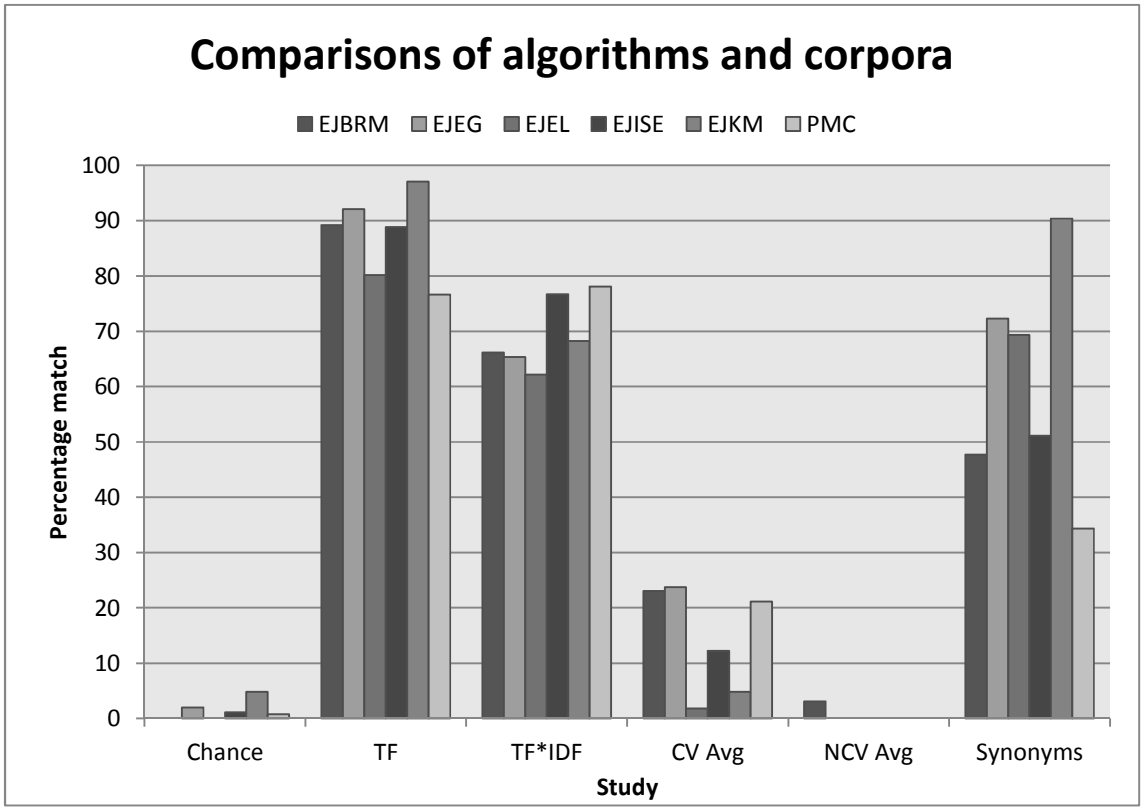

Figure 1

F. Roget and Synonyms

The synonym results show a good improvement over the baseline results (nearly 60% on average), although particular corpora fared poorly (the medical corpus PMC for example, compared to the Knowledge Management corpus). The results are shown in Table VI.

TABLE VI. ROGET AND SYNONYM RESULTS

Corpus	Papers	Matched	Percentage	*Increase*
EJBRM	65	31	47.69%	*47.69*
EJEG	101	73	72.28%	*70.30*
EJEL	111	77	69.37%	*69.37*
EJISE	90	46	51.11%	*50.00*
EJKM	104	94	90.38%	*85.57*
PMC	137	47	34.31%	*33.58*
Average			60.86%	*59.42*

V. DISCUSSION

The results outlined in Section IV above show that the Term Frequency algorithm performed best – the pure statistical method of simply counting how often a word occurred in the document.

However, as has been pointed out in the results section, the matching criteria was naïve. It is perhaps the contention of this paper, but a suggested reason for this is that keyphrases supplied by the paper authors are always likely to contain at least one "common" word that would show up in a frequency count. This would also explain the poor results produced by Inverse Document Frequency algorithm, as common words in the corpus are likewise likely to be keyphrases supplied by the author. For example, it is likely that a paper in the e-Journal on Knowledge Management both frequently uses the phrase "knowledge" and has it as an author-assigned keyphrase.

Furthermore, as shown by Sood et al. [4] author-assigned keywords are inappropriate chosen 51.15% of the time. These factors combined suggest that the matching criteria needs to be changed for future work and a Recall/Precision model as used by Frantziy et al. would seem appropriate.

VI. CONCLUSION AND FURTHER WORK

This study has shown that for the naïve comparison method used the results are biased towards phrases that occur most often in the document. However, further studies need to be run with a more standard set of evaluation criteria (such as Recall and Precision) and to be tested on a wider range of corpora – including Reuters-21578 corpus and the remainder of the PMC corpus.

In addition, work needs to be undertaken to validate the outputs of the algorithms by human judges to assess the suitability of both the keyphrases provided by the authors and by the algorithms.

ACKNOWLEDGMENT

The authors would like to thank the School of Systems Engineering for the studentship, which enabled this project, and the contributions from the reviewers of this paper.

REFERENCES

[1] R. Hussey, S. Williams, and R. Mitchell. 2011. "A Comparison of Methods for Automatic Document Classification", Proceedings of BAAL, The Forty Fourth Annual Meeting of the British Association for Applied Linguistics. Bristol, United Kingdom.

[2] K. Frantziy, S. Ananiadou, and H. Mimaz. 2000. "Automatic Recognition of Multi-Word Terms: the C-value/NC-value Method", International Journal on Digital Libraries , 3 (2), pp. 117-132.

[3] R. Hussey, S. Williams, and R. Mitchell. 2011. "Keyphrase Extraction by Synonym Analysis of *n*-grams for E-Journal Classification", Proceedings of eKNOW, The Third International Conference on Information, Process, and Knowledge Management, pp. 83-86. Gosier, Guadeloupe/France. http://www.thinkmind.org/index.php?view=article&articleid=eknow_2011_4_30_60053 [Last access: 5 September 2011]

[4] S.C. Sood, S.H. Owsley, K.J. Hammond, and L. Birnbaum. 2007. "TagAssist: Automatic Tag Suggestion for Blog Posts". Northwestern University. Evanston, IL, USA. http://www.icwsm.org/papers/2--Sood-Owsley-Hammond-Birnbaum.pdf [Last accessed: 13 December 2010]

[5] Technorati. 2006. "Technorati". http://www.technorati.com [Last accessed: 13 December 2010]

[6] E. Frank, G.W. Paynter, I.H. Witten, C. Gutwin, and C.G. Nevill-Manning. 1999. "Domain-Specific Keyphrase Extraction", Proceedings 16th International Joint Conference on Artificial Intelligence, pp. 668–673. San Francisco, CA Morgan Kaufmann Publishers.

[7] P.M. Roget. 1911. "Roget's Thesaurus of English Words and Phrases (Index)". http://www.gutenberg.org/etext/10681 [Last accessed: 13 December 2010]

[8] Academics Conferences International. 2009. "ACI E-Journals". http://academic-conferences.org/ejournals.htm [Last accessed: 13 December 2010]

[9] PubMed Central. 2011. "PubMed Central Open Access Subset". http://www.ncbi.nlm.nih.gov/pmc/tools/openftlist/ [Last accessed: 14 September 2011]

[10] M.F. Porter. 1980. "An algorithm for suffix stripping", Program, 14(3) pp. 130–137.

 ISBN: 978-1-61208-181-6