ISSN 2007-9737

An Ensemble of Automatic Keyword Extractors: TextRank, RAKE and TAKE

Tayfun Pay[1], Stephen Lucci[2], James L. Cox[3]

[1,3] Computer Science Department,
Graduate Center of New York, New York,
United States

[2] Computer Science Department,
The City College of New York, New York,
United States

[3] Brooklyn College of New York,
Computer and Information Science Department, Brooklyn,
United States

tpay@gradcenter.cuny.edu, lucci.stephen@gmail.com, cox@sci.brooklyn.cuny.edu

Abstract. We construct an ensemble method for automatic keyword extraction from single documents. We utilize three different unsupervised automatic keyword extractors in building our ensemble method. These three approaches provide candidate keywords for the ensemble method without using their respective threshold functions. The ensemble method combines these candidate keywords and recomputes their scores after applying pruning heuristics. It then extracts keywords by employing dynamic threshold functions. We analyze the performance of our ensemble method by using all parts of the Inspect data set. Our ensemble method achieved a better overall performance when compared to the automatic keyword extractors that were used in its development as well as to some recent automatic keyword extraction methods.

Keywords. Data mining, text mining, text analysis, ensemble methods.

1 Introduction

This paper is an expansion of the work that was originally presented in [22]. This extended version includes a thorough explanation of the automatic keyword extractors that are used in building our ensemble method. We also provide a sample input and output for our ensemble method as well as the methods used in its construction. We present the performance of our ensemble method on additional data sets. We also compare its performance with some recent automatic keyword extractors.

There has been enormous amount of data that has been generated in recent years and there is a need to process this data for different purposes. Simultaneously, there has been a growing interest in designing keyword[1] extractors that unearth words or sequence of words that concisely represent a document.

These automatic keyword extraction approaches could be graph based [14], statistics based [18], clustering based [16], linguistic based [11], or other [10]. They could also be some combination of the aforementioned methods as in [20, 5, 12]. There has been also various studies done using semi supervised [1] as well as supervised [15] approaches.

Extracting more significant keywords is important for many different tasks in big data such

[1] We use the words, keyword and keyphrase interchangeably although some authors refer to the former as having a single word and the latter as having more than one word.

as classification [7], clustering [28], indexing [9] and data-analysis [6]. For instance, when more significant keywords are extracted then the subsequently utilized classification algorithms could potentially place the documents into more relevant categories. Similarly, more significant keywords could conceivably assist the clustering algorithms to create more appropriate clusters with the given documents. Additionally, when more significant keywords are extracted then they could possibly be used to decide on the placement of the document within the given database more accurately. There is basically no down side to extracting more significant keywords.

There has also been growing interest in ensemble based machine learning methods, such as the ones in [24, 29, 2]. An ensemble based machine learning approach combines several machine learning methods, wherein each one is ran independently with the same input and the output is then agreed upon by merging their outcomes. If a simple majority opinion is obtained among these approaches, it is referred to as hard voting. Instead, if each approach assigns a probability of class membership, then the average probability obtained is referred to as soft voting. It has been recognized that the performance of ensemble based machine learning methods are often better than any single approach used in their construction. This is particularly true when these machine learning approaches are autonomous from each other, such that they make uncorrelated errors.

In this light, we decided to construct an ensemble method for automatic keyword extraction. We employ the following unsupervised automatic keyword extractors in the construction of our ensemble method: TextRank [19], RAKE [23] and TAKE [21]. In the following section, we discuss how these automatic keyword extractors work. We then explain how to construct our ensemble method.

In the two sections that follow, we introduce the Inspect data set from [13] and our metrics for measuring the performance of our approaches; and go through a sample input and output to the three automatic keyword extractors as well as to our ensemble method. We then analyze the performance of our ensemble method and compare it to the methods used in its construction as well as to other automatic keyword extractors. Finally, we conclude with prospects for our current and future work on this topic.

2 Automatic Keyword Extractors

2.1 Text-Rank

Text-Rank [19] utilizes a part of speech (pos) tagger to assign a pos-tag for each word. It only considers adjectives and nouns as possible keyword components. These words are treated as vertices in a graph wherein an edge is drawn between each word that appears within a given co-occurrence window. A co-occurrence window of size n for a given word consists of the $n-1$ words that appear to the left and to the right of it. Then the ranking algorithm, page-rank [4], is executed until convergence is reached. The top third scoring words are selected for post-processing. At this stage, multi-word keywords are constructed by consulting the original text to determine if any of the selected words appear next to each other. Any word that does not appear next to another word in the list of the top third scoring words of the document are extracted as single-word keywords, and the rest are extracted as multi-word keywords.

2.2 RAKE

RAKE (Rapid Automatic Keyword Extraction) [23] utilizes a stop-list to locate candidate keywords. Any sequence of words that appear between two stop-list words and/or punctuation marks are marked as candidate keywords. Then the frequency and the degree values of each word in the list of candidate keywords are calculated. The frequency of a word is the total number of its occurrences within the list of candidate keywords. The degree of a word is the total number of words that it appears with, within the list of candidate keywords. Then each word is assigned a score of degree over frequency. The cumulative score of each candidate keyword is computed by summing up the scores of the words that it contains. The top third scoring candidate keywords are extracted as keywords.

2.3 TAKE

TAKE (Totally Automated Keyword Extraction) [21] utilizes a pos-tagger to assign a pos-tag for each word and then marks all noun-phrases as candidate keywords. A noun-phrase consists of zero or more adjectives followed by one or more nouns. Then all candidate keywords are filtered out in the following manner: 1) if they contain a word from a stop-list and 2) if they contain only one word with a frequency of one that does not appear in the first ten-percent of the document. TAKE calculates the candidate keyword scores in the same manner as RAKE. All candidate keywords that have a score higher than the value computed by the dynamic threshold function are extracted as keywords.

3 Ensemble of Automatic Keyword Extractors

There are several utility functions that need to be provided and several parameters that need to be set for the automatic keyword extractors in question to execute. First of all, RAKE and TAKE need access to a stop-list, the former needs a stop-list to be able to come up with candidate keywords and the latter needs one to filter the list of candidate keywords.

We use the fox stop-list from [8] for the aforementioned purposes. Second of all, TAKE and TextRank need a pos-tag for each word to be able to construct a set of candidate keywords. We use the default pos-tagger from the NLTK library [3] for this purpose. Finally, we set the co-occurrence window in TextRank to two.

Our ensemble method consists of the following four stages: A) Receiving a list of candidate keywords from each automatic keyword extractor. B) Filtering the list of candidate keywords according to pruning heuristics. C) Combining and recalculating the scores of candidate keywords. D) Applying a dynamic threshold function to extract keywords.

3.1 Candidate Keyword Selection

We remove the threshold function of each automatic keyword extractor. In doing so, the ensemble method gets the total set of candidate keywords found by each approach. The scores of each candidate keyword for each approach are normalized by dividing them by the highest score within their set. This provides the ensemble method with a list of candidate keywords for each approach with a possible score that is greater than zero and less than or equal to one.

There are various reasons why we remove the threshold function of each automatic keyword extractor. In one scenario, some candidate keywords that are discarded, due to scoring lower than the threshold function, might be found by another approach. In another extreme scenario, the normalized score of some candidate keyword might actually be higher in one approach than when compared to another. However, the one with the higher normalized score might not be selected due to scoring lower than the value set by the respective threshold function. Therefore, we take into account all of the candidate keywords from all of the automatic keyword extractors.

3.2 Pruning Heuristics

The ensemble method then applies a pruning heuristic. This heuristic removes any candidate keyword that is located by a single approach and consists of a single word. The rationale here is that there is no statistical significance for a keyword that consists of a single word and was only found by a single keyword extractor.

3.3 Recomputation of Candidate Keyword Scores

The next step is to recompute the candidate keyword scores. As we combine the three different lists of candidate keywords, the scores of candidate keywords located by more than one approach are summed and then multiplied by the total number of approaches that found them.

The rationale behind multiplying the summed scores by the total number of approaches is that their cumulative score might still be too low to pass

the threshold function in the next stage. This can happen even when a candidate keyword has been located by all of the approaches. Therefore, we want to additionally reward the candidate keywords that were found by more than one approach.

3.4 Dynamic Threshold Function

The keywords are finally extracted by applying dynamic threshold functions. In one method, the overall mean is calculated and any candidate keyword that scores higher than the mean is extracted as a keyword for the document. And in the other method, the overall median is utilized for this same purpose.

The dynamic threshold functions do not undermine the contextual properties of each document. This is because they do not treat them in the same way as a static threshold function would have. For instance, if a static threshold function was used, such as extracting top third scoring candidate keywords as keywords, then documents with the same number of candidate keywords would always have the same number of keywords extracted. On the other hand, we let the characteristics of the document determine how many keywords are extracted by employing dynamic threshold functions.

4 Data Set

The data set that we used was introduced in [13], and subsequently used in [19, 23, 21] for evaluating their automatic keyword extraction methods. This data set contains 2000 titles and abstracts for journal papers from *Computer Science and Information Technology*. These items are divided into a training set, validation set and testing set that contain 1000 and two sets of 500 documents, respectively. Only the testing set was used in [22] as well as in [19, 23, 21] since these are unsupervised methods. However, we also used the validation set and the training set for the purposes of testing. We will refer to the testing set, the validation set and the training set as data sets I, II and III, respectively.

Data set I contains 500 documents where there are 4912 manually assigned keywords of which only 3837 are present in the titles and abstracts. Data set II contains 500 documents where there are 4575 manually assigned keywords of which only 3509 are present in the titles and abstracts. Finally, Data set III contains 1000 documents where there are 9788 manually assigned keywords of which only 7552 are present in the titles and abstracts.

We calculated the following parameters: extracted keywords, correct keywords, precision, recall and f-measure:

$$Precision = \frac{correct\ keywords}{extracted\ keywords},$$

$$Recall = \frac{correct\ keywords}{manually\ assigned\ keywords},$$

$$F - Measure = 2 \times \frac{precision \times recall}{precision + recall}.$$

Precision provides us with the percentage of extracted keywords that are correct. Recall provides the percentage of manually assigned keywords that are extracted. As previously noted in [13], both precision and recall are equally important so that they are given the same weight.

We use the total number of manually assigned keywords in the calculation of recall and f-measure, since it was done this way in [19, 23, 21]. Therefore, the highest obtainable recall for data set I is 78.1, for data set II is 76.7 and for data set III is 77.2.

5 Sample Input - Output

Table 1 illustrates the keywords that were extracted by the aforementioned methods and our ensemble approach from the following sample input, which is the title and the abstract of the paper in [25] and is from the Inspect data set [13]. The manually assigned keywords that are present in the title and the abstract are italicized.

"Title: An *optimization* approach to plan for *reusable software components*.

Abstract: It is well acknowledged in *software engineering* that there is a great potential for accomplishing significant *productivity improvements* through the implementation of a successful

software reuse program. On the other hand, such gains are attainable only by instituting detailed *action plans* at both the organizational and program level. Given this need, the paucity of research papers related to planning, and in particular, *optimized planning* is surprising. This research, which is aimed at this gap, brings out an application of *optimization* for the planning of *reusable software components* (SCs). We present a model that selects a set of SCs that must be built, in order to lower development and *adaptation costs*. We also provide implications to *project management* based on *simulation*, an approach that has been adopted by other cost models in the *software engineering* literature. Such a prescriptive model does not exist in the literature."

Table 1. M1 = TextRank, M2 = RAKE, M3= TAKE and EN = Ensemble Method. The boldfaced candidate keywords are manually assigned keywords, N = did not locate, C = located as candidate keyword, E = extracted as keyword

$CandidateKeywords$	$M1$	$M2$	$M3$	EN
successful software reuse program	E	E	E	E
software reuse program	N	N	N	N
software engineering literature	E	E	E	E
software engineering	E	E	E	E
productivity improvements	E	N	E	E
significant productivity improvements	N	E	N	C
optimization approach	E	C	C	E
optimization	C	C	E	E
approach	C	C	C	E
detailed action plans	E	N	N	C
action plans	N	E	E	E
optimized planning	C	C	C	E
reusable software components	E	E	E	E
adaptation costs	E	C	C	E
development costs	N	N	N	N
lower development	N	C	N	C
development	C	N	C	C
project management based	N	E	N	C
project management	N	N	C	C
management	C	N	N	N
simulation	C	C	N	C
program level	C	E	E	E
prescriptive model	E	E	C	E
research papers related	N	E	N	C
research papers	E	N	E	E
organizational	E	C	N	C

In this example, there are 11 manually assigned keywords of which 10 of them are present in the text. (The manually assigned keyword *adaptation costs* is not present in the text as a whole.) TextRank extracted 11 keywords and 4 of them were in the set of manually assigned keywords. RAKE extracted 10 keywords and 3 of them were in the set of manually assigned keywords. TAKE extracted 9 keywords and 5 of them were in the set of manually assigned keywords. The ensemble method extracted 14 keywords and 7 of them were in the set of manually assigned keywords.

It can be observed from table 1 that different automatic keyword extractors obtain different candidate keywords and extract varying keywords for the given document. The ensemble method was also able to extract keywords that were found only as candidate keywords by some automatic keyword extractor. For example, *optimized planning* was merely located as a candidate keyword by all of the automatic keyword extractors, but it was only extracted as a keyword by our ensemble approach. It is also interesting to see that none of the automatic keyword extractors were able to locate the keyword *software reuse program*, but instead located and extracted its superset as a keyword, *successful software reuse program*.

6 Analysis

The performance of our ensemble method along with the automatic keyword extractors that were used in its development on data sets I, II and III are illustrated in tables 2 and 3, 4 and 5, and 6 and 7, respectively.

For all three data sets, our ensemble method has the highest number of correct keywords extracted, highest recall as well as the highest f-measure compared to any of the individual automatic keyword extractors. This is true when either the mean or the median dynamic threshold function is utilized.

The only limitation is with respect to precision, where the method in [21] with the corresponding dynamic threshold function has a higher precision than our ensemble method, but a much lower recall. This is normal with respect to how ensemble based methods work. Because each approach contributed to the ensemble method certain keywords that were different from one another; and some of these keywords matched the manually assigned keywords and some did not.

In turn, this increased the recall, but brought down the precision at the same time. When we look at the results within each data set, we observe that our ensemble method achieved a higher precision with the mean dynamic threshold function compared to its iteration with the median dynamic threshold function. And it achieved a higher recall with the median dynamic threshold function compared to its iteration with the mean dynamic threshold function. This is also true with respect to the the method in [21]. It seems as though if you want higher precision then use the mean dynamic threshold function and if you want higher recall then use the median dynamic threshold function.

We also compare our ensemble method to the most recent automatic keyword extractor that was tested on the Inspec data set. The supervised recurrent neural network method in [26] achieves a precision of 31.0, recall of 27.5 and f-measure of 35.8 on data set I. This result is comparable to the performance of TextRank, but is inferior to the result achieved by our ensemble method.

There are some other methods, namely [16] and [27], that also employed the Inspect data set to test the performance of their automatic keyword extractors. The method in [16] calculated their recalls and consequently their f-measures, using the manually assigned keywords that are present in the abstracts. This provided them with an obtainable recall of 100. We cannot compare our methods to theirs in a fair manner since we cannot recompute their performance metrics given the limited information they provided. On the other hand, the authors of paper [27] created a subset of the Inspec data set by removing any document that does not contain all of its manually assigned keywords. This also provided them with an obtainable recall of 100. Once again, we cannot do a fair comparison between their methods and ours.

7 Conclusion

The quantity of data that is being collected is growing exponentially, and consequently, it becomes important that higher quality keywords be extracted. This is due to the fact that more

Table 2. Precision Recall and F-Measure for Data Set I

method	*precision*	*recall*	*f-measure*
Ensemble - (T=Mean) [22]	46.7	50.9	48.7
Ensemble - (T=Median) [22]	42.1	55.9	48.0
TAKE - (T=Mean) [21]	50.4	33.7	40.4
TAKE - (T=Median) [21]	44.3	46.9	45.6
RAKE - (ka-stoplist) [23]	33.7	41.5	37.2
RAKE - (fox-stoplist) [23]	26.0	42.2	32.1
TextRank (UnD. w=2) [19]	31.2	43.1	36.2
TextRank (UnD. w=3) [19]	28.2	38.6	32.6

Table 3. Extracted and correct keywords for Data Set I

method	*extracted*	*correct*
Ensemble - (T=Mean) [22]	5353	2501
Ensemble - (T=Median) [22]	6523	2746
TAKE - (T=Mean) [21]	3279	1653
TAKE - (T=Median) [21]	5197	2304
RAKE - (ka-stoplist) [23]	6052	2037
RAKE - (fox-stoplist) [23]	7893	2054
TextRank (UnD. w=2) [19]	6784	2116
TextRank (UnD. w=3) [19]	6715	1897

Table 4. Precision Recall and F-Measure for Data Set II

method	*precision*	*recall*	*f-measure*
Ensemble - (T=Mean)	44.0	49.1	46.7
Ensemble - (T=Median)	39.3	54.6	45.7
TAKE - (T=Mean)	45.1	34.5	39.1
TAKE - (T=Median)	40.0	46.2	42.9
RAKE - (fox-stoplist)	23.9	42.4	30.6
TextRank (UnD. w=2)	31.0	45.3	36.8

Table 5. Extracted and correct keywords for Data Set II

method	*extracted*	*correct*
Ensemble - (T=Mean)	5120	2248
Ensemble - (T=Median)	6357	2496
TAKE - (T=Mean)	3501	1578
TAKE - (T=Median)	5291	2114
RAKE - (fox-stoplist)	8124	1942
TextRank (UnD. w=2)	6681	2073

Table 6. Precision Recall and F-Measure for Data Set III

method	*precision*	*recall*	*f-measure*
Ensemble - (T=Mean)	43.8	50.2	46.9
Ensemble - (T=Median)	39.4	55.7	46.1
TAKE - (T=Mean)	46.1	35.4	40.1
TAKE - (T=Median)	40.7	48.4	44.2
RAKE - (fox-stoplist)	24.5	42.0	30.9
TextRank (UnD. w=2)	28.9	43.3	34.7

Table 7. Extracted and correct keywords for Data Set III

method	*extracted*	*correct*
Ensemble - (T=Mean)	11203	4918
Ensemble - (T=Median)	13835	5450
TAKE - (T=Mean)	7509	3464
TAKE - (T=Median)	11631	4734
RAKE - (fox-stoplist)	16748	4107
TextRank (UnD. w=2)	14631	4237

significant keywords would yield better results for the subsequently utilized machine learning algorithms. Additionally, this allows more accurate classification of documents and correct placement in databases.

We presented an unsupervised ensemble method for automatically extracting keywords from single documents. We showed that our ensemble method obtained better overall performance compared to the individual methods used in its development as well as to some of the recent approaches that were studied. We can also infer that our ensemble method is stable since it achieved this performance on three different data sets.

Although our ensemble method achieved better overall performance with respect to how well it found the keywords, it was nonetheless computationally slow because of TextRank. Our studies with regard to this has been presented in [17], where we construct yet another ensemble method without using TextRank and achieve comparable results, but with faster computation times. We wish to continue this line of research, where we will explore other types of ensemble methods for keyword extraction and experiment on different data sets.

References

1. **Aggarwal, A., Sharma, C., Jain, M., & Jain, A. (2018).** Semi supervised graph based keyword extraction using lexical chains and centrality measures. *Computación y Sistemas*, Vol. 22, No. 4.
2. **Alam, T., Ahmed, C. F., Zahin, S. A., Khan, M. A. H., & Islam, M. T. (2018).** An effective ensemble method for multi-class classification and regression for imbalanced data. *Industrial Conference on Data Mining*, pp. 59–74.
3. **Bird, S. & Loper, E. (2002).** Nltk: the natural language toolkit. *ETMTNLP 02 Proceedings of the ACL-02.*, Vol. 1, pp. 63–70.
4. **Brin, S. & Page, L. (1998).** The anatomy of a large-scale hypertextual web search engine. *Computer Networks and ISDN Systems.*, Vol. 30, pp. 1–7.
5. **Campos, R., Mangaravite, V., Pasquali, A., J., A. M., Nunes, C., & Jatowt, A. (2018).** A text feature based automatic keyword extraction method for single documents. *European Conference on Information Retrieval*, pp. 684–691.
6. **Castillo, E., Cervantes, O., & Vilarino, D. (2017).** Text analysis using different graph-based representations. *Computación y Sistemas*, Vol. 21, No. 4, pp. 581–599.
7. **Castro, D., Adame, Y., Pelaez, M., & Muñoz, R. (2017).** Authorship verification, neighborhood-based classification. *Computación y Sistemas*, Vol. 21, No. 2.
8. **Fox, C. (1989).** A stop list for general text. *ACM SIGIR Forum*, Vol. 24, pp. 19–21.
9. **Gelbukh, A., Sidorov, G., & Guzmán-Arenas, A. (2005).** Document indexing with a concept hierarchy. *Computación y Sistemas*, Vol. 8, No. 4, pp. 281–292.
10. **Giamblanco, N. & Siddavaatam, P. (2017).** Keyword and keyphrase extraction using newton's law of universal gravitation. *Electrical and Computer Engineering (CCECE), 2017 IEEE 30th Canadian Conference on*, IEEE, pp. 1–4.
11. **Hu, X. & Wu, B. (2006).** Automatic keyword extraction using linguistic features. *Data Mining Workshops, 2006. ICDM Workshops 2006. Sixth IEEE International Conference on*, IEEE, pp. 19–23.
12. **Huh, J. (2018).** Big data analysis for personalized health activities: Machine learning processing for automatic keyword extraction approach. *Symmetry*, Vol. 10, pp. 93.
13. **Hulth, A. (2003).** Improved automatic keyword extraction given more linguistic knowledge. *Proceedings of the 2003 conference on empirical methods in natural language processing.*, pp. 216–223.
14. **Litvak, M. & Last, M. (2008).** Graph-based keyword extraction for single-document summarization. *Proceedings of the workshop on Multi-source Multilingual Information Extraction and Summarization*, Association for Computational Linguistics, pp. 17–24.

15. **Liu, F., Liu, F., & Liu, Y. (2008).** Automatic keyword extraction for the meeting corpus using supervised approach and bigram expansion. *Spoken Language Technology Workshop, 2008. SLT 2008. IEEE*, IEEE, pp. 181–184.
16. **Liu, Z., Li, P., Zheng, Y., & Sun, M. (2009).** Clustering to find exemplar terms for keyphrase extraction. *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume 1*, Association for Computational Linguistics, pp. 257–266.
17. **Lucci, S., Cox, J. L., & Pay, T. (2018).** Another perspective on ensemble methods for automatic keyword extraction. *2018 IEEE International Conference on Big Data (Big Data)*, IEEE, pp. 5424–5426.
18. **Matsuo, Y. & Ishizuka, M. (2004).** Keyword extraction from a single document using word co-occurrence statistical information. *International Journal on Artificial Intelligence Tools*, Vol. 13, No. 01, pp. 157–169.
19. **Mihalcea, R. & Tarau, P. (2004).** Textrank: Bringing order into texts. *Association for Computational Linguistics.*
20. **Naidu, R., Bharti, S. K., Babu, K. S., & Mohapatra, R. K. (2018).** Text summarization with automatic keyword extraction in telugu e-newspapers. *Smart Computing and Informatics*, pp. 555–564.
21. **Pay, T. (2016).** Totally automated keyword extraction. *2016 IEEE International Conference on Big Data (Big Data)*, IEEE, pp. 3859–3863.
22. **Pay, T. & Lucci, S. (2017).** Automatic keyword extraction: An ensemble method. *2017 IEEE International Conference on Big Data (Big Data)*, IEEE, pp. 4816–4818.
23. **Rose, S., Engel, D., Cramer, N., & Cowley, W. (2010).** Automatic keyword extraction from individual documents. *Text Mining.*, pp. 1–20.
24. **Sabzevari, M., Martínez-Muñoz, G., & Suárez, A. (2018).** A two-stage ensemble method for the detection of class-label noise. *Neurocomputing*, Vol. 275, pp. 2374–2383.
25. **Sundarraj, R. (2002).** An optimization approach to plan for reusable software components. *European Journal of Operational Research*, Vol. 142, pp. 128–137.
26. **Villmow, J., Wrzalik, M., & Krechel, D. (2018).** Automatic keyphrase extraction using recurrent neural networks. *International Conference on Machine Learning and Data Mining in Pattern Recognition*, pp. 210–217.
27. **Wang, R., Liu, W., & McDonald, C. (2014).** Corpus-independent generic keyphrase extraction using word embedding vectors. *Software Engineering Research Conference*, volume 39, pp. 1–8.
28. **Yagunova, E., Pronoza, E., & Kochetkova, N. (2018).** Construction of paraphrase graphs as a means of news clusters extraction. *Computación y Sistemas*, Vol. 22, No. 4.
29. **Zhang, Y., Fu, K., Sun, H., Sun, X., Zheng, X., & Wang, H. (2018).** A multi-model ensemble method based on convolutional neural networks for aircraft detection in large remote sensing images. *Remote Sensing Letters*, Vol. 9, pp. 11–20.

Article received on 19/01/2019; accepted on 12/02/2019.
Corresponding author is Tayfun Pay.