International Journal of Electrical and Computer Engineering (IJECE)
Vol. 5, No. 3, June 2015, pp. 409~420
ISSN: 2088-8708

Text Preprocessing using Annotated Suffix Tree with Matching Keyphrase

Ionia Veritawati*, Ito Wasito, T. Basaruddin****
* Department of Informatics, Pancasila University, Indonesia
** Department of Computer Science, University of Indonesia, Indonesia

Article Info

Article history:

Received Dec 18, 2014
Revised Apr 29, 2015
Accepted May 14, 2015

Keyword:

2-Means Clustering
Annotated Suffix Tree
Keyphrase
Preprocessing
TF-IDF

ABSTRACT

Text document is an important source of information and knowledge. Most of the knowledge needed in various domains for different purposes is in form of implicit content. A content of text is represented by keyphrases, which consists of one or more meaningful words. Keyphrases can be extracted from text through several steps of processing, including text preprocessing. Annotated Suffix Tree (AST) built from the documents collection itself is used to extract the keyphrase, after basic text preprocessing that includes removing stop words and stemming are applied. Combination of four variations of preprocessing is used. Two words (bi-words) and three-words of phrases extracted are used as a list of keyphrases candidate which can help user who needs keyphrase information to understand content of documents. The candidate of keyphrase can be processed further by learning process to determine keyphrase or non keyphrase for the text domain with manual validation. Experiments using simulation corpus in which keyphrases are determined from them show that keyphrases of two and three words can be extracted more than 90%. Using real corpus of economy, keyphrases or meaningful phrases can be extracted about 70%. The proposed method can be an effective way to find candidate keyphrases from collection of text documents which can reduce non keyphrases or non meaningful phrases from list of keyphrase candidates and can detect keyphrases separated by stopwords.

Corresponding Author:

Ionia Veritawati,
Department of Informatics,
Pancasila University,
Srengseng Sawah Street, Jagakarsa, Jakarta, Indonesia
Email: ioniaver11@gmail.com

1. INTRODUCTION

Text used as data has increased rapidly in many domain areas. It becomes a problem when a person or a department needs the content of text or document collection as information for their purposes. A big text data causes difficulty in knowing the content of the text or document collection. The collections (corpus) have implicit information which can be extracted to give meaningful information. Accurate information requires processing of the text as an unstructured data and physically as documents.

Keyphrase (KP) is a meaningful phrase consisting of one or more words which can be extracted from a document collection using some different methods. KEA, called Automatic Keyphrase Extraction, is a method for extracting keyphrases using Naive Bayes [1]. Some other methods use Semantic Analysis [2] lexical chains [3], and a ranking approach by SVM [4]. Keyphrasescan also be extracted using thesaurus database as compound terms [5] and extracted using entropy and transition point approach as index term [6].

Usually, keyphrases are features of text collection in which their numbers are calculated based on their presence for values in the features. Keyphrases have been used in many applications such as for determining index of a digital library [7]; for supportinga text-based decision system in financial sequence prediction [8] andquestion answering system [9]; for ranking topical keyphrases from content-representative document titles [10]; for clustering document [11] and text from structured data [12]; for applying in query-oriented summarization [13], information extraction [14] [15], text categorization [16] and information retrieval [17].

Text Preprocessing as an initial process has been used in many experiments such as for exploring the impact of preprocessing on text classification [18]; for compressing natural language [19]; and for selectingfeature [20]. Uysal [18] has compared a few steps of preprocessing.In this paper, text preprocessing is combined by Annotatted Suffix Tree (AST), which consists of collection of two words and frequencies from a document. The AST in this paper is applied to match and score a list of inputted keyphrases and also to extract keyphrases automatically if there are no input keyphrases to be matched.

This paper consists of four sections. The second section describes text preprocessingand methods.It includesconcept of Annotated Suffix Tree (AST), algorithm of AST building, algorithm of AST matching andalgorithm of automatic keyphrase extraction that combines text preprocessing and AST. This section also explains the methodology of experiments. The third section presents results and analysis of results from the experiments. The final sectioncontainsconclusions and future work.

2. PREPROCESSING AND METHODS

2.1. Text Preprocessing

Text Preprocessing is a systematic and basic process applied in a collection of text documents related to removalof meaningless characters, unimportant words and elimination suffix or prefix from a word [18]. The result of text preprocessing is a list of meaningful words which can represent the content of a document or a collection of documents. The result list will be used in variousapplications, which are described in previous section.

Meaningless characters in text or document are comma, point, question tag and others. Removal of the characters and also the change the capital letters into small characters make the next step of preprocessing easier and is followed bythe removal unimportant words (stop words) such as conjunction and adverb.The list of all the words is collected first and then the unimportant words found are removed from the document collection. The next process continues by using a stemmer which is applied to eliminate prefix or suffix from a word that may form verb or noun. The stemmed words can be used as features of documents. Frequency, as a score of each feature, is arranged in an element of score matrix between documents (columns) and keyphrases (rows). The score matrix is normalized using TF-IDF calculation. The normalized keyphrase scores in the matrix or table are important to find content domain of the document collection and also helpful when they are used for querying or clustering documents.

2.2. Setting of Proposed Text Preprocessing

Text preprocessing in this paper has four settings shown in Table 1. The purpose of using different settings are to find the best result of keyphrase matching process which will be described in section 2.4.

Table 1. Setting of Proposed Text Preprocessing

Setting	Remove Stop Words (i)	Stem (j)
(1)	(2)	(3)
1	No (0)	No (0)
2	No (0)	Yes (1)
3	Yes (1)	No (0)
4	Yes (1)	Yes (1)

Each of the four settings (table 1) is applied to an original corpus. Preprocessing results of the four settings are four collections of documents (corpus) appropriate with each setting. The settings can be expressed as$x_{i,j}$; where x is a word, i is index of removal of stop words (table 1, column 2), and j is index of stem (table 1, column 3), which have different states in each setting. Index values of $x_{i,j}$ include"not removing stopwords" (i = 0) or "removing" them (i=1), and "not using stemmer" (j=0) or "using" it (j=1). For example, $x_{0,1}$means the word x will not be removed if it is a stop word and the word x will be stemmed.

To find a keyphrase of two or three words from a corpus, usually the process is sequential, using an array structure to arrange the words (2-gram or 3-gram from N-gram method), besides using methods described in previous section. In this paper, a different method for matching and extracting keyphrases of single word until three words using Annotated Suffix Tree (AST) is proposed. Each of four preprocessed corpus from setting model data (table 1) is used to build four various tree structures as AST and each of AST is used in the processes of keyphrase matching and automatic keyphrase extraction. The specific methods will be explained in the next section.

2.3. Annotated Suffix Tree

Suffix tree in general is a tree that consists of characters as suffix of a string (collection of characters) [21]. The root node and sub-trees of suffix tree do not have value. Value is put in each vertex of sub-tree. For example, the word "bananas" consist of suffixes as follows: "bananas", "ananas", "nanas", "anas", "nas", "as", "s" and "$" as an end character. Each suffix is arranged in vertex of the suffix tree. It is usually used for string matching. Ukkonen's online algorithm [21] is proposed to build the suffix tree and the step using prefix rather than suffix. For example, prefixes of "bananas" are as follows: "b", "ba", "ban", "bana", "banan", "banana", "bananas".

Annotated Suffix Tree (AST) is a different concept of suffix tree proposed by Pampapathi [22] for spam filtering. Pampapathi developed AST as a structure of a word that consists of characters and its frequencies which are put at nodes of treerather than string. AST is also used to match string patternsofspam words. Each of suffix tree models above is used for a single word.

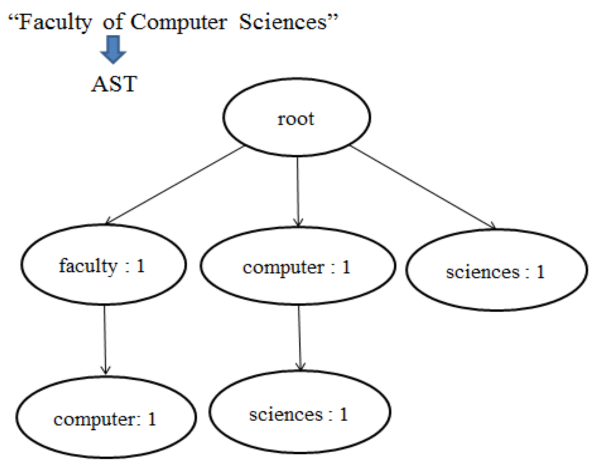

Figure 1. The Proposed AST Illustration

In this paper, the proposed Anottated Suffix Tree (AST) algorithm is adopted from Ukkonen concept in developing online suffix tree and Pampapathi [22] in arranging nodes of tree as the place for putting characters. Figure 1 shows the AST illustration of bi-words. The AST proposed is arranged to put a bi-words, not a character, and its frequency in a node. The depth of the tree has two levels. The number of all nodes in level 1 of the tree is the total number of unique words in the document developed as AST, and the number of frequency at each node of that level is the total number of words.

```
Read a document D(i) has been Preprocessed
Split text of D(i) into Array of words (1..number of words)
Create root of tree
 For j= 1 .. (number of words -1)
{Insert word(j), word (j+1)  into tree}
    Traverse level 1,
    If word(j) = node1 → add frequency  of node1 {level 1}
        If  node2 (subtree) of node1 = word(j+1)
        add frequency of node2 {level 2}
        Else Insert word(j+1) as a new node2 (subtree)  of node1 at level 2
    Else
        Insert word(j) as a new node1 (subtree) of root at level 1
        Traverse level 1,
        if there is no node1 == word(j+1)
            Insert word(j+1) as a new node1 (subtree) of root at level 1
```

Figure 2. AST Development Algorithm

The proposed algorithm to develop AST is described in figure 2. The main process puts every two words of a text document into AST at level one and two. If the first word that is going to be inserted has existed at a node of level one of the tree, the node will split into a new sub-tree to insert the second word and its frequency.

2.4. Matching Process

Matching process algorithm (figure 3) in this paper is the process of scoring a list of inputted keyphrases into a document collection using AST developed from each document.The process of matching and extracting keyphrases can be done by traversing AST and matching every word in nodes with the inputted keyphrases as word or bi-words one by one, and count the frequency of occurrences of respective matching words.

```
Input a Keyphrase (KP) will be matched with AST
Score=0
Split KP into words (1.. number of KP)
For j= 1 .. (Nword-1)
{matching  and scoring word(j), word (j+1)  with AST}
   Traverse level 1,
   If word(j) = node1  {level 1}
      Traverse level 2
      If  node2 (subtree) of node1 = word(j+1) {level 2}  →
      score=score + frequency of node2
   score_matching = score / (number of KP -1)
   Insert score_matching  as an element table of document vs KP
```

Figure 3. Keyphrase Matching Algorithm using AST

Formulation for keyphrase (KP) matching score in the process of algorithm (figure 3) is as follows:

Score of KP=
(weighted KP * frequency in node level 2/(number of KP – 1)   (1)

In this experiment, the weights of word are equal to 1.

```
Preprocessing documents collection $D_i$ [i=1.. number of documents]
      (option : remove stop words + stem)
Preprocessing Keyphrases List - KP [1.. number of KP]
      (option : stem)
For i = 1 ..number of documents
      Develop AST(i) of  D(i)
      For j = 1 ..number of KP
            Score = Matching  between AST(i), KP(j) (figure 3)
            Insert  score into table $T_1$(i,j) {single word}
            Insert  score into table $T_2$(i,j) {bi-words / three-words}
Score table normalization using TF-IDF (single word):
        Score = tf * log (N/n)
                              // tf : frequency
// N : number of documents
                              // n : number of documents, which KP is presence
Score table normalization using TF-IDF modification (bi-words / three-words) :
        Score = tf * (log(N) - log (N/n)]
```

Figure 4. Table Score Arranging Algorithm

The process of matching keyphrases is applied to each wordof four types of corpus resulted from text preprocessing referred to setting in table 1 in which each document in every setting is developed to four different AST. Therefore, each inputted keyphrase will be matched to each of those different AST. There are two types of inputted keyphrases: list of stemmed keyphrases which will be matched with stemmed corpus, and non-stem keyphrases which will be matched with non-stem corpus.Normalized score of single word which is processed using standard TF-IDF (figure 4) will be small for a high frequency of the word.

Meanwhile, normalized score of bi-words and three-words which are processed using modified TF-IDF (figure 4) will alsohigh for a high frequency of bi-words and three-words.

2.5. Automatic Keyphrase Extraction

At the previous section, AST is applied to match and score an inputted keyphrase, and then the score is arranged into feature table between documents and keyphrases. At this section, AST is used to find keyphrases from a collection of documents automatically.

The proposed method consists of four steps (figure 5). The initial step is extracting all keyphrases from AST, according to minimum threshold of frequencies at the nodes. The extraction process uses four types of corpus from preprocessing, which is coded by combination of “remove stop word” and “stem”and which is described in section 2.2 (table1). The codes are 00, 10, 01 and 11. For example, code “10” refers to extracted keyphrases from corpus with which applies “remove stop word” and does not apply “stem”. From the four types of corpus, the results are four types of keyphrases lists (KP00, KP10, KP01 and KP11) and also four types of tables of document versus keyphrases (TF-IDF00, TF-IDF10, TF-IDF01 and TF-IDF11).

```
 Extract KP from preprocessing : 00 - 10 - 01 - 11 {List 0}
 STEP I (Search KP)  : compare : KP00 - KP10
      if KP match :
           match with KP 01 + KP11 (without stem)
 STEP II (Search Suffixes word) : compare KP result from step I
      match with KP 01 + KP11 (with stem)
 STEP III (Search “word + stopwords”) : compare KP 00-01
match and remove stopwords
 STEP IV : eliminate overlap words (1-3 words) of KP result from step I - III
```

Figure 5. Automatic Keyphrase Extraction Algorithm

The process of algorithm in Step I (figure 5) compares keyphrases between the lists of keyphrases to find the same keyphrases. The next step, step II and III (figure 5) searches keyphrases from word withaffixes and from phrases of three-words that have a stop word in the middle of the phrases. The list of keyphrases resulted from step I – III is checked at the last step (step IV). The process is to eliminate keyphrases that overlap each other. For example, it eliminates bi-words keyphrases which are included in three-words keyphrases, single words which are included in bi-words keyphrases, and between keyphrases which are the same.

2.6. Methodology

The methodology of the experiments is shown in figure 6.Experiments using AST are applied into two parts. The first part is a process to match inputted keyphrases with AST of each document in a corpus to get score matching, refer to algorithm in figure 4. The second part refer to algorithm in figure 5 is a process to extract keyphrases automatically from a corpus. Each experiment is evaluated separately.

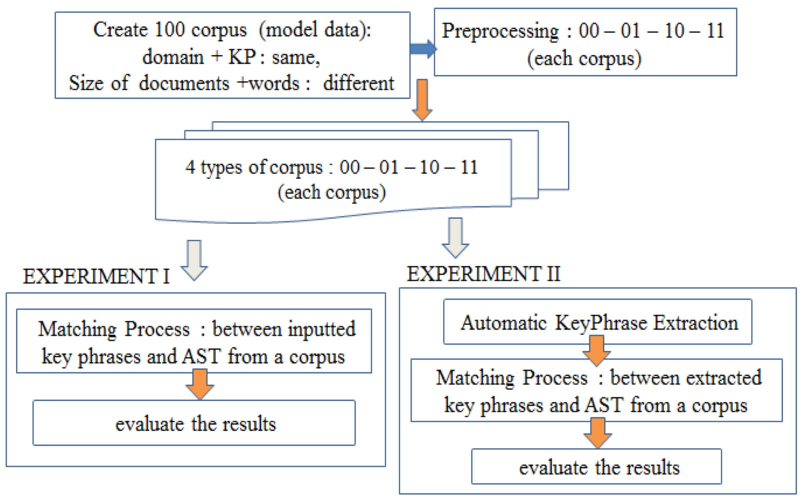
Create 100 corpus (model data): domain + KP : same, Size of documents +words : different
Preprocessing : 00 – 01 – 10 – 11 (each corpus)
4 types of corpus : 00 – 01 – 10 – 11 (each corpus)
EXPERIMENT I
Matching Process : between inputted key phrases and AST from a corpus
evaluate the results
EXPERIMENT II
Automatic KeyPhrase Extraction
Matching Process : between extracted key phrases and AST from a corpus
evaluate the results

Figure 6. Methodology of Experiment

2.7. Data for Simulation

For easier evaluation, data used in the experiments are simulation data generated from a corpus generator (figure 7). The number of keyphrases, words with affixes and the list of stop words for three domains are determined first. In this model, if stemmer is applied, words with affixesthat can create a keyphrase with a higher score. All the words combined with list of stop words become a list of combined words. Each index of combined words is randomized and then the randomized words become a simulation document. After repeating the process, several created documents are joined in a formatted corpus and ready to be used as simulation (model) data for further process. The model data corpus generated and real data documents from economy domain are used for simulation and investigationof performance of the proposed method.

Figure 8 and figure 9 are views of sample of keyphrases and words with affixes in Indonesian language. The phrases consist of single until three-words, and the stop words have been removed but the phrases are not stemmed (code:10, refer to setting of table 1). Stemmer used in these experiments is stemmer for Indonesian language.

- Input :
  - List stop words
  - files : list Keyphrase + words with affixes
- Process (each – document):
  - random -- stop words + Keyphrase + words with affixes
  - tag Formatting
  - Save document in a corpus file
- Output :
  - A corpus file

```
G= General
Domain = A; {A | A part of G}
Document= D; {D | D part of A}
Word =w; {w | w part of D}
Keyphrase=p;
Word with affix = x; {p, x | p,x part of D}
Stop word = s; {s | s part of G}
{p,x,s | p,x,s part of w}
N1= number of documents (D)
N2 = number of a keyphrase (p) at Di ; i: index of a document
N3 = number of a word with affix (x) at $D_i$
N4 = number of a stop word at $D_i$
$nKP_i$= number of different keyphrase
$nX_i$= number of different keyphrase
$nS_i$= number of different keyphrase
w ← x, p, s
R= (nKPi*N2) + (nXi*N3) + (nSi*N4)
r = randomize (1..R)
for r= 1 .. R, save w(r) to Di
create a formatted corpus of N1 documents
```

Figure 7. Automatic Corpus Generation Algorithm

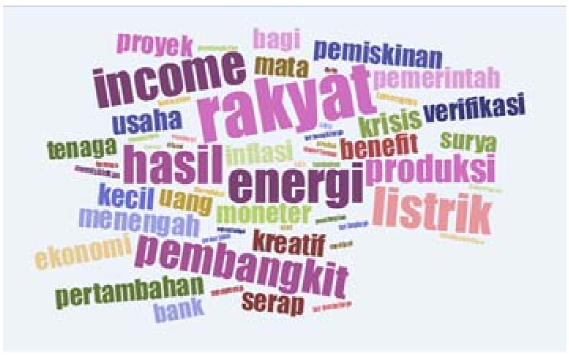

Figure 8. Ilustration: List of Words as a Single Word in Experiment

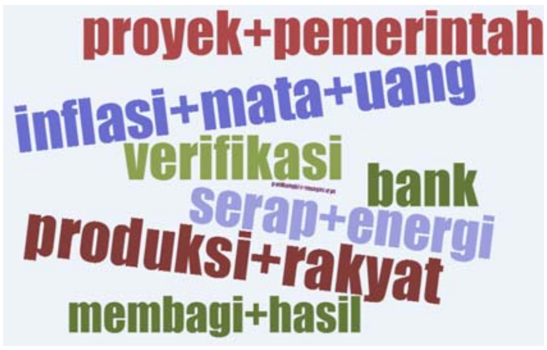

Figure 9. Ilustration: List of Words as Bi-Words and Three-Words in Experiment

3. RESULTS AND DISCUSSION

3.1. Experiment of Keyphrase Matching

ExperimentI (figure 6) is a process for keyphrase matching. The list of keyphrases model and inputted keyphrase using for matching process is determined first fromthree subdomains of economy (figure 8 and 9). A combination of100-corpus is generated and arranged in fivesettingsof model data (table 2). Each model data consists of 20 corpus whichare generated by algorithm for Automatic Corpus Generation (figure 7). Each corpus consists of 15–900 documents. Each document consists of 20 keyphrases model where the numbers are 2-40 words or phrases for each keyphrase, 24 words with affixes where the numbers are 2-40 for each word. Eachdocument also consists of a list of stop words with 4-60 words for each stop word. The list of inputted keyphrase has the same contentas the list of the keyphrase model.

Table 2. Setting of Model Data (Simulation Data)

No.	Model Data Setting
1	(number of KP) = (number of wordswithaffixes)
2	(number of KP) > (number of wordswith affixes)
3	(numberof KP) < (number of wordswithaffixes)
4	(numberof KP) >> (numberof wordswithaffixes)
5	(number of KP) << (number of wordswithaffixes)

The results of experiment I, which matchingprocess between inputted keyphrases (KP) with AST developed fromeach document of a corpus, shown in table 3 (column 4). It showsas averages and standard deviations of percentage of total matching score from 100 corpus processed. Each score is related to combination of four types of text preprocessing (column 5, table 3), which has been explained at section 2.2 (table 1). Number of keyphrases model (Score0) determined first in corpus generator (figure 7) and is also used for comparison in column 1-3 of table 3.

Table 3. Average and Deviation Standard – Results of Matching of List of Keyphrases using Model Data (Matching Score Comparison)

No.	Number of Matching Keyphrase /Total Keyphrase Model (% of Average ±Deviation Standard)				Preprocessing
	= Score0	> Score0	< Score0	Total Score of KP	StopWords+ Stem
	(1)	(2)	(3)	(4)	(5)
1	48.42±9.39	26.50±5.16	8.57±7.41	83.49±2.64	00
2	50.38±13.73	32.38±12.36	15.44±13.83	98.20±2.83	10
3	32.54±9.49	41.81±5.35	8.57±7.41	82.92±2.59	01
4	33.00±13.73	47.69±12.56	15.30±13.77	96.94±3.12	11

The scoresof (extracted keyphrases = number of keyphrases model (=Score0, column 1 of table 3) will result in48% or higher, if the text is preprocessed without stemmer, without or with “remove stop words” (00 and 10, column 5 of table 3). In the same experiment, the scores of (extracted keyphrases > number of keyphrases model (>Score0, column 2 of table 3) will result in 41% or higher, if the preprocessed text apply stemmer, without or with “remove stop words” (01 and 11). Other result, the scores of (extracted keyphrases <number of keyphrases model (<Score0, column 3 of table 3)will result in 15.3 % or higher, if the text is preprocessed with “remove stop words” and without or with stemmer (10 and 11).

Total score of matching process of experiment I (column 4 of table 3) will result inbetter score about 96%, if the process uses corpus which has been preprocessed by “remove stop words” and by “stem” or “non-stem” process (code 10 and 11, column 5 of table 3) compared to result in about 82%, from corpus which is preprocessed without “removing stop words” and by “stem” or “non-stem” process (code 00 and 01).Based on every score value, it can be seen that thescore processed by ”removing stopwords” will make the total score of keyphrase matching higher compare toscore processed by stemming words. Generally, process of keyphrase matching will give a better score if the inputted keyphrases are matched with AST from a corpus with“removing stopwords” and “non-stemmed words” (code 10) or a corpus with“removing stopwords” and “stemmed words” (code 11).

“Removing stop words” has the most contribution in finding key phrases from text based on inputted key phrases model as references for key phrase extraction. Meanwhile,applying“stemmed words” could reduce a small number of keyphrases extracted because of stemmer which sometimes diminish meaning words of domain representation.

3.2. Experiment of Automatic Keyphrase Extraction using Model Data

Experiment II (figure 6) is a process for automatic keyphrase extraction.Each settingof model datain table 4 and table 5 (column 1) which consists of 8 corpus has the same model data as the ones in experiment I (referring tomodel data settingintable 2). The results of experiment II are shown in column 2, 3, 4, 5 of table 4 and column 2, 3 of table 5.

Table 4shows the comparison between the number of extractedkeyphrases and the number of keyphrases model. The experiments use stemmed corpus and non-stemmed corpus for extracting 1-3 words as keyphrases. The resultsof automatic keyphrases extractions are more than 90% (column 2-5, table 4). It means almost all keyphrases model with stem or without stem for 1-3 words or 2-3 words from corpus are extracted automatically, using settings of model data (column 1, table 4). Extracting keyphrases using non stemmed words give the same or higher percentage of extraction.

Table 4. Average and Deviation Standard – Results of Automatic Extraction ofKeyphrases using Model Data of Five SettingsComparedto Number of Keyphrases Model

No	Model Data Setting	Number of Match Extracted KP /Number of KP Model (% of Average ±Deviation Standard)			
		1-3 Words		2-3 Words	
		NonStem	Stem	NonStem	Stem
	(1)	(2)	(3)	(4)	(5)
1	(no. of KP) = (no. of words&affixes)	100.00± 0.00	100.00± 0.00	100.00± 0.00	100.00± 0.00
2	(no. of KP) > (no. of words& affixes)	93.75±11.57	93.75±11.57	92.65±13.62	92.65±13.62
3	(no. of KP) < (no. of words&affixes)	95.00±8.86	93.75±10.94	97.79±4.38	96.32±6.99
4	(no. of KP)>>(no.of words&affixes)	94.38±10.50	94.38±10.50	93.38±12.35	93.38±12.35
5	(no. of KP)<<(no. of words&affixes)	97.50±3.78	93.75±7.91	99.26±2.08	94.85±6.62

According to five model data settings (column 1, table 4), model with number of keyphrases lower than number of words with affixes (row 3 and 5, table 4). They have better results compared to model with number of keyphrases greater than number of words with affixes (row2 and 4, table 4).It is because keyphrases from setting in row 3 and 5 are separated wellfrom words with affixesso that the keyphrases can be matched more easily. Especially in the case of model data setting at row 1, where there is equality between the number of keyphrases and words with affixes, the extracted results can reach 100%. This setting model shows that the method may extract all keyphrases for a specific case.

Table 5. Average and Deviation Standard – Results of Automatic Extraction of Keyphrases using Model Data of Five Settings Comparedto Number of All Phrases Extracted

No	Model DataSetting	Number of MatchingExtracted KP /Number of Extracted KP (% of Average ±Deviation Standard)	
		1-3 Stem Words	2-3 Stem Words
	(1)	(2)	(3)
1	(no. of KP) = (no. of words&affixes)	86.50±1.28	84.49± 1.43
2	(no. of KP) > (no. of words& affixes)	77.47±10.74	80.24±16.44
3	(no. of KP) < (no. of words&affixes)	83.01±5.83	82.76±5.37
4	(no. of KP)>>(no. of words&affixes)	93.38±13.60	93.38±16.44
5	(no. of KP)<<(no. of words&affixes)	81.79±8.57	80.03±9.11

Table 5 shows the number of stemmed and extracted keyphrases model compared to number of all phrases extracted. Actually, phrases extracted are not always keyphrases. Phrasesor keyphrases to be extracted in this experiment arederived from the stemmed corpus. The results of the comparison for 1-3 words of matchingextracted keyphrases are about 77- 93% (column 2, table 5) and the results of the comparison for 2-3 words are about 80-93% (column 3, table 5). It means, there are about 10-13% extracted phrases which are non-keyphrases.

According to five model data settings (column 1, table 5), model with number of keyphrases equal or lower than number of words with affixes (row 1, 3 and 5, table 5) give better results than. Using model with number of keyphrases greater than number of words with affixes (row 2, table 5). It is because it has the same reason with cases in table 4 in which keyphrases are separated well from words with affixes.As a result, the keyphrases can be matched more easily. Especially for model data setting at row 4, in which the number of keyphrases very greater than the number of words with affixes, the extracted resultsmay reach 93%, due to a great number of keyphrases. Consequently, almost all keyphrases are extracted.

Results by model data (table 4) for automatic keyphrase extraction show that almost all key phrases model (more than 90%) can be extracted using Annotated Suffix Tree (AST) combined with text preprocessing. More realistic results is shown in table 5, in which all phrases extracted using the proposed method result in more than 77% keyphrases. It means that the proposed method is good enough to be used for extractingkeyphrases or phrases which represent domain of text. Compared to the other method, such as KEA,a method for extracting keyphrases using Naive Bayes [1] which needstraining datafor developing a model prediction of keyphrases, the proposed method extracts keyphrases automatically without training data.

3.3. Experiment of Automatic Keyphrase Extraction using Modified Model Data and Real Data

This experiment is still related to experiment II in which the datain table 6 is a modification of model data in experiment I by adding a list of non-keyphrases which frequencies are lower than matching threshold. To generate modified model data, addition algorithm (figure 10) is combined to corpus generator (figure 7). The purpose of the additional data to the model is to make a closer condition to real text data. Results of this experiment are focused on extraction of 2-3 words, because it determines whether a phrase has meaning or not. Results of automatic keyphrase extractionin column 2 of table 6 use keyphrases model to be compared.

Setting of modified model data (column 1 of table 6) consists of two model data corpus which use the same type of domain, keyphrases and words with affixes(some contain keyphrases). In model data I, the number of keyphrases is greater thanthe number ofwords with affixes. The number of each non keyphrase is 50 and the number of documents is 120. The number of keyphrases, the number of words with affixes, the number of stop words andthe number of non keyphrases for each document are respectively 5, 3, 12 and 1. In model data II, the number of keyphrasesis smaller thanthe number of words with affixes. The number of other parameters is the same asthe one in model data I. The two models consist of the same list of non-keyphrases.

```
Combine this code with algorithm infigure 7 :
non-keyphrase = y;  {y | y part of G}
N5 = number of non keyphrase at Di
nYi= number of different non-keyphrase
w ← x, p, s, y
R= (nKPi*N2) + (nXi*N3) + (nSi*N4) + (nYi*N5)
r = randomize (1..R)
for r= 1 .. R, save w(r) to Di
create a formatted corpus of N1 documents
```

Figure 10. Part of Modified Corpus Generator by Adding Non Keyphrases

Table 6. Results of Automatic Extraction of Keyphrases using Model Data by Addition of Non Keyphrases (2-3 Keyphrases)

Modified Model Data	Number of Extracted KPModel / Number of KP Model (%)	Number of MeaningfulPhrase/ Number of Extracted Phrase (%)	SeparatedKP Candidates of Main Domain (2-Means)	Table TF-IDF from Prepro-cessing
(1)	(2)	(3)	(4)	(5)
I	94.12	44.44	>90%	00, 01, 11
II	86.00	36.00	>90%	00, 01, 10

The comparisons of number in Table 6(column 2 and 3) focus only on 2-3 keyphrases. The resultsof extraction in column 2 of table 6 show that more than 80% keyphrases model can be extracted. By validating the score results in column 3 of table 6 manually, about 36-44% phrasesextracted are meaningful phrases and, the resultsof all meaningful phrases extracted are keyphrases. The results in column 2 and 3 of table6show that model data I give better extraction resultsthan the results in model II. It means that the process of automatic keyphrase extraction using AST in this experiment will have a better output of keyphrases for data in which distribution of keyphrases is more significant than the number of other phrases (case study of model data I and II).

The list of extracted phrases and their matching score as a results from automatic keyphrase extraction isarranged by Table Score Arranging Algorithm (figure 4) into four types of TF-IDF tables (TF-IDF00, TF-IDF01, TF-IDF10, TF-IDF11) from corpus in model data I and II. Each of four types of preprocessing (table 1) is applied in the scoring process. The table of TF-IDF (documents versus extracted phrases) becomes an input of 2-means clustering as an unsupervised learning process.The objective of this clustering is to separate keyphrases candidate from extracted phrases (main domain) in a corpus. By checking manually,one cluster containing dominant meaningful phrases from the results of clustering, more than 90% of the keyphrase candidates or meaningful phrases(Column 4 of table 6) can be separated. The tables of TF-IDF that provide a significant separation of keyphrase candidatesare shown in column (5) of table 6.Input tablesinput which give good clustering results are TF-IDF00 and TF-IDF01. Generally, automatic keyphrase extraction using modified model data combined by 2-means clustering give performance score up to 90% to separate clusters consisting of extract keyphrase candidates from a documents collection.

The results of experiment II using text data from 3 corpus of real documents collection mainly in economy domain and extraction of 2-3 words are presented in table 7. The matching evaluation is done manually because comparison data for real data document are not provided, due tothe non existence of keyphrases model. Based on manual check of all meaningful phrases extracted, the results (column 2, table 7) show that by using the real data, more than 70% of extracted phrases compared to all phrases are meaningfulphrases, and the meaningful words / phrases are not always keyphrases. For example, “kenaikan harga” (increasingof price) is meaningful words / phrase but it is not a keyphrase.

Table 7. Result of Automatic Extraction of Keyphrases using Real Data

Real Data (no. of doc)	MeaningfulPhrases/ All Phrases (%)	SeparatedKP Candidates of Main Domain(2-Means)
(1)	(2)	(3)
10	76.92	>90%
29	71.43	>90%
240	72.00	>90%

The same step of clustering process to this real data is applied to modified model data for separatingmeaningfulphrases or keyphrase candidates from all phrases extracted. The results of 2-means clusteringchecked manually to a cluster containing dominant meaningful phrases are about 90% phrases (column 3 of table 7)having meaningfulones.The results score of clustering to real data are almost the same asresults score of experiment using modified model data (column 4 of table 6). Generally, AST can be used to extractkeyphrasesautomatically and combined by clustering method to get meaningful phrases. It will give a better result if the inputted data for clustering is table of TF-IDF00 or TF-IDF01, which “removing stopwords” should not be applied and “stemmed words” can be applied as an option.

4. CONCLUSION

Keyphrase matching, scoring and extraction using AST technique and combination of text preprocessing and followed by clustering propose a method to extract keyphrases from text, which generally extracts meaningful phrases. In the initial process, it runs automatically and it does not need domain experts. Manual checking is done at the end of experiment to give a list of extracted keyphrase candidates. It can reduce efforts of experts because they determine keyphrase candidates that have been extracted at the end of experiments.

Specifically, memory used in AST structure is lower, and text preprocessing to determine keyphrases is more efficient than conventional process using 2-gram or 3-gram method.In general, the proposed method can be used as a semi-automatic keyphrase extraction from documents collection which is not a text of a specific domain corpus.As a result, the method can be generally used in other domains, because it can detect keyphrases without checking the meaning, when it runs automatically.

Comparison to other methodssuch as N-gram and extract key phrases using Naive Bayes is a relative comparison, which the differenceslie in data structure and rule. Althoughit shows relative comparison, the proposed method shows the ability to extract keyphrases according to the experiments.

Future work of this experiment will use the candidate of keyphrases to be clustered or classified with more precisely, including keyphrase or non-keyphrase categories related to the domain investigated. The results can be used to develop a knowledge based on a domain and also use them as a candidate of query in Information Retrieval System.

ACKNOWLEDGEMENTS

We wish to acknowledge Prof. Boris Mirkin for his contributions to this research.This research was supported partially by Grant from Directorate of Higher Education of Indonesia.

REFERENCES

[1] I.H. Witten, *et.al.*, "KEA : Practical Automatic Keyphrase Extraction", *http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.55.3127*, Hamilton, New Zealand, 1999.
[2] M. H.Haggag, "Keyword Extraction using Semantic Analysis", *International Journal of Computer Applications*, vol. 61, pp. 1–6, 2013.
[3] G. Ercanand I. Cicekli, "Using Lexical Chains for Keyword Extraction", *Information Processing & Management*, vol. 43, pp. 1705–1714, 2007.
[4] X. Jiang, *et al.*, "A Ranking Approach to Keyphrase Extraction", *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval - SIGIR '09*, 2009, pp. 756-757.
[5] Y. Abuzir and T. Sabbah, "First Token Algorithm for Searching Compound Terms Using Thesaurus Database", *Journal of Computer Science*, vol. 8, pp. 61–67, 2012.
[6] H.A. Rahman and S.A. Noah, "A Comparative Analysis of the Entropy and Transition Point Approach in Representing Index Terms of Literary Text", *Journal of Computer Science*, vol. 7, pp. 1088–1093, 2011.
[7] N. Erbs, *et al.*, "Bringing Order to Digital Libraries: from Keyphrase Extraction to Index Term Assignment", *D-Lib Magazine*, vol. 19, pp. 1–16, 2013.
[8] S.W.K. Chan and J. Franklin, "A Text-Based Decision Support System for Financial Sequence Prediction", *Decision Support Systems*, vol. 52, pp. 189–1981, 2011.
[9] A.A.I.N.E. Karyawati, E. Winarko, Azhari and A. Harjoko, "Ontology-based Why-Question Analysis Using Lexico-Syntactic Patterns", *International Journal of Electrical and Computer Engineering (IJECE*), vol. 5(2), pp. 318-332, 2015.
[10] M. Danilevsky, *et al.*, "KERT: Automatic Extraction and Ranking of Topical Keyphrases from Content-Representative Document Titles", http://arxiv.org/abs/1306.0271, accessed July 2014.
[11] M. Arshad, "Implementation of Kea -Keyphrase Extraction Algorithm by Using Bisecting K-Means Clustering Technique for Large and Dynamic Data", *International Journal of Advanced Technology and Engineering Research*, 2009. IJATER 2009,pp. 134–136.
[12] E.B. Foroutan and H. Khotanlou, "Improving Semantic Clustering Using with Ontology and Rules", *International Journal of Electrical and Computer Engineering (IJECE)*,vol. 4(4), pp. 548-556, 2014.
[13] W. Wang, *et. al.*, "Exploring Hypergraph-Based Semi-Supervised Ranking for Query-Oriented Summarization," *Information Sciences*, vol. 237, pp. 271–286, 2013.
[14] M.V. Kumar, *et al.*, "Analysis of Intelligent Data Mining for Information Extraction using Java", *Journal of Computer Science*, vol. 9, pp. 1451–1455, 2013.
[15] A.M. Saif and M.J.A. Aziz, "Collocation Extraction from Arabic Corpus", *Journal of Computer Science*, vol. 7, pp. 6–11, 2011.
[16] K. Chekima and P. Anthony, "Categorizer Agent for Electronic Computer Science Academic Papers", *American Journal of Economics and Business Admnistration*, vol. 3, pp. 213–2181, 2011.
[17] M. Thangaraj and V. Gayathri, "A Context-Based Technique Using Tag-Tree for an Effective Retrieval", *Journal of Computer Science*, vol. 9, pp. 1602–1617, 2013.
[18] A.K. Uysal and S. Gunal, "The Impact of Preprocessing on Text Classification", *Information Processing & Management*, vol. 50, pp. 104–112, 2014.
[19] M.A. Martínez-Prieto, *et al.*, "Natural Language Compression on Edge-Guided Text Preprocessing", *Information Sciences*, vol. 181,pp. 5387–5411, 2011.
[20] H. Ogura, *et.al.*, "Comparison of Metrics for Feature Selection in Imbalanced Text Classification", *Expert Systems with Applications*, vol. 38, pp. 4978–4989, 2011.
[21] R. Giegerich and S. Kurtz, "From Ukkonen to McCreight and Weiner : A Unifying View of Linear-Time Suffix Tree Construction", *Algorithmica*, vol. 19, pp. 331–353, 1997.
[22] R. Pampapathi, *et al.*, "A Suffix Tree Approach to Anti-Spam Email Filtering", *Machine Learning*, vol. 65,pp. 309-338, 2006.

BIOGRAPHIES OF AUTHORS

Ionia Veritawati received a Master Degree on Software Engineering from Bandung Institute of Technology, Indonesia, in 1999. She is currently Ph.D student of Computer Science Faculty in University of Indonesia. Her areas of interest include Machine Learning, Text Mining, Information Retrieval and Software Engineering.

Ito Wasito received a Master Degree and Ph. D on Data Mining from University of London, 1996 and 2003, respectively. He is currently Senior Lecturer and researcher of University of Indonesia, Faculty of Computer Science. His areas of interest *include* Machine Learning Application, Data Mining, Bioinformatics and Biomedical Informatics.

T. Basaruddin received a Master Degree and Ph.D on Computing from University of Manchester (UK) in 1988 and 1990 respectively. He is currently Professor and researcher of University of Indonesia, Faculty of Computer Science. His areas of interest include Numerical Linear Algebra and Applications