

# Indexing Text Documents Based on Topic Identification

Manonton Butarbutar and Susan McRoy

Department of Electrical Engineering and Computer Science,  
University of Wisconsin – Milwaukee, USA  
{anton, mcroy}@cs.uwm.edu

**Abstract.** This work provides algorithms and heuristics to index text documents by determining important topics in the documents. To index text documents, the work provides algorithms to generate topic candidates, determine their importance, detect similar and synonym topics, and to eliminate incoherent topics. The indexing algorithm uses topic frequency to determine the importance and the existence of the topics. Repeated phrases are topic candidates. For example, since the phrase ‘index text documents’ occurs three times in this abstract, the phrase is one of the topics of this abstract. It is shown that this method is more effective than either a simple word count model or approaches based on term weighting.

## 1 Introduction

One of the key problems in indexing texts by topics is to determine which set of words constitutes a topic. This work provides algorithms to identify topics by determining which sets of words appear together within a certain proximity and how often those words appear together in the texts.

To count the frequencies of topics in texts accurately, a system must be able to detect topic repetition, similarity, synonymy, parallelism, and implicit references. However, these factors are not all equally important. We have found that topic repetition and topic similarity are the most useful and are sufficient to produce good indices.

The work described in this paper provides algorithms to detect similar topics in texts. For example, if a text contains phrase ‘a native American history book’ and ‘this book is about the history of native Americans’, *iIndex*, the implementation of this work, detects that both phrases are similar, counts the frequency of topic ‘native American history book’ as two, and makes the phrase a candidate topic.

The *iIndex* system also detects topics that are synonyms, sums their frequencies, and represents them together as one synonym meaning. This is important because the same topic can be expressed in several different ways. For example, the phrases ‘topic identification’, ‘topic determination’, ‘topic discovery’, ‘finding topic’, ‘locating topic’, and ‘topic spotting’ can all serve as synonyms.

The *iIndex* system extracts shorter and best phrases from texts to serve as candidates. For example, the phrase ‘blood pressure’ is selected over ‘pressure of the blood’. In addition, unlike previous approaches, such as [WEK01], *iIndex* extracts

any important phrases from texts, not just simple noun phrases. For example, terms such as ‘high blood pressure has no symptoms’ and ‘blood pressure should be monitored more frequently’ are extracted from texts, terms that would be missed by a noun phrase indexer.

The major contributions of this work are techniques and algorithms to determine and to order the most important topics in text documents and to index text documents efficiently based on important topics in the texts without employing complex English parsers. It efficiently solves the problem of finding important topics in texts, the problem which requires exponential computation time, by carefully selecting subsets of the problem that are practical to compute yet useful and cover 97% of the problem domain. The approach also provides a method that defines topic synonyms with inference complexity  $O(\log n)$  or better.

## 2 Background

Over the past 30 years, a number of approaches to information retrieval have been developed, including word-based ranking, link-based ranking, phrase-based indexing, concept-based indexing, rule-based indexing, and logical inference-based indexing [Ha92, Sa89].

The closest work to iIndex is that of Johnson [JCD+99] and Aronson [ABC+00]; iIndex, however, applies a much richer set of techniques and heuristics than these two approaches. For example, iIndex allows one to configure the maximum number of words in a phrase (default is 10), whereas in prior work the phrase size has been fixed (3 in Johnson and 6 in Aronson). The iIndex system also uses *limited stemming* (which we will describe later) as opposed to standard stemming with its weakness described in Section 3.3. iIndex also considers complete text documents as its input, rather than just the title and the abstract of documents as in Aronson. Finally, iIndex uses a set of configurable matching techniques as opposed to just one technique as in Johnson.

Fagan [Fa87] examined the effectiveness of using phrases for document retrieval. His phrase construction is limited to 2-word phrases and uses the standard stemming. The precision improvement by the use of phrase vary from -1.8% to 20.1%. Kelledy and Smeaton [KS97] concluded that the use of phrases improves the precision of information retrieval. They use upto 3-word phases and employes standard stemming. They only use phrases that appear in at least 25 different documents, whereas iIndex uses any phrases that are repeated in any document. Consequently, they will miss phrases that are repeated in one document such as ‘limited stemming’ in this document. They do not consider phrase variants ‘department of defence’ and ‘defence department’ are equivalent, whereas iIndex does. Mitra et al. [MBSC97] repeated the experiment by Fagan with a larger set of about 250,000 documents. They also use 2-word phrases that appear in at least 25 documents, employ standard stemming, and ignore word order. They concluded that the use of phrases does not have significant effect on the precision of the high rank retrieval results but useful for the low rank results.

The work by Wacholder [WEK01] is also related, however this work, as mentioned above, indexes only simple noun phrases whereas iIndex considers all types of

phrases. Moreover, Wacholder ranks the topics by the frequency of the head of the noun, which is a single word, whereas iIndex ranks the topics by the frequency of the topic (phrase) as a whole.

Woods [Wo97] provides another approach to topic identification. Unlike iIndex it does not use frequency in determining topic rankings.

### 3 Indexing by Topic

A *topic* is a set of words that has meaning. A topic is normally a phrase. Topics are *determined* by detecting which set of words appears together within certain proximity and how often those words appear together in the documents. The more frequent the set of words in the document, the better the chance that set of words represents an important concept or topic in the document. Generally, the more (significant) words in a topic the more specific the topic is. We would like to find topics that are as specific as possible. Similar topics are grouped (and later stored) by a process that we call *topic canonization*. This process involves converting the words in a phrase to their base forms and then ordering the words alphabetically. The resulting phrase is called the *canonical phrase*. We discuss our methods for determining topic length, topic proximity, and topic frequency below.

A *sentence* or a *phrase* is a string of characters between topic separators. *Topic separators* are special characters such as period, semicolon, question mark, and exclamation mark that separate one topic from another. A *word* is a string of characters consisting of only a-z, A-Z, and 0-9. The approach ignores tokens that are numbers, hyphens, possessive apostrophes and blank characters.

The *topic length* is the number of *significant words* that constitute a topic (sentence or phrase.) Significant words are those that have not been predefined as stop words. A *stop word* is high-frequency word that has no significant meaning in a phrase [Sa89]. iIndex uses 184 stop words. They are manually selected as follows: all single characters from a to z, all pronouns, terms frequently used as variable names such as *t1*, *t2*, *s1*, *s2*, and words that were selected manually, after evaluating the results of indexing several documents using iIndex.

The maximum and minimum values for topic length are configurable parameters of iIndex (discussed in Section 4). iIndex also provides default settings. The default value for maximum length of topics is 10 and the minimum length is 2. These maximum and minimum were selected because it has been reported that the average length of large queries to a major search engine (Alta Vista) is 2.3 words [SHMM98].

*Topic proximity* is the maximum distance of words apart that constitute a topic. For example, phrase 'a topic must be completely within a sentence' is about 'sentence topic' and both words are apart by 6 positions. For this example, the topic proximity is 6.

The *topic frequency* or reference count is the number of times that a topic, similar topic, or synonymous topic is repeated in the document. In our approach, the importance of a topic is measured by its frequency. A topic is *relevant* to a unit of a document if the topic is referenced more than once in the unit. A *unit* of a document can be the whole document, a section, or a paragraph.

### 3.1 Indexing Algorithm

Given a set of documents  $D$ , find a set of  $w$ -word topics that are repeated  $r$  times in the documents. The words that constitute a topic should not be separated by more than  $p$  positions.

For example, given document  $D = \text{"abcdbc"}$ , where each of the letters represents a word, the list of phrases of any 2 words of at most 1 position apart is  $\{\text{ab, bc, cd, db, bc}\}$ . Each of the phrases has frequency 1 except phrase 'bc' with frequency 2. The phrases with the highest frequency are the most important topic. In this case, the only topic recognized for this document is 'bc', a topic with frequency of at least 2.

Let  $D$  be the set of all documents. Let  $u$  be a unit of a document  $d$  in  $D$ . By default,  $u$  represents the whole document. Let  $X$  be the *index* of  $D$ , which is the set of topics that are repeated at least  $r$  times in  $u$ . Each index entry  $x$  in  $X$  represents a relation between topic  $t$ , unit  $u$ , and the frequency of  $t$  in  $u$  denoted as  $x(t, u, f)$ . The whole index is represented by  $X(T, U, F)$  where  $T$  is the set of all topics in  $D$ ,  $U$  is the set of all units in  $D$ , and  $F$  is a set of integers. By definition,  $\{x(t, u, f1)\}$  union  $\{x(t, u, f2)\} = \{x(t, u, f1+f2)\}$  i.e. we sum the frequencies of  $t$  in  $u$ . The frequency of topic  $t$  in unit  $u$  is denoted by  $x(f)$  for a given index entry  $x(t, u, f)$ .

#### Algorithm 1 Indexing Algorithm

1. For each  $u$  in  $d$ , do the following.
  - a. Let  $Xu$  be the index of  $u$ . Initialize  $Xu$  to empty.
  - b. Let  $s$  be a sentence in  $u$ .
  - c. Remove stop words and numbers from  $s$ . Ignore  $s$  if it is reduced to one word or less.
  - d. For each sentence  $s$  in  $u$  do the following.
    - i. Generate topic candidates  $T$  from  $s$  (Section 3.2).
    - ii. For each topic  $t$  in  $T$ , do the following.
      1. Perform limited stemming on  $t$  (Algorithm 2).
      2. Perform topic canonization on  $t$ .
    - iii. Eliminate topics in  $T$  that are overlapping in position.
    - iv. Merge and sum the frequencies of topics  $T$  that are the same, similar or synonym, to produce index entry  $x(t, u, f)$  and add it into  $Xu$ . Notice that  $x(t, u, f1+f2)$  replaces both  $x(t, u, f1)$  and  $x(t, u, f2)$  in  $Xu$ .
  - e. Remove index entries  $x$  from  $Xu$  that do not satisfy any of the following conditions:
    - i. Topic  $t$  consists of significant words less than  $w$ .
    - ii. Topic  $t$  contains duplicate words.
    - iii. Topic  $t$  is a subset of other topics and  $t$  is not a stand-alone topic.
  - f. For each topic  $t$  in  $Xu$ , remove extraneous words from  $t$  (Algorithm 5). Remove  $t$  if it is reduced to one word or less.
2. For each document  $d$  in  $D$  do the following.
  - g. Let  $Xd$  be the index of  $D$ . Set  $Xd$  is the union of  $Xu$  from each  $u$  in  $d$ . In doing so, replace  $u$  with  $d$  in index entry  $x(t, u, f)$ .
  - h. Remove  $x$  from  $Xd$  if  $x(f) < r$ .
3. The index  $X$  is the union of  $Xd$  and  $Xu$  from each  $u$  in  $d$  and from each  $d$  in  $D$ .

### 3.2 Topic Generation

Given a sentence of length  $s$ , generate all possible phrases (topics) of length 2 to  $w$  words where words can be up to  $p$  positions apart from each other. The algorithm systematically generates all possible phrases as described in the following example.

#### 3.2.1 An Example

Let's generate all 3-word phrases of at most 3 positions apart from a text document "abcde...z". In this case, each letter represents a word. For a 3-word phrase, there are only 2 possible slots inside the phrase as shown in pattern  $XzXzX$ , where  $X$  represents one word and  $z$  represents a slot. For each slot  $z$ , we may skip 0, 1, or 2 words, i.e. at most 3 positions apart. The list of patterns is shown in **Table 1**. The dash signs in the patterns represent words that are skipped.

**Table 1.** List of patterns for generating topic candidates

#	Slots	Patterns	Phrases	#Phrases
1	0 0	XXX	abc, bcd, cde, ...	$24 = 26-3+1-(0+0)$
2	0 1	XX-X	abd, bce, cdf, ...	$23 = 26-3+1-(0+1)$
3	0 2	XX--X	abe, bcf, cdg, ...	$22 = 26-3+1-(0+2)$
4	1 0	X-XX	acd, bde, cef, ...	$23 = 26-3+1-(1+0)$
5	1 1	X-X-X	ace, bdf, ceg, ...	$22 = 26-3+1-(1+1)$
6	1 2	X-X--X	...	$21 = 26-3+1-(1+2)$
7	2 0	X--XX	...	$22 = 26-3+1-(2+0)$
8	2 1	X--X-X	...	$21 = 26-3+1-(2+1)$
9	2 2	X--X--X	...	$20 = 26-3+1-(2+2)$

The number of patterns is  $3^2 = 9$ . The number of phrases,  $24 + 23 + \dots + 20 = 198$ , is less than  $9 * 24 = 216$ , because there are 9 patterns each of which cannot generate more than 24 phrases (each phrase contains at least 3 words).

#### 3.2.2 Computational Complexity

The number of patterns consist of  $w$  words of at most  $p$  positions apart is  $p^{w-1}$ . An upper bound of the number of phrases consist of  $w$  words of at most  $p$  positions apart generated from one sentence of length  $s$  is  $(s - w + 1)p^{w-1}$ . Thus, the number of phrases  $f(s, w, p)$  is less than  $(s - w + 1)p^{w-1}$ . The number of phrases consisting of 2 to  $w$  words is  $g(s, w, p) = \sum_{i=2}^w f(s, i, p)$ .

#### 3.2.3 Computational Performance

##### *Worst Case*

**Table 2** shows the performance of iIndex on the worst-case scenario on generating all possible phrases from *one* sentence consist of unique words  $w_1, w_2, \dots, w_{124}$ . The value of  $s = 124$  is the longest sentence found among all text documents evaluated in this work. The value of  $w = 10$  is the default value set for iIndex.

The numbers in the table were computed by iIndex. The computer specified in Section 4 ran out of memory when the iIndex tried to compute  $g(124, 10, 3)$ . Therefore, the computation times for  $g(124, 10, 3)$  are estimated numbers as indicated by the stars on them.

**Table 2.** The performance of a worst-case scenario

$g(s, w, p)$	Patterns	Phrases	Minutes
$g(124, 10, 1)$	9	1071	0
$g(124, 10, 2)$	1,022	114,437	14
$g(124, 10, 3)$	29,523	3,158,934	*386
$g(124, 10, 4)$	349,524	35,767,926	*4,371
$g(124, 10, 5)$	2,441,405	238,647,305	*29,161

#### *Average Case*

Although the worst case scenarios are almost impossible to compute, the empirical results show that the average cases can be computed very efficiently (less than a second) as shown in **Table 3**. The table shows the performance of generating all possible phrases from one sentence consisting of 15 unique words. The value of  $s = 15$  and  $w = 3$  are based on the average sentence length and average topic length computed from all text documents evaluated in this work.

**Table 3.** The performance of an average-case scenario

$g(s, w, p)$	Patterns	Phrases	Milliseconds
$g(15, 3, 1)$	2	27	30
$g(15, 3, 2)$	6	75	40
$g(15, 3, 3)$	12	138	40
...	...	...	...
$g(15, 3, 12)$	111	555	90

#### *Best Case*

The best-case scenario is the condition when almost all of the cases are covered and it is still practical to compute. The length of sentences that cover 97% of sentences in all documents evaluated in this work is 43. The length of topic that cover 97% of the topics generated from all the documents is 6. Based on those values, the performance of the algorithm is computed as shown in **Table 4**. The empirical results show that we can efficiently compute  $g(43, 6, 3)$  in 7 seconds. That means it is practical to compute the index of text documents that contains sentences up to 43 words long, topics up to 6 words long, and topic proximities up to 3 positions apart.

**Table 4.** The performance of the best-case scenario

$g(s, w, p)$	Patterns	Phrases	Seconds
$g(43, 6, 1)$	5	200	0
$g(43, 6, 2)$	62	2,279	0

$g(43, 6, 3)$	363	12,327	7
$g(43, 6, 4)$	1,364	42,722	53
$g(43, 6, 5)$	3,905	112,250	156

With this approach, we efficiently solve the problem of finding important topics in texts, the problem which requires exponential computation time, by carefully selecting subsets of the problem that are practical to compute yet useful and cover 97% of the problem domain.

### 3.3 Similar Topic Detection

Topic  $t_1$  is *similar* to topic  $t_2$  if both of them have the same set of significant base words. *Significant* words are those that are not stop words. *Base* words are those that have been converted to their root forms by the process called limited stemming described below. Examples of similar topics are ‘repeated term’, ‘repeated terms’, ‘term repetition’, and ‘repetition of terms’.

*Limited stemming* is the process of converting word forms to their base forms (stems, roots) according to a set of conversion rules,  $F$ , as part of the simple grammar  $G$  described in Section 3.4. Only those words in  $F$  are converted to their base forms in addition to the automatic conversion of regular forms as described in the following algorithm.

Set  $F$  includes a list of irregular forms and their corresponding base forms as defined in the WordNet [Mi96] list of exceptions (adj.exc, adv.exc, noun.exc, verb.exc). Examples of irregular forms are ‘goes’, ‘went’, and ‘gone’ with base form ‘go’. The stemming is represented by one rule:  $go \rightarrow goes | went | gone$ .

Word forms that have the same sense in all phrases but are not included in the WordNet list of exceptions are manually added to  $F$ . Examples of such word forms are ‘repetition’ with base form ‘repeat’ and the word ‘significance’ with base ‘significant’.

#### Algorithm 2 Limited Stemming Algorithm

This algorithm returns the base form of a given word  $w$  or null.

1. If word  $w$  is defined in  $F$  then return its base form.
2. Else
  - a. If either suffix ‘s’, ‘ed’, or ‘ing’ exists at the end of word  $w$  then truncate the suffix from  $w$  to produce  $w'$ .
  - b. If length of  $w'$  is at least 2 then return  $w'$ .
  - c. Return null.

The limited stemming algorithm above has been developed to avoid some of the errors that arise when a generic stemming algorithm (such as described in [Sa89]) predicts that two words have the same meaning when they do not [Ha92, Fa87]. For example, the word ‘importance’ should not be stemmed to ‘import’ because the two words are semantically unrelated.

As mentioned above, stop words and word order are ignored when determining topics. When these ideas are combined with limited stemming, the following phrases are detected as similar: ‘repeated terms’, ‘repeated term’, ‘term repetition’, ‘repetition

of terms'. This heuristic will not always work. For example, it will never be able to distinguish between 'absence of evidence' and 'evidence of absence'. However, we have found very few cases of this sort.

### Algorithm 3 Similar Topic Detection

The following algorithm determines if topic  $t1$  is similar to topic  $t2$ .

1. Remove stop words from  $t1$  and  $t2$ .
2. Perform limited stemming on  $t1$  and  $t2$ .
3. Order words in  $t1$  alphabetically.
4. Order words in  $t2$  alphabetically.
5. Return true if  $t1$  is identical to  $t2$ .

## 3.4 Synonymous Topic Detection

Phrases that have the same meaning are called *phrase synonyms* or *topic synonyms*. In addition to topic canonization, phrase synonyms can be defined explicitly by adding production rules,  $S$ , to the simple grammar  $G$  defined below. For example, the following production rule specifies that phrases 'topic identification', 'determine topics', 'discover topics', and 'topic spotting' are synonyms:  $\text{topic identification} \rightarrow \text{determine topics} \mid \text{discover topics} \mid \text{topic spotting}$ .

The rules in  $S$  are manually constructed to improve the quality of the index. However, the iIndex produces good indices without defining any rules in  $S$ .

Phrase synonyms share one meaning called the *synonym meaning*, which is represented by the string at the head of the production rule. In the above example, the synonym meaning is string 'topic identification'. Each phrase (node) in the production rule represents a set of similar phrases.

Topic  $t1$  is *synonymous* to topic  $t2$  if and only if the synonym meaning of  $t1$  is literally the same as the synonym meaning of  $t2$ .

A *simple grammar*,  $G$ , is used to represent both stems for words and synonyms for topics. It is called a simple grammar because it can be implemented with a simple look up table with logarithmic complexity  $O(\log n)$  where  $n$  is the number of entries in the table which is the number of terms (nodes) in the production rules. The grammar could be implemented with constant complexity  $O(1)$  by employing good hashing technique.

There are 4519 rules defined in the current implementation of iIndex. The rules define 11452 mappings of one string to another.

### Algorithm 4 Synonymous Topic Detection

1. Remove stop words from  $t1$  and  $t2$ .
2. Convert topic  $t1$  and  $t2$  to their canonical phrases.
3. Let  $g1$  be the set of synonym rules contains  $t1$ . Let  $g2$  be the set of synonym rules that contains  $t2$ . Both  $g1$  and  $g2$  are subsets of the simple grammar  $G$ .
4. If intersection of  $g1$  and  $g2$  is not empty, then  $t1$  and  $t2$  are synonym otherwise they are not.

### 3.5 Topic Elimination

The iIndex generates some *incoherent* phrases such as ‘algorithm for determining’ and ‘automatic indexing involves’ during the indexing process. Those phrases need to be removed from the index.

Topics that contain duplicate words are also removed because they are mostly incoherent. An example of a phrase with duplicate words is ‘string the string’. The iIndex generates the phrase from [Ka96] because the phrase is repeated twice (ignoring stop words) as follows.

“... denotes the empty *string*, *the string* containing no elements ...”  
“... machine has accepted the *string* or that *the string* belongs ...”

#### 3.5.1 Remove Extra Words from Topics

This section describes heuristics to remove some incoherent phrases or to transform them into coherent phrases.

Define  $B$  as the set of words and phrases to be eliminated from the beginning of the topics and  $E$  from the end of the topics.  $S$  is the set of stop words. Set  $B$  and  $E$  are manually constructed and represent some sort of knowledge for the iIndex. Examples are  $B = \{\text{according to, based on, following, mentioned in}\}$  and  $E = \{\text{using, the following, involves, for combining, for determining, to make}\}$ .

##### Algorithm 5 Removal of Extraneous Words from Topics

1. Remove consecutive words or phrases from the beginning of topic  $t$  if they belong to  $B$  or  $S$ .
2. Remove consecutive words or phrases from the end of  $t$  if they belong to  $E$  or  $S$ .
3. If  $t$  is reduced to one word or less then do not use  $t$ , otherwise use  $t$ .

#### 3.5.2 Noun Phrase Filter

Noun phrases can be recognized using algorithms such as [WEK01]. However, limiting the index entries to just noun phrases is too restrictive. It will ignore the following important topics found by the iIndex: ‘high blood pressure has no symptoms’, ‘blood pressure should be monitored more frequently’.

Any indexer based on noun phrases suffers from the same problem. The iIndex, on the other hand, finds any important topics in text documents.

## 4 Implementation

The iIndex system has been written in C++. Computation is performed on a laptop, Pentium 4, 2 GHz, Microsoft Windows 2000 Professional, 768 MB memory, and 37 GB hard drive.

The inputs to iIndex are plain text documents in ASCII format. The limited stemming is defined in file forms.txt, topic synonyms in rules.txt, stop words in stopWords.txt, topic separators in topicSeparator.txt, and parameters such as  $s = 50$ ,  $w = 10$ ,  $p = 1$ ,  $r = 2$  are configurable in param.txt, where  $s$  is the maximum length of

sentences,  $w$  is the maximum length of topics,  $p$  is the proximity of topics, and  $r$  is the minimum phrase frequency to be qualified as a topic.

## 5 Result and Evaluation

The iIndex correctly and efficiently finds the most important topics in various types and lengths of text documents from sentences, paragraphs, short papers, extended papers, training manuals, to PhD dissertations. Titles and abstracts are not marked in any special way and thus are not known to iIndex. The topics extracted from texts are ordered by their importance (topic frequencies).

The iIndex finds 477 topics in [Wi98] training manual as shown in **Table 5** (N = sequence number, TF = topic frequency, WF = word frequency average). It correctly extracts topic ‘blood pressure measurement’ as the third most important topic, the topic mentioned in the title of the text. It is indeed true that the text is about blood pressure, high blood pressure, and blood pressure measurement as suggested by the first 3 most important topics.

**Table 5.** List of important topics in blood pressure measurement manual

N	TF	WF	Topics
1	250	306	blood pressure
2	56	227	high blood pressure
3	46	217	blood pressure measurement
4	19	38	american heart association
5	19	157	blood vessels

The iIndex finds 42 topics in [Ka96], a short paper. It correctly extracts the topic ‘finite state technology’ as the second most important topic, which is exactly the title of the paper. It is indeed true that the paper is about finite state, finite state technology, and regular language as suggested by the first 3 most important topics.

The iIndex finds 2172 topics in [Wo97], an extended paper. It correctly extracts the topic ‘conceptual indexing’ as the first most important topic, which is exactly the title of the paper. It is indeed true that the text is about conceptual indexing, conceptual taxonomy, and retrieval system as suggested by the first 3 most important topics.

The iIndex finds 2413 topics in [Li97], a PhD thesis. It correctly extracts phrase ‘topic identification’ as the second most important topic, the topic mentioned in the title. It is indeed true that the text is about topic signatures, topic identification, precision and recall as suggested by the first 3 most important topics.

### 5.1 Speed of Indexing

Overall, the iIndex is very effective and very efficient in finding the most important topics in text documents. It takes 34 seconds to index a 100-page text [Wo97]. It takes only 3 seconds to find 482 important topics among 23166 possible phrases in [Wi98] training manual and less than 1 second to find 43 important topics among 5017 possible phrases in [Ka96].

## 5.2 Comparisons to the Word Count Model

The word count model ranks the topics based on the word frequency average listed in column WF of **Table 6**. The word count model ranks topic ‘blood pressure cuff’ extremely high (2<sup>nd</sup>), a topic that is mentioned just 2 times in [Wi98] document, higher than topic ‘blood pressure measurement’, a topic that is mentioned 46 times in the document. It is incorrect to conclude that topic ‘blood pressure cuff’ is more important than topic ‘blood pressure measurement’ in the document. On the other hand, the iIndex correctly infers that topic ‘blood pressure measurement’ is much more important (3<sup>rd</sup>) than topic ‘blood pressure cuff’ (262<sup>nd</sup>) in the document as shown in **Table 5**. The iIndex order the importance of topics in the text document better than the word count model does.

**Table 6.** List of important topics in blood pressure measurement manual by word count average order

N	TF	WF	Topics
1	250	306	blood pressure
2	2	242	blood pressure cuff
3	8	229	blood pressure to clients
4	56	227	high blood pressure
5	17	221	elevated blood pressure

## 5.3 Comparisons to the TFIDF / Term Weighting Model

The best term weighting model is *tfidf* according to Salton [Sa89] who evaluated 287 different combinations of term-weighting models. However, *tfidf* fails to find the most important topic ‘voting power’ from the Wall Street Journal text [Li97, page 73] while the iIndex correctly finds it as shown in **Table 7**. The iIndex finds more specific and meaningful topics such as voting power, million shares, and eastern labor costs, while *tfidf* finds less specific topics such as Lorenzo, holder, voting, proposal, etc. The iIndex is better at identifying important and specific topics than *tfidf*.

**Table 7.** List of important topics in Wall Street Journal text identified by *tfidf* and iIndex

Rank	<i>tfidf</i>		iIndex	
	Term	Weight	Phrase	Frequency
1	Lorenzo	19.90	voting power	2
2	holder	9.66	million shares	2
3	voting	9.05	eastern labor costs	2
4	proposal	8.03		
5	50.7%	7.61		
...				
16	power	5.01		
...				

## 6 Conclusions

This paper presents, iIndex, an effective and efficient approach to indexing text documents based on topic identification. A topic is any meaningful set of words that is repeated at least twice in the texts. The determination of topics is based on the repetition of the words that appear together within texts. To measure topic frequencies in texts better, the iIndex detects topics that are similar or synonymous.

## References

- [ABC+00] A. R. Aronson, O. Bodenreider, H. F. Chang, S. M. Humphrey, J. G. Mork, S. J. Nelson, et al. The NLM indexing initiative. *Proc AMIA Symp 2000*(20 Suppl):17-21.
- [Fa87] J. L. Fagan. Automatic Phrase Indexing for Document Retrieval: An Examination of Syntactic and Non-Syntactic Methods. *Proceedings of the Tenth ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 91-108, June 1987.
- [Ha92] D. Harman. Ranking Algorithms. In William B. Frakes and Richardo Baeza-Yates, editors, *Information Retrieval Data Structures & Algorithms*, pages 363-392, Prentice Hall, New Jersey, 1992.
- [JCD+99] D. B. Johnson, W. W. Chu, J. D. Dionisio, R. K. Taira, H. Kangarlo. Creating and Indexing Teaching Files from Text Patient Reports. *Proc AMIA Symp 1999*:814-8.
- [Ka96] R. M. Kaplan. Finite State Technology. In Ronald A. Cole, Editor in Chief, *Survey of the State of the Art in Human Language Technology*, Chapter 11.6, Center for Spoken Language Understanding, Oregon Graduate Institute, USA, 1996.
- [KS97] F. Kellely, A. F. Smeaton. Automatic Phrase Recognition and Extraction from Text. *Proceedings of the 19<sup>th</sup> Annual BCS-IRSG Colloquium on IR Research*, Aberdeen, Scotland, April 1997.
- [Li97] C. Y. Lin. Robust Automated Topic Identification. *PhD Thesis*, University of Southern California, 1997.
- [MBSC97] M. Mitra, C. Buckley, A. Singhal, C. Cardie. An Analysis of Statistical and Syntactic Phrases. *Proceedings of RIAO97, Computer-Assisted Information Searching on the Internet*, pages 200-214, Montreal, Canada, June 1997.
- [Mi96] G. A. Miller. WordNet: A Lexical Database for English. *Communications of the ACM*, 38(11):39-41, 1996.
- [Sa89] G. Salton. *Automatic Text Processing*. Addison-Wesley, 1989.
- [SHMM98] C. Silverstein, M. Henzinger, H. Marais, M. Moricz. Analysis of a very large AltaVista query log. Tech. rep. 1998-014, Digital Systems Research Center, 1998.
- [WEK01] N. Wacholder, D. K. Evans, J. L. Klavans. Automatic Identification and Organization of Index Terms for Interactive Browsing. *Joint Conference on Digital Libraries 2001*:126-34.
- [Wi98] Blood Pressure Affiliate Faculty of the American Heart Association of Wisconsin. Blood Pressure Measurement Education Program Manual. American Heart Association of Wisconsin, Milwaukee, 1998.
- [Wo97] A. W. Woods. Conceptual Indexing: A Better Way to Organize Knowledge. Technical Report SMLI TR 97-61, Sun Microsystems Laboratories, Mountain View, CA, 1997.