Automatic Keyword Extraction from Individual Documents

CHAPTER · MARCH 2010

DOI: 10.1002/9780470689646.ch1

Stuart Rose
Pacific Northwest National …
15 PUBLICATIONS **100** CITATIONS
SEE PROFILE

Nick Cramer
Pacific Northwest National …
14 PUBLICATIONS **97** CITATIONS
SEE PROFILE

1

Automatic keyword extraction from individual documents

Stuart Rose, Dave Engel, Nick Cramer and Wendy Cowley

1.1 Introduction

Keywords, which we define as a sequence of one or more words, provide a compact representation of a document's content. Ideally, keywords represent in condensed form the essential content of a document. Keywords are widely used to define queries within information retrieval (IR) systems as they are easy to define, revise, remember, and share. In comparison to mathematical signatures, keywords are independent of any corpus and can be applied across multiple corpora and IR systems.

Keywords have also been applied to improve the functionality of IR systems. Jones and Paynter (2002) describe Phrasier, a system that lists documents related to a primary document's keywords, and that supports the use of keyword anchors as hyperlinks between documents, enabling a user to quickly access related material. Gutwin et al. (1999) describe Keyphind, which uses keywords from documents as the basic building block for an IR system. Keywords can also be used to enrich the presentation of search results. Hulth (2004) describes Keegle, a system that dynamically provides keyword extracts for web pages returned from a Google search. Andrade and Valencia (1998) present a system that automatically annotates protein function with keywords extracted from the scientific literature that are associated with a given protein.

Text Mining: Applications and Theory edited by Michael W. Berry and Jacob Kogan
© 2010, John Wiley & Sons, Ltd

1.1.1 Keyword extraction methods

Despite their utility for analysis, indexing, and retrieval, most documents do not have assigned keywords. Most existing approaches focus on the manual assignment of keywords by professional curators who may use a fixed taxonomy, or rely on the authors' judgment to provide a representative list. Research has therefore focused on methods to automatically extract keywords from documents as an aid either to suggest keywords for a professional indexer or to generate summary features for documents that would otherwise be inaccessible.

Early approaches to automatically extract keywords focus on evaluating corpus-oriented statistics of individual words. Jones (1972) and Salton et al. (1975) describe positive results of selecting for an index vocabulary the statistically discriminating words across a corpus. Later keyword extraction research applies these metrics to select discriminating words as keywords for individual documents. For example, Andrade and Valencia (1998) base their approach on comparison of word frequency distributions within a text against distributions from a reference corpus.

While some keywords are likely to be evaluated as statistically discriminating within the corpus, keywords that occur in many documents within the corpus are not likely to be selected as statistically discriminating. Corpus-oriented methods also typically operate only on single words. This further limits the measurement of statistically discriminating words because single words are often used in multiple and different contexts.

To avoid these drawbacks, we focus our interest on methods of keyword extraction that operate on individual documents. Such document-oriented methods will extract the same keywords from a document regardless of the current state of a corpus. Document-oriented methods therefore provide context-independent document features, enabling additional analytic methods such as those described in Engel et al. (2009) and Whitney et al. (2009) that characterize changes within a text stream over time. These document-oriented methods are suited to corpora that change, such as collections of published technical abstracts that grow over time or streams of news articles. Furthermore, by operating on a single document, these methods inherently scale to vast collections and can be applied in many contexts to enrich IR systems and analysis tools.

Previous work on document-oriented methods of keyword extraction has combined natural language processing approaches to identify part-of-speech (POS) tags that are combined with supervised learning, machine-learning algorithms, or statistical methods.

Hulth (2003) compares the effectiveness of three term selection approaches: noun-phrase (NP) chunks, n-grams, and POS tags, with four discriminative features of these terms as inputs for automatic keyword extraction using a supervised machine-learning algorithm.

Mihalcea and Tarau (2004) describe a system that applies a series of syntactic filters to identify POS tags that are used to select words to evaluate as keywords. Co-occurrences of the selected words within a fixed-size sliding window are accumulated within a word co-occurrence graph. A graph-based ranking algorithm (TextRank) is applied to rank words based on their associations in the graph, and then top ranking words are selected as keywords. Keywords that are adjacent in the document are combined to form multi-word keywords. Mihalcea and Tarau (2004) report that TextRank achieves its best performance when only nouns and adjectives are selected as potential keywords.

Matsuo and Ishizuka (2004) apply a chi-square measure to calculate how selectively words and phrases co-occur within the same sentences as a particular subset of frequent terms in the document text. The chi-square measure is applied to determine the bias of word co-occurrences in the document text which is then used to rank words and phrases as keywords of the document. Matsuo and Ishizuka (2004) state that the degree of biases is not reliable when term frequency is small. The authors present an evaluation on full text articles and a working example on a 27-page document, showing that their method operates effectively on large documents.

In the following sections, we describe Rapid Automatic Keyword Extraction (RAKE), an unsupervised, domain-independent, and language-independent method for extracting keywords from individual documents. We provide details of the algorithm and its configuration parameters, and present results on a benchmark dataset of technical abstracts, showing that RAKE is more computationally efficient than TextRank while achieving higher precision and comparable recall scores. We then describe a novel method for generating stoplists, which we use to configure RAKE for specific domains and corpora. Finally, we apply RAKE to a corpus of news articles and define metrics for evaluating the exclusivity, essentiality, and generality of extracted keywords, enabling a system to identify keywords that are essential or general to documents in the absence of manual annotations.

1.2 Rapid automatic keyword extraction

In developing RAKE, our motivation has been to develop a keyword extraction method that is extremely efficient, operates on individual documents to enable application to dynamic collections, is easily applied to new domains, and operates well on multiple types of documents, particularly those that do not follow specific grammar conventions. Figure 1.1 contains the title and text for a typical abstract, as well as its manually assigned keywords.

RAKE is based on our observation that keywords frequently contain multiple words but rarely contain standard punctuation or stop words, such as the function words *and*, *the*, and *of*, or other words with minimal lexical meaning. Reviewing the manually assigned keywords for the abstract in Figure 1.1, there is only one keyword that contains a stop word (*of* in *set of natural numbers*). Stop words are typically dropped from indexes within IR systems and not included in various text analyses as they are considered to be uninformative or meaningless. This reasoning is based on the expectation that such words are too frequently and broadly used to aid users in their analyses or search tasks. Words that do

Compatibility of systems of linear constraints over the set of natural numbers

Criteria of compatibility of a system of linear Diophantine equations, strict inequations, and nonstrict inequations are considered. Upper bounds for components of a minimal set of solutions and algorithms of construction of minimal generating sets of solutions for all types of systems are given. These criteria and the corresponding algorithms for constructing a minimal supporting set of solutions can be used in solving all the considered types of systems and systems of mixed types.

Manually assigned keywords:
linear constraints, set of natural numbers, linear Diophantine equations, strict inequations, nonstrict inequations, upper bounds, minimal generating sets

Figure 1.1 A sample abstract from the Inspec test set and its manually assigned keywords.

carry meaning within a document are described as content bearing and are often referred to as content words.

The input parameters for RAKE comprise a list of stop words (or stoplist), a set of phrase delimiters, and a set of word delimiters. RAKE uses stop words and phrase delimiters to partition the document text into candidate keywords, which are sequences of content words as they occur in the text. Co-occurrences of words within these candidate keywords are meaningful and allow us to identify word co-occurrence without the application of an arbitrarily sized sliding window. Word associations are thus measured in a manner that automatically adapts to the style and content of the text, enabling adaptive and fine-grained measurement of word co-occurrences that will be used to score candidate keywords.

1.2.1 Candidate keywords

RAKE begins keyword extraction on a document by parsing its text into a set of candidate keywords. First, the document text is split into an array of words by the specified word delimiters. This array is then split into sequences of contiguous words at phrase delimiters and stop word positions. Words within a sequence are assigned the same position in the text and together are considered a candidate keyword.

Figure 1.2 shows the candidate keywords in the order that they are parsed from the sample technical abstract shown in Figure 1.1. The candidate keyword

Compatibility – systems – linear constraints – set – natural numbers – Criteria – compatibility – system – linear Diophantine equations – strict inequations – nonstrict inequations – Upper bounds – components – minimal set – solutions – algorithms – minimal generating sets – solutions – systems – criteria – corresponding algorithms – constructing – minimal supporting set – solving – systems – systems

Figure 1.2 Candidate keywords parsed from the sample abstract.

linear Diophantine equations begins after the stop word *of* and ends with a comma. The following word *strict* begins the next candidate keyword *strict inequations*.

1.2.2 Keyword scores

After every candidate keyword is identified and the graph of word co-occurrences (shown in Figure 1.3) is complete, a score is calculated for each candidate keyword and defined as the sum of its member word scores. We evaluated several metrics for calculating word scores, based on the degree and frequency of word vertices in the graph: (1) word frequency (*freq(w)*), (2) word degree (*deg(w)*), and (3) ratio of degree to frequency (*deg(w)/freq(w)*).

The metric scores for each of the content words in the sample abstract are listed in Figure 1.4. In summary, *deg(w)* favors words that occur often and in longer candidate keywords; *deg(minimal)* scores higher than *deg(systems)*. Words that occur frequently regardless of the number of words with which they co-occur are favored by *freq(w)*; *freq(systems)* scores higher than *freq(minimal)*. Words that predominantly occur in longer candidate keywords are favored by *deg(w)/freq(w)*; *deg(diophantine)/freq(diophantine)* scores higher than *deg(linear)/freq(linear)*. The score for each candidate keyword is computed as the sum of its member

	algorithms	bounds	compatibility	components	constraints	constructing	corresponding	criteria	diophantine	equations	generating	inequations	linear	minimal	natural	nonstrict	numbers	set	sets	solving	strict	supporting	system	systems	upper
algorithms	2						1																		
bounds		1																							1
compatibility			2																						
components				1																					
constraints					1								1												
constructing						1																			
corresponding	1						1																		
criteria								2																	
diophantine									1	1			1												
equations									1	1			1												
generating											1			1					1						
inequations												2				1					1				
linear					1				1	1			2												
minimal											1			3				2	1			1			
natural															1		1								
nonstrict												1				1									
numbers															1		1								
set														2				3				1			
sets											1			1					1						
solving																				1					
strict												1									1				
supporting														1				1				1			
system																							1		
systems																								4	
upper		1																							1

Figure 1.3 The word co-occurrence graph for content words in the sample abstract.

	algorithms	bounds	compatibility	components	constraints	constructing	corresponding	criteria	diophantine	equations	generating	inequations	linear	minimal	natural	nonstrict	numbers	set	sets	solving	strict	supporting	system	systems	upper
deg(w)	3	2	2	1	2	1	2	2	3	3	3	4	5	8	2	2	2	6	3	1	2	3	1	4	2
freq(w)	2	1	2	1	1	1	1	2	1	1	1	2	2	3	1	1	1	3	1	1	1	1	1	4	1
deg(w) / freq(w)	1.5	2	1	1	2	1	2	1	3	3	3	2	2.5	2.7	2	2	2	2	3	1	2	3	1	1	2

Figure 1.4 Word scores calculated from the word co-occurrence graph.

minimal generating sets (8.7), linear diophantine equations (8.5), minimal supporting set (7.7), minimal set (4.7), linear constraints (4.5), natural numbers (4), strict inequations (4), nonstrict inequations (4), upper bounds (4), corresponding algorithms (3.5), set (2), algorithms (1.5), compatibility (1), systems (1), criteria (1), system (1), components (1),constructing (1), solving (1)

Figure 1.5 Candidate keywords and their calculated scores.

word scores. Figure 1.5 lists each candidate keyword from the sample abstract using the metric *deg(w)/freq(w)* to calculate individual word scores.

1.2.3 Adjoining keywords

Because RAKE splits candidate keywords by stop words, extracted keywords do not contain interior stop words. While RAKE has generated strong interest due to its ability to pick out highly specific terminology, an interest was also expressed in identifying keywords that contain interior stop words such as *axis of evil*. To find these RAKE looks for pairs of keywords that adjoin one another at least twice in the same document and in the same order. A new candidate keyword is then created as a combination of those keywords and their interior stop words. The score for the new keyword is the sum of its member keyword scores.

It should be noted that relatively few of these linked keywords are extracted, which adds to their significance. Because adjoining keywords must occur twice in the same order within the document, their extraction is more common on texts that are longer than short abstracts.

1.2.4 Extracted keywords

After candidate keywords are scored, the top T scoring candidates are selected as keywords for the document. We compute T as one-third the number of words in the graph, as in Mihalcea and Tarau (2004).

The sample abstract contains 28 content words, resulting in $T = 9$ keywords. Table 1.1 lists the keywords extracted by RAKE compared to the sample abstract's manually assigned keywords. We use the statistical measures precision, recall and F-measure to evaluate the accuracy of RAKE. Out of nine keywords extracted, six are true positives; that is, they exactly match six of the manually assigned keywords. Although *natural numbers* is similar to the assigned

Table 1.1 Comparison of keywords extracted by RAKE to manually assigned keywords for the sample abstract.

Extracted by RAKE	Manually assigned
minimal generating sets	minimal generating sets
linear diophantine equations	linear Diophantine equations
minimal supporting set	
minimal set	
linear constraints	linear constraints
natural numbers	
strict inequations	strict inequations
nonstrict inequations	nonstrict inequations
upper bounds	upper bounds
	set of natural numbers

keyword *set of natural numbers*, for the purposes of the benchmark evaluation it is considered a miss. There are therefore three false positives in the set of extracted keywords, resulting in a precision of 67%. Comparing the six true positives within the set of extracted keywords to the total of seven manually assigned keywords results in a recall of 86%. Equally weighting precision and recall generates an F-measure of 75%.

1.3 Benchmark evaluation

To evaluate performance we tested RAKE against a collection of technical abstracts used in the keyword extraction experiments reported in Hulth (2003) and Mihalcea and Tarau (2004), mainly for the purpose of allowing direct comparison with their results.

1.3.1 Evaluating precision and recall

The collection consists of 2000 Inspec abstracts for journal papers from Computer Science and Information Technology. The abstracts are divided into a training set with 1000 abstracts, a validation set with 500 abstracts, and a testing set with 500 abstracts. We followed the approach described in Mihalcea and Tarau (2004), using the testing set for evaluation because RAKE does not require a training set. Extracted keywords for each abstract are compared against the abstract's associated set of manually assigned uncontrolled keywords.

Table 1.2 details RAKE's performance using a generated stoplist, Fox's stoplist (Fox 1989), and T as one-third the number of words in the graph. For each method, which corresponds to a row in the table, the following information is shown: the total number of extracted keywords and mean per abstract; the number of correct extracted keywords and mean per abstract; precision; recall; and F-measure. Results published within Hulth (2003) and Mihalcea and Tarau

Table 1.2 Results of automatic keyword extraction on 500 abstracts in the Inspec test set using RAKE, TextRank (Mihalcea and Tarau 2004) and supervised learning (Hulth 2003).

	Extracted keywords		Correct keywords				
Method	Total	Mean	Total	Mean	Precision	Recall	F-measure
RAKE ($T = 0.33$)							
KA stoplist ($df > 10$)	6052	12.1	2037	4.1	**33.7**	41.5	**37.2**
Fox stoplist	7893	15.8	2054	4.2	26	42.2	32.1
TextRank							
Undirected, co-occ. window = 2	6784	13.6	2116	4.2	31.2	43.1	36.2
Undirected, co-occ. window = 3	6715	13.4	1897	3.8	28.2	38.6	32.6
(Hulth 2003)							
Ngram with tag	7815	15.6	1973	3.9	25.2	**51.7**	33.9
NP chunks with tag	4788	9.6	1421	2.8	29.7	37.2	33
Pattern with tag	7012	14	1523	3	21.7	39.9	28.1

the, and, of, a, in, is, for, to, we, this, are, with, as, on, it, an, that, which, by, using, can, paper, from, be, based, has, was, have, or, at, such, also, but, results, proposed, show, new, these, used, however, our, were, when, one, not, two, study, present, its, sub, both, then, been, they, all, presented, if, each, approach, where, may, some, more, use, between, into, 1, under, while, over, many, through, addition, well, first, will, there, propose, than, their, 2, most, sup, developed, particular, provides, including, other, how, without, during, article, application, only, called, what, since, order, experimental, any

Figure 1.6 Top 100 words in the generated stoplist.

(2004) are included for comparison. The highest values for precision, recall, and F-measure are shown in bold. As noted, perfect precision is not possible with any of the techniques as the manually assigned keywords do not always appear in the abstract text. The highest precision and F-measure are achieved using RAKE with a generated stoplist based on keyword adjacency, a subset of which is listed in Figure 1.6. With this stoplist RAKE yields the best results in terms of F-measure and precision, and provides comparable recall. With Fox's stoplist, RAKE achieves a high recall while experiencing a drop in precision.

1.3.2 Evaluating efficiency

Because of increasing interest in energy conservation in large data centers, we also evaluated the computational cost associated with extracting keywords with RAKE and TextRank. TextRank applies syntactic filters to a document text to identify content words and accumulates a graph of word co-occurrences in a window size of 2. A rank for each word in the graph is calculated through a series of iterations until convergence below a threshold is achieved.

We set TextRank's damping factor $d = 0.85$ and its convergence threshold to 0.0001, as recommended in Mihalcea and Tarau (2004). We do not have access to the syntactic filters referenced in Mihalcea and Tarau (2004), so were unable to evaluate their computational cost.

To minimize disparity, all parsing stages in the respective extraction methods are identical, TextRank accumulates co-occurrences in a window of size 2, and RAKE accumulates word co-occurrences within candidate keywords. After co-occurrences are tallied, the algorithms compute keyword scores according to their respective methods. The benchmark was implemented in Java and executed in the Java SE Runtime Environment (JRE) 6 on a Dell Precision T7400 workstation.

We calculated the total time for RAKE and TextRank (as an average over 100 iterations) to extract keywords from the Inspec testing set of 500 abstracts, after the abstracts were read from files and loaded in memory. RAKE extracted keywords from the 500 abstracts in 160 milliseconds. TextRank extracted keywords in 1002 milliseconds, over 6 times the time of RAKE.

Referring to Figure 1.7, we can see that as the number of content words for a document increases, the performance advantage of RAKE over TextRank increases. This is due to RAKE's ability to score keywords in a single pass whereas TextRank requires repeated iterations to achieve convergence on word ranks.

Based on this benchmark evaluation, it is clear that RAKE effectively extracts keywords and outperforms the current state of the art in terms of precision, efficiency, and simplicity. As RAKE can be put to use in many different systems and applications, in the next section we discuss a method for stoplist generation that may be used to configure RAKE on particular corpora, domains, and languages.

1.4 Stoplist generation

Stoplists are widely used in IR and text analysis applications. However, there is remarkably little information describing methods for their creation. Fox (1989) presents an analysis of stoplists, noting discrepancies between stated conventions and actual instances and implementations of stoplists. The lack of technical rigor associated with the creation of stoplists presents a challenge when comparing text analysis methods. In practice, stoplists are often based on common function words and hand-tuned for particular applications, domains, or specific languages.

We evaluated the use of term frequency as a metric for automatically selecting words for a stoplist. Table 1.3 lists the top 50 words by term frequency in the training set of abstracts in the benchmark dataset. Additional metrics shown for each word are document frequency, adjacency frequency, and keyword frequency. Adjacency frequency reflects the number of times the word occurred adjacent to

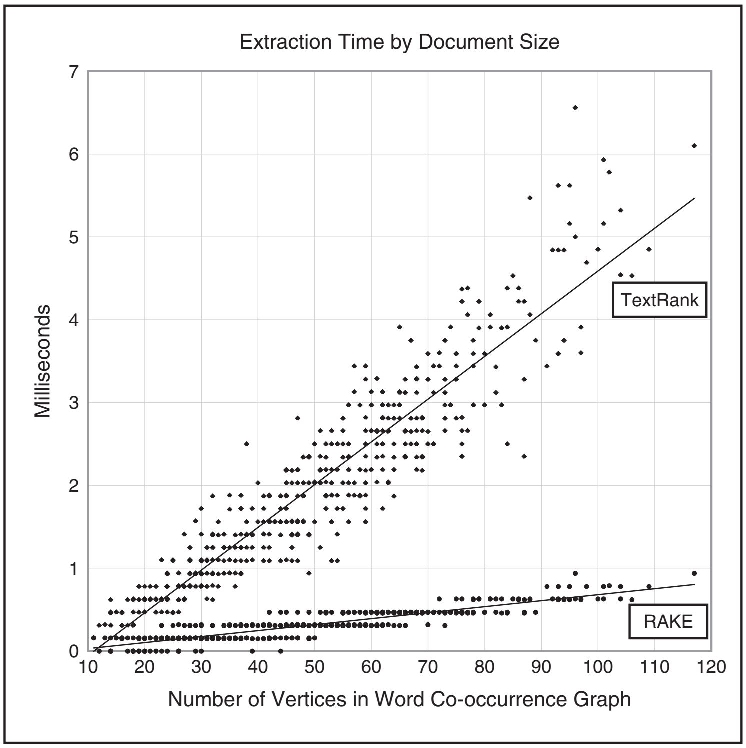

Figure 1.7 Comparison of TextRank and RAKE extraction times on individual documents.

an abstract's keywords. Keyword frequency reflects the number of times the word occurred within an abstract's keywords.

Looking at the top 50 frequent words, in addition to the typical function words, we can see that *system*, *control*, and *method* are highly frequent within technical abstracts and highly frequent within the abstracts' keywords. Selecting solely by term frequency will therefore cause content-bearing words to be added to the stoplist, particularly if the corpus of documents is focused on a particular domain or topic. In those circumstances, selecting stop words by term frequency presents a risk of removing important content-bearing words from analysis.

We therefore present the following method for automatically generating a stoplist from a set of documents for which keywords are defined. The algorithm is based on the intuition that words adjacent to, and not within, keywords are less likely to be meaningful and therefore are good choices for stop words.

To generate our stoplist we identified for each abstract in the Inspec training set the words occurring adjacent to words in the abstract's uncontrolled keyword list. The frequency of each word occurring adjacent to a keyword was accumulated across the abstracts. Words that occurred more frequently within keywords than adjacent to them were excluded from the stoplist.

Table 1.3 The 50 most frequent words in the Inspec training set listed in descending order by term frequency.

Word	Term frequency	Document frequency	Adjacency frequency	Keyword frequency
the	8611	978	3492	3
of	5546	939	1546	68
and	3644	911	2104	23
a	3599	893	1451	2
to	3000	879	792	10
in	2656	837	1402	7
is	1974	757	1175	0
for	1912	767	951	9
that	1129	590	330	0
with	1065	577	535	3
are	1049	576	555	1
this	964	581	645	0
on	919	550	340	8
an	856	501	332	0
we	822	388	731	0
by	773	475	283	0
as	743	435	344	0
be	595	395	170	0
it	560	369	339	13
system	**507**	**255**	**86**	**202**
can	452	319	250	0
based	451	293	168	15
from	447	309	187	0
using	428	282	260	0
control	**409**	**166**	**12**	**237**
which	402	280	285	0
paper	398	339	196	1
systems	**384**	**194**	**44**	**191**
method	**347**	**188**	**78**	**85**
data	**347**	**159**	**39**	**131**
time	**345**	**201**	**24**	**95**
model	**343**	**157**	**37**	**122**
information	**322**	**153**	**18**	**151**
or	315	218	146	0
s	314	196	27	0
have	301	219	149	0
has	297	225	166	0
at	296	216	141	0
new	294	197	93	4
two	287	205	83	5

(*continued overleaf*)

Table 1.3 (*Continued*)

Word	Term frequency	Document frequency	Adjacency frequency	Keyword frequency
algorithm	**267**	**123**	**36**	**96**
results	262	221	129	14
used	262	204	92	0
was	254	125	161	0
these	252	200	93	0
also	251	219	139	0
such	249	198	140	0
problem	**234**	**137**	**36**	**55**
design	**225**	**110**	**38**	**68**

To evaluate this method of generating stoplists, we created six stoplists, three of which select words for the stoplist by term frequency (TF), and three which select words by term frequency but also exclude words from the stoplist whose keyword frequency was greater than their keyword adjacency frequency. We refer to this latter set of stoplists as keyword adjacency (KA) stoplists since they primarily include words that are adjacent to and not within keywords.

Table 1.4 Comparison of RAKE performance using stoplists based on term frequency (TF) and keyword adjacency (KA).

Method	Stoplist size	Extracted keywords		Correct keywords		Precision	Recall	*F*-measure
		Total	Mean	Total	Mean			
RAKE ($T = 0.33$)								
TF stoplist (*df* > 10)	1347	3670	7.3	606	1.2	16.5	12.3	14.1
TF stoplist (*df* > 25)	527	5563	11.1	1032	2.1	18.6	21.0	19.7
TF stoplist (*df* > 50)	205	7249	14.5	1520	3.0	21.0	30.9	25.0
RAKE ($T = 0.33$)								
KA stoplist (*df* > 10)	763	6052	12.1	2037	4.1	33.7	41.5	37.2
KA stoplist (*df* > 25)	325	7079	14.2	2103	4.3	29.7	42.8	35.1
KA stoplist (*df* > 50)	147	8013	16.0	2117	4.3	26.4	43.1	32.8

Each of the stoplists was set as the input stoplist for RAKE, which was then run on the testing set of the Inspec corpus of technical abstracts. Table 1.4 lists the precision, recall, and F-measure for the keywords extracted by each of these runs. The KA stoplists generated by our method outperformed the TF stoplists generated by term frequency. A notable difference between results achieved using the two types of stoplists is evident in Table 1.4: the F-measure improves as more words are added to a KA stoplist, whereas when more words are added to a TF stoplist the F-measure degrades. Furthermore, the best TF stoplist underperforms the worst KA stoplist. This verifies that our algorithm for generating stoplists is adding the right stop words and excluding content words from the stoplist.

Because the generated KA stoplists leverage manually assigned keywords, we envision that an ideal application would be within existing digital libraries or IR systems and collections where defined keywords exist or are easily identified for a subset of the documents. Stoplists only need to be generated once for particular domains, enabling RAKE to be applied to new and future articles, facilitating the annotation and indexing of new documents.

1.5 Evaluation on news articles

While we have shown that a simple set of configuration parameters enables RAKE to efficiently extract keywords from individual documents, it is worth investigating how well extracted keywords represent the essential content within a corpus of documents for which keywords have not been manually assigned. The following section presents results on application of RAKE to the Multi-Perspective Question Answering (MPQA) Corpus (CERATOPS 2009).

1.5.1 The MPQA Corpus

The MPQA Corpus consists of 535 news articles provided by the Center for the Extraction and Summarization of Events and Opinions in Text (CERATOPS). Articles in the MPQA Corpus are from 187 different foreign and US news sources and date from June 2001 to May 2002.

1.5.2 Extracting keywords from news articles

We extracted keywords from title and text fields of documents in the MPQA Corpus and set a minimum document threshold of two because we are interested in keywords that are associated with multiple documents.

Candidate keyword scores were based on word scores as *deg(w)/freq(w)* and as *deg(w)*. Calculating word scores as *deg(w)/freq(w)*, RAKE extracted 517 keywords referenced by an average of 4.9 documents. Calculating word scores as *deg(w)*, RAKE extracted 711 keywords referenced by an average of 8.1 documents.

This difference in average number of referenced document counts is the result of longer keywords having lower frequency across documents. The metric *deg(w)/freq(w)* favors longer keywords and therefore results in extracted keywords that occur in fewer documents in the MPQA Corpus.

In many cases a subject is occasionally presented in its long form and more frequently referenced in its shorter form. For example, referring to Table 1.5, *kyoto protocol on climate change* and *1997 kyoto protocol* occur less frequently than the shorter *kyoto protocol*. Because our interest in the analysis of news articles is to connect articles that reference related content, we set RAKE to score words by *deg(w)* in order to favor shorter keywords that occur across more documents.

Because most documents are unique within any given corpus, we expect to find variability in what documents are essentially about as well as how each document represents specific subjects. While some documents may be primarily about the *kyoto protocol*, *greenhouse gas emissions*, and *climate change*, other documents may only make references to those subjects. Documents in the former set will likely have *kyoto protocol*, *greenhouse gas emissions*, and *climate change* extracted as keywords whereas documents in the latter set will not.

In many applications, users have a desire to capture all references to extracted keywords. For the purposes of evaluating extracted keywords, we accumulate

Table 1.5 Keywords extracted with word scores by *deg(w)* and *deg(w)/freq(w)*.

	Scored by *deg(w)*		Scored by *deg(w)/freq(w)*	
Keyword	*edf(w)*	*rdf(w)*	*edf(w)*	*rdf(w)*
kyoto protocol legally obliged developed countries	2	2	2	2
eu leader urge russia to ratify kyoto protocol	2	2	2	2
kyoto protocol on climate change	2	2	2	2
ratify kyoto protocol	2	2	2	2
kyoto protocol requires	2	2	2	2
1997 kyoto protocol	2	4	4	4
kyoto protocol	31	44	7	44
kyoto	10	12	–	–
kyoto accord	3	3	–	–
kyoto pact	2	3	–	–
sign kyoto protocol	2	2	–	–
ratification of the kyoto protocol	2	2	–	–
ratify the kyoto protocol	2	2	–	–
kyoto agreement	2	2	–	–

counts on how often each extracted keyword is referenced by documents in the corpus. The referenced document frequency of a keyword, *rdf(k)*, is the number of documents in which the keyword occurred as a candidate keyword. The extracted document frequency of a keyword, *edf(k)*, is the number of documents from which the keyword was extracted.

A keyword that is extracted from all of the documents in which it is referenced can be characterized as *exclusive* or *essential*, whereas a keyword that is referenced in many documents but extracted from a few may be characterized as *general*. Comparing the relationship of *edf(k)* and *rdf(k)* allows us to characterize the exclusivity of a particular keyword. We therefore define keyword exclusivity *exc(k)* as shown in Equation (1.1):

$$exc(k) = \frac{edf(k)}{rdf(k)}. \tag{1.1}$$

Of the 711 extracted keywords, 395 have an exclusivity score of 1, indicating that they were extracted from every document in which they were referenced. Within that set of 395 exclusive keywords, some occur in more documents than others and can therefore be considered more essential to the corpus of documents. In order to measure how essential a keyword is, we define the essentiality of a keyword, *ess(k)*, as shown in Equation (1.2):

$$ess(k) = exc(k) \times edf(k). \tag{1.2}$$

Figure 1.8 lists the top 50 essential keywords extracted from the MPQA corpus, listed in descending order by their *ess(k)* scores. According to CERATOPS, the MPQA corpus comprises 10 primary topics, listed in Table 1.6, which are well represented by the 50 most essential keywords as extracted and ranked by RAKE.

In addition to keywords that are essential to documents, we can also characterize keywords by how general they are to the corpus. In other words, how

united states (32), human rights (24), kyoto protocol (22), international space station (18), mugabe (16), space station (14), human rights report (12), greenhouse gas emissions (12), chavez (11), taiwan issue (11), president chavez (10), human rights violations (10), president bush (10), palestinian people (10), prisoners of war (9), president hugo chavez (9), kyoto (8), taiwan (8), israeli government (8), hugo chavez (8), climate change (8), space (8), axis of evil (7), president fernando henrique cardoso (7), palestinian (7), palestinian territories (6), taiwan strait (6), russian news agency interfax (6), prisoners (6), taiwan relations act (6), president robert mugabe (6), presidential election (6), geneva convention (5), palestinian authority (5), venezuelan president hugo chavez (5), chinese president jiang zemin (5), opposition leader morgan tsvangirai (5), french news agency afp (5), bush (5), north korea (5), camp x-ray (5), rights (5), election (5), mainland china (5), al qaeda (5), president (4), south africa (4), global warming (4), bush administration (4), mdc leader (4)

Figure 1.8 Top 50 essential keywords from the MPQA Corpus, with corresponding ess(k) *score in parentheses.*

Table 1.6 MPQA Corpus topics and definitions.

Topic	Description
argentina	Economic collapse in Argentina
axisofevil	Reaction to President Bush's 2002 State of the Union Address
guantanamo	US holding prisoners in Guantanamo Bay
humanrights	Reaction to US State Department report on human rights
kyoto	Ratification of Kyoto Protocol
mugabe	2002 Presidential election in Zimbabwe
settlements	Israeli settlements in Gaza and West Bank
spacestation	Space missions of various countries
taiwan	Relations between Taiwan and China
venezuela	Presidential coup in Venezuela

government (147), countries (141), people (125), world (105), report (91), war (85), united states (79), china (71), president (69), iran (60), bush (56), japan (50), law (44), peace (44), policy (43), officials (43), israel (41), zimbabwe (39), taliban (36), prisoners (35), opposition (35), plan (35), president george (34), axis (34), administration (33), detainees (32), treatment (32), states (30), european union (30), palestinians (30), election (29), rights (28), international community (27), military (27), argentina (27), america (27), guantanamo bay (26), official (26), weapons (24), source (24), eu (23), attacks (23), united nations (22), middle east (22), bush administration (22), human rights (21), base (20), minister (20), party (19), north korea (18)

Figure 1.9 Top 50 general keywords from the MPQA Corpus, with corresponding gen(k) *score in parentheses.*

often was a keyword referenced by documents from which it was not extracted? In this case we define generality of a keyword, *gen(k)*, as shown in Equation (1.3):

$$gen(k) = rdf(k) \times (1.0 - exc(k)). \tag{1.3}$$

Figure 1.9 lists the top 50 general keywords extracted from the MPQA corpus, listed in descending order by their *gen(k)* scores. It should be noted that general keywords and essential keywords are not mutually exclusive. Within the top 50 for both metrics, there are several shared keywords: *united states*, *president*, *bush*, *prisoners*, *election*, *rights*, *bush administration*, *human rights*, and *north korea*. Keywords that are both highly essential and highly general are essential to a set of documents within the corpus but also referenced by a significantly greater number of documents within the corpus than other keywords.

1.6 Summary

We have shown that our automatic keyword extraction technology, RAKE, achieves higher precision and similar recall in comparison to existing techniques.

In contrast to methods that depend on natural language processing techniques to achieve their results, RAKE takes a simple set of input parameters and automatically extracts keywords in a single pass, making it suitable for a wide range of documents and collections.

Finally, RAKE's simplicity and efficiency enable its use in many applications where keywords can be leveraged. Based on the variety and volume of existing collections and the rate at which documents are created and collected, RAKE provides advantages and frees computing resources for other analytic methods.

1.7 Acknowledgements

This work was supported by the National Visualization and Analytics Center™ (NVAC™), which is sponsored by the US Department of Homeland Security Program and located at the Pacific Northwest National Laboratory (PNNL), and by Laboratory Directed Research and Development at PNNL. PNNL is managed for the US Department of Energy by Battelle Memorial Institute under Contract DE-AC05-76RL01830.

We also thank Anette Hulth, for making available the dataset used in her experiments.

References

Andrade M and Valencia A 1998 Automatic extraction of keywords from scientific text: application to the knowledge domain of protein families. *Bioinformatics* **14**(7), 600–607.

CERATOPS 2009 MPQA Corpus http://www.cs.pitt.edu/mpqa/ceratops/corpora.html.

Engel D, Whitney P, Calapristi A and Brockman F 2009 Mining for emerging technologies within text streams and documents. *Proceedings of the Ninth SIAM International Conference on Data Mining*. Society for Industrial and Applied Mathematics.

Fox C 1989 A stop list for general text. *ACM SIGIR Forum*, vol. 24, pp. 19–21. ACM, New York, USA.

Gutwin C, Paynter G, Witten I, Nevill-Manning C and Frank E 1999 Improving browsing in digital libraries with keyphrase indexes. *Decision Support Systems* **27**(1–2), 81–104.

Hulth A 2003 Improved automatic keyword extraction given more linguistic knowledge. *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing*, vol. 10, pp. 216–223 Association for Computational Linguistics, Morristown, NJ, USA.

Hulth A 2004 *Combining machine learning and natural language processing for automatic keyword extraction*. Stockholm University, Faculty of Social Sciences, Department of Computer and Systems Sciences (together with KTH).

Jones K 1972 A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation* **28**(1), 11–21.

Jones S and Paynter G 2002 Automatic extraction of document keyphrases for use in digital libraries: evaluation and applications. *Journal of the American Society for Information Science and Technology*.

Matsuo Y and Ishizuka M 2004 Keyword extraction from a single document using word co-occurrence statistical information. *International Journal on Artificial Intelligence Tools* **13**(1), 157–169.

Mihalcea R and Tarau P 2004 Textrank: Bringing order into texts. In *Proceedings of EMNLP 2004* (ed. Lin D and Wu D), pp. 404–411. Association for Computational Linguistics, Barcelona, Spain.

Salton G, Wong A and Yang C 1975 A vector space model for automatic indexing. *Communications of the ACM* **18**(11), 613–620.

Whitney P, Engel D and Cramer N 2009 Mining for surprise events within text streams. *Proceedings of the Ninth SIAM International Conference on Data Mining*, pp. 617–627. Society for Industrial and Applied Mathematics.