I.J. Intelligent Systems and Applications, **2017, 11, 26-33**
Published Online November 2017 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijisa.2017.11.04

Document Summarization using TextRank and Semantic Network

Ahmad Ashari
Department of Computer Science and Electronics,
Faculty of Mathematics and Natural Sciences, Universitas Gadjah Mada
E-mail: ashari@ugm.ac.id

Mardhani Riasetiawan
Department of Computer Science and Electronics,
Faculty of Mathematics and Natural Sciences, Universitas Gadjah Mada
E-mail: mardhani@ugm.ac.id

Received: 16 March 2017; Accepted: 08 June 2017; Published: 08 November 2017

***Abstract*—**The research has implemented document summarizing system uses TextRank algorithms and Semantic Networks and Corpus Statistics. The use of TextRank allows extraction of the main phrases of a document that used as a sentence in the summary output. The TextRank consists of several processes, namely tokenization sentence, the establishment of a graph, the edge value calculation algorithms using Semantic Networks and Corpus Statistics, vertex value calculation, sorting vertex value, and the creation of a summary. Testing has done by calculating the recall, precision, and F-Score of the summary using methods ROUGE-N to measure the quality of the system output. The quality of the summaries influenced by the style of writing, the selection of words and symbols in the document, as well as the length of the summary output of the system. The largest value of the F-Score is 10% of the length ta of the document with the F-Score 0.1635 and 150 words with the F-Score 0.1623**.**

***Index Terms*—**TextRank, Semantic Network, Document Summarization, Rouge-N, F-Score.

I. Introduction

The development of technology and the dissemination of information has converged into the wide variety of data and information [1,2,3]. The information dissemination is accompanied by the proliferation of available information, such as documents. However, increasing the amount of information available does not always make it easy for the reader. The problem occurs because everyone does not have enough time to read through all the information available. Especially after reading, it turns out the information available in a document is not following the desired information reader. Therefore, we need a document summarization approach to give the reader a general overview of a document before reading. To resolve this problem, do the method of reading the document at a glance (skimming) [4,5]. But sometimes they make some of the information on the document becomes hard to understand because the information may require other information previously unreadable due to come into skimming.

In addition to skimming, another method that can be done is to read a summary of a document. Summary is a representation of a document containing the main focus of the document. A summary can improve the effectiveness of the reader in searching and finding the desired document [6]. However, a document generally do not have a summary for a summary of a document is time consuming and cost [7]. In addition, a summary document is not easy because a summary should be able to represent the whole of the contents of the document.

The research implemented document summarization system uses TextRank algorithms, Semantic Networks and Corpus Statistics. The use of TextRank allows extraction of the main phrases of a document that has used as a sentence in the summary output. Documents Summarization on TextRank consists of several processes, namely tokenization sentence, the establishment of a graph, the edge value calculation algorithms using Semantic Networks and Corpus Statistics, vertex value calculation, sorting vertex value, and the creation of a summary.

II. Related Works

Mihalcea and Tarau [8] proposed a framework called TextRank to model the document into a form that every vertex in the graph represents graph word or phrase in the document. For the determination of the weight or the edge on TextRank, can be used several algorithms similarity measures such as String Kernel, Cosine Similarity, Levenshtein, and others [4,5,9]. The basic idea of this TextRank is voting, or recommendations associated with each vertex will give vertex are calling a vote. The higher the vote on a vertex, the vertex increasingly important in presenting the document. TextRank is the basis of the algorithm to be used in this study.

Zikra [10] to implement document summarization system on research using the PageRank algorithm which is the basis of TextRank. Zikra research using measures such as cosine similarity Similarity and Content Similarity Overlap. The Cosine Similarity works in each sentence should represented into a vector which is usually filled with a term-frequency of sentences are then calculated the distance, while Content Similarity Overlap will calculate how much the same words between sentences.

Hoffmann and Pham [11] conducted a study on the system with a document viewer method by topic using the tree and graft. Unlike the TextRank that uses similarity measure for defining relationships between vertex, Hoffmann and Pham uses a rule-based system called KAFDIS to defines relationships among vertices so that relations may be between vertex must first be determined. KAFDIS has relation such as elaboration, components, comparison, assumptions, consequences, and others. The rules on KAFDIS should be able to detect the structure of the sentence in the document and its relation to the other sentences in the document using the POS Tag.

The study proposed by Park [12], Aji and Kaimal [13] using a model representation of the document in the form of a model vector or matrix containing the term-frequency. Park in his research using Non-negative Matrix Factorization (NMF) and Pseudo Relevance Feedback (PRF) to document summarization. In this study, conducted summarizing method with the query-based approach, so as to produce a summary that is relevant to a given user query. While Aji and Kaimal [13] in his research using Point wise Positive Mutual Information (PPMI) which is a variant of Point wise Mutual Information (PMI) algorithm.

Aliguliyev [1] proposed the use of methods summarized document by clustering the sentences in the document. Just like the previous studies, this study uses a model vector or matrix as a representation of the document using the weighting method TF-ISF (Term Frequency-Inverse Sentence Frequency) which is an algorithm derived from the TF-IDF to perform weighting on the term-frequency vector. Clustering method used is K-Mean by using Cosine Similarity to calculate the distance between the data to the cluster center. After the cluster obtained, sentences representation of each cluster is taken to be the summary.

Fang, et.al [16] proposed CoRank as a word-senteces co-ranking model. The model combines the word-senteces with the graph-based with unsupervised ranking model. The approach has produces document summary. Abdi, et.al [17] proposed a query based summarization with word semantic relation and syntact composition. The approach addressed the fail when produces the meanring in statistical methods. The query based implemented the semantic and syntatic similiarity between sentences and query. The approach key phrases identification and natching with information and literatures based on hierarchical structure has been proposed to produce document collection and summarization [18]. The adaptive neuro fuzzy inference system (ANFIS) has proposed as text summarization approach. ANFIS has capabilities to increase the precission, recall and F-measure [19].

In research to be conducted, TextRank algorithm implemented by the similarity measure in the form of algorithms Semantic Networks and Corpus Statistics. The use of algorithms Semantic Networks and Corpus Statistic This allows the system to detect the sentence structure to improve the accuracy in calculating the similarity between sentences.

III. Methods

A general description of the system can be seen in Fig.1. The schematic design of the system, there are two main modules, namely TextRank module, and the module Sentence Similarity. In TextRank module, the document has processed into an undirected graph. Documents that have been processed into sentences on TextRank module is then delivered to module Sentence Similarity to quantify the similarity between the sentence that was then sent back to the module TextRank to be the edge. After all processes in TextRank completed, then the sentences issued on the graph in the form of plain text form of a summary.

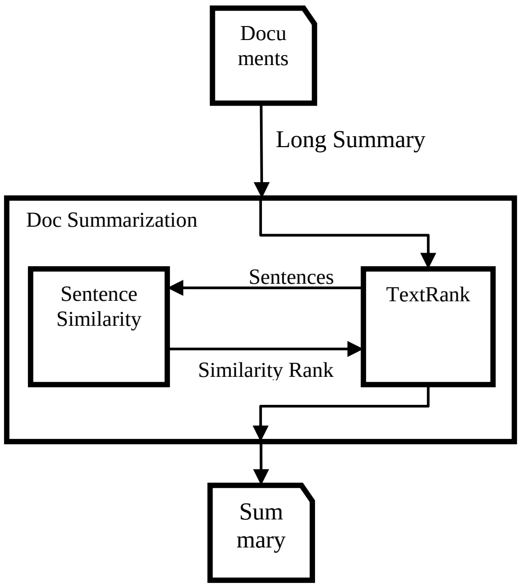


Fig.1. System Design

TextRank module design includes several major processes, namely splitting the document into sentences, the creation of graphs, calculation of the degree of similarity between sentences, and counting rank within the graph [8,12,14,15]. Chronology of the process of this module begins with the receipt of the documents that has been extracted. Documents in the form of plain text that has entered into the system will be processed first by the type of encoding specified in the input parameters, then proceed with the next process, namely tokenization. In the process tokenization, each sentence in the document is detected then split into tokens sentence. After tokens sentences were formed, then the next process is the manufacture of undirected graph with each vertex containing tokens sentences preconceived and Initial Value (IV), which is 1. Value edge of the graph is obtained from the process the next module Sentence Similarity.

The next process is to calculate the values at each vertex. Vertex value at iteration n and n + 1 will be recorded to gain error rate (errors) by calculating the difference. If the value of the error is already less than the specified threshold, then continue the next process, if not, repeat the calculation of the value of the vertex. This vertex value calculation process can be performed in parallel because the value of a vertex at iteration n independently of each other due vertex values used for iteration n + 1 is the vertex at iteration n. The graph showing the vertex, the next process is the vertex sort by a value ranging from the largest (descending order). Sentences on vertices that has been ordered are then taken to be used as a summary in the form of plain text with a word length specified in the input parameters.

This similarity accepts input values in the form of a couple of sentences from TextRank module. Every sentence received tokenization process has been conducted so that it will be a collection (set) words. The next process is the calculation of semantic similarity and word order similarity. The output of the counting process semantic similarity is the similarity values between 0 and 1, as well as the word order similarity. After the value of the two processes previously obtained, then the two values will be combined to obtain the overall similarity value which would then be sent to the module to be used as a value TextRank edge on graph.

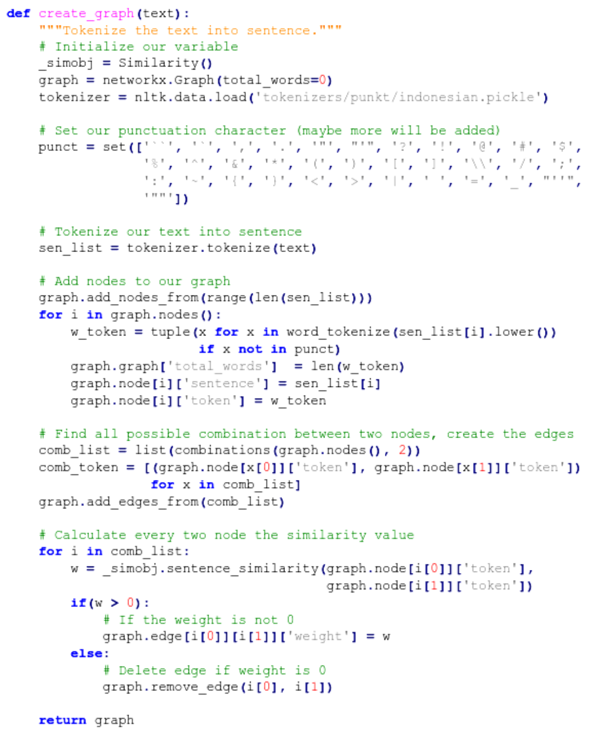
IV. Implementation

Implementation of this system is divided into two large modules as described in the previous section, the module TextRank and Sentence Similarity. TextRank module is responsible for handling and document summarization process. Sentence Similarity module tasked with finding a similarity between the sentence that was subsequently used on the module TextRank. However, this implementation is not limited to these two major modules, are also add-on modules such as modules that handle input, output, and the parameters of the program, as well as modules that handle Corpus used in module Sentence Similarity.

A. Implementation of TextRank Module

In the implementation of this TextRank module, used Python library named NetworkX used to handle the graph creation process. This NetworkX using Python dictionary-based data types to accommodate the data vertex, edge, and graphs so that the data can be accessed quickly. Besides NetworkX, one NLTK module is also used in this module to handle tokenization sentence, namely Punct.

This function was originally named create_graph call a function that will produce a non-directional graph includes a token sentence and edge that contains the similarity between sentences. The resulting graph will then be calculated vertex value by calling a function named calculate_rank which will produce a list containing the value of the vertex, with the index of the list is the same as the index of vertices in a graph. After the vertex values obtained the list, then the list will be further sorted in descending order based on the contents of later retrieved index. List index gained will be used in the next process for the preparation of the summary by taking the sentence on the vertex with the corresponding index in the list of indexes that had previously has sequenced. The length of the word also needs to be calculated so as not to vary much with word length specified in the input parameter (sum_length). This function will then produce the output of string which will then sent to the module that handles input and output to issued in the form of a file.



```
def create_graph(text):
    """Tokenize the text into sentence."""
    # Initialize our variable
    _simobj = Similarity()
    graph = networkx.Graph(total_words=0)
    tokenizer = nltk.data.load('tokenizers/punkt/indonesian.pickle')

    # Set our punctuation character (maybe more will be added)
    punct = set(['``', '`', ',', '.', '"', "'", '?', '!', '@', '#', '$',
                 '%', '^', '&', '*', '(', ')', '[', ']', '\\', '/', ';',
                 ':', '~', '{', '}', '<', '>', '|', ' ', '=', '_', "''",
                 '""'])

    # Tokenize our text into sentence
    sen_list = tokenizer.tokenize(text)

    # Add nodes to our graph
    graph.add_nodes_from(range(len(sen_list)))
    for i in graph.nodes():
        w_token = tuple(x for x in word_tokenize(sen_list[i].lower())
                        if x not in punct)
        graph.graph['total_words']  = len(w_token)
        graph.node[i]['sentence'] = sen_list[i]
        graph.node[i]['token'] = w_token

    # Find all possible combination between two nodes, create the edges
    comb_list = list(combinations(graph.nodes(), 2))
    comb_token = [(graph.node[x[0]]['token'], graph.node[x[1]]['token'])
                  for x in comb_list]
    graph.add_edges_from(comb_list)

    # Calculate every two node the similarity value
    for i in comb_list:
        w = _simobj.sentence_similarity(graph.node[i[0]]['token'],
                                        graph.node[i[1]]['token'])
        if(w > 0):
            # If the weight is not 0
            graph.edge[i[0]][i[1]]['weight'] = w
        else:
            # Delete edge if weight is 0
            graph.remove_edge(i[0], i[1])

    return graph
```

Fig.2. Create_graph Function

Later in create_graph function as shown in Fig.2, initially to initialize the object of Sentence Similarity modules in a variable named _simobj that has used to compute the similarity between sentences. NetworkX used to create graphs that contain total_words property used to accommodate the total words in the input document to be used in determining the length of the summary output. To process tokenization in the document into sentences, use one of the modules NLTK, Punct which was previously train to detect the expressions on the input document. Furthermore, punct variable that contains the list of punctuation symbols used as a basis to remove symbols of punctuation in the sentence. The next step is to fill the vertices in a graph with a token sentences and words from the sentences that will be the token is used to calculate the similarity between sentences in Sentence Similarity module. After all and vertex filled, then all the vertices in a graph will be connected with each other with the edge whose value is calculated based on the similarity of the sentence on the vertex-related uses on _simobj functions that have previously has initialized. Edge that contains the value 0 will be removed as vertex-vertex connected has no similarity at all. This function will then produce the output of which is an object graph of NetworkX.

The next function is a function calculate_rank tasked to calculate the value of the input vertex of the. This function will perform as many repetitions max_iter times to calculate the vertex value. Maximum iterations given that the function does not perform repetition does not end because the threshold has not met. Furthermore, the values of vertices in a graph represented with a list of named val and new_val with the same index of the list with an index of vertices in a graph. If the sum of the difference in the overall vertex at iteration n (val) and n + 1 (new_val) -in this case is a delta-less than the specified error threshold, then the iteration will be terminated and the function will return the list new_val iteration where these conditions occur , If the function already reached the limit of maximum iterations, the function will be stopped and return the list new_val.

B. Implementation of Similarity Module

Sentence Similarity on the implementation of this module, WordNet which is one of the modules of NLTK used as a database to search for related words. This module has implemented as an object for easy storing and accessing results similarity in function words in it. Suppose that in a document the word 'eat' and 'rice' appear 10 times, if the results of the calculation of similarity between the word 'eat' and 'rice' has not saved, then the system will search for the similarity between the words as much as 10 times. It has caused the program runtime is significantly slower due to the process of finding similarity between the words is an expensive process regarding of computing and runtime. In addition to computing and runtime issues, easy access to various global variables in the object is also a consideration in the implementation of this module because this global variable will be accessed and used on many of the functions in this module.

Sim_sets variable is a variable of type Python dictionary that can store data in the form of key-value. This variable is used to store a collection of similarity between the words that has previously calculated. Key stored here is a list containing pairs (kata1, kata2) and value here shows similarity between kata1 and kata2. Besides sim_sets, stored too many variables that were used in other functions such as language, corpus, alpha, beta, threshold, and used in calculating the similarity measure.

Sentence_similarity who receive such couples enter a sentence that has become tokens previously stored in the vertex of the graph. This function will call three other functions, namely comb_similarity which will produce sim_sets calculated using the function word_similarity with the input of all possible unique combinations of words in a sentence input, then semantic_similarity used to calculate the value of semantic similarity of the sentence, and word_order_similarity that will count value based on sequence similarity sentence. In the end, this function will provide a combined output of semantic similarity between the value and the value of word order similarity in the form of overall similarity.

Word_similarity which has the task to calculate the degree of similarity between the two words of input by considering the path length and depth along the nearest parent (lowest common subsume) between two words in WordNet. First, the function will search synset in WordNet two input words which, if not found, then the value 0 will be returned. Furthermore, the results of the search will produce two previous synset, synset list of two words given that then searched all the possible combinations and calculated the length of the path between synset. The length of the path of synset has the possibility of providing value None, which means two synset were located in a different tree. Suppose the word 'street' is a verb and the word 'road' which is a noun. Both words are equally the word 'road', but has different properties so that the two has not compared, therefore the length of the path between the two synset None, or no path connecting the two. If the path length of the entire combination synset is None, then the function will return a value of 0. Otherwise, it would have taken a couple of synset that has the shortest path length.

C. Implementation of Corpus Statistic

The Corpus object initialization, the function will read the file corpus that has been provided by the corpus of research results Dinakaramani et al. [3], which contains 10,000 sentences tokenized result are 262,330 tokens. Another module is a module that regulates the input, output, and display program. This module uses Python library named Click to set the input parameters in the CLI. Partially parameters set by the Book, input parameter to accept the document input, output summary and length summaries. Click can set the parameter name, parameter description, the default value of the parameter and others.

D. Testing

Tests on this system using Java implementation of such a program named ROUGE 2.0. The program will accept input in the form of a summary document as plain text. The summary input document has separated into two folders named reference to hold summaries or summaries ideal man, and a folder has named system to accommodate a summary of artificial systems.

Once the summary document has loaded, the next is the configuration settings ROUGE 2.0 which has located in a file with the name rouge.properties. The thing to note in this study is the parameter project.dir, ngram, output, and outputfile. Project.dir parameter is the name of the folder that holds folders and system reference, ngram is the number of pairs of word n-gram that will be used in ROUGE-N, the output is output from ROUGE type 2.0, and output file is the output file of ROUGE 2.0 if the previous output parameters filled with files. After configuration is complete, run the program named rouge2.0.jar with the command java -jar rouge2.0.jar, then the program will automatically calculate the value of the F-Score, re-call and precision.

V. Result and Discussion

The testing is done in two stages, the first stage is to test the summary document by the length parameter summary in the form of a percentage of the document, and the second stage is to examines the summary document by the length parameter summary form of the number of words.

Tests using percentages do as much as 6 times the length of the percentage of a summary document has tested is 5%, 10%, 15%, 20%, 25% and 30%. The test results ROUGE-N will generate recall, precision and F-Score which will then be averaged by columns based on the length of the summary. The test results shows in Table 1.

Table 1. Summarization (Percentage)

Average	**Summarization (%)**					
	5%	**10%**	**15%**	**20%**	**25%**	**30%**
Recall	0,1530	0,2404	0,2811	0,3225	0,3576	0,3930
Precision	0,1655	0,1289	0,1005	0,0868	0,0778	0,0706
F-Score	0,1540	0,1635	0,1448	0,1340	0,1257	0,1178

Tests using the word count done as much as 6 times the length of each summary is 50, 100, 150, 200, 250, and 300 words. Just as before, the test results ROUGE-N generate recall, precision and F-Score which will then be averaged by columns based on the length of the summary. The test results shows in Table 2.

Table 2. Summarization (Length)

Average	**Summarization (%)**					
	50	**100**	**150**	**200**	**250**	**300**
Recall	0,0668	0,1188	0,1698	0,1924	0,2129	0,2315
Precision	0,1925	0,1828	0,1634	0,1400	0,1229	0,1136
F-Score	0,0964	0,1404	0,1623	0,1588	0,1522	0,1491

From the second testing that has been done, the recall value in the second test will increase along with increasing the length of the summary. This is due to the longer summary of the output system, the more the words in the summary output system that are relevant to the words in the summary of the man-made causes the expanding recall value obtained. But the great recall value which does not guarantee the accuracy of the summary produced by the system with a summary of man-made, because the recall value simply counting how many relevant value received by the system. Suppose the 1000 data received by the system, already covers 50 of the 50 relevant data, the recall value of the system is 1 (100%) due to the received data already covers all relevant data, although the value obtained is also entering 950 incorrect data.

Fig.3 and Fig.4, the value of precision that results from both tests tend to decrease with increasing the amount of long summary. This is because the longer the output summary of the system, the more the words in the summary output system that are not relevant to a summary of man-made cause a decline in the value of precision given the precision value has obtained by calculating whether all the accepted values is the value of the relevant system. Suppose the data received from the 1000 system, obtained 50 of the 50 relevant data, then the value of precision of the system is simply 50/1000 (5%) although data received already covers all relevant data. This is because the 1000 data received, precision or accuracy of the system are just 50 of the 1000 Data.

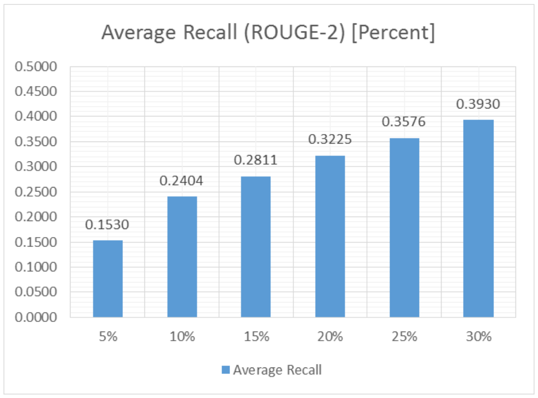


Fig.3. Recall Average Percentage

From the above test results, it can be seen that the precision and recall value inversely with the amount of data in this study the long-summary-received. To overcome this, use the F-Score is the average harmonic of recall and precision. Value F-Score has used to measure the degree of similarity of the summary output with a summary of man-made systems. As for the average chart F-Score of the test shows in Fig.5 for testing with the length of the sentence in the form of percentages, and Fig.6 for testing with the length of the sentence in the form of the word count.

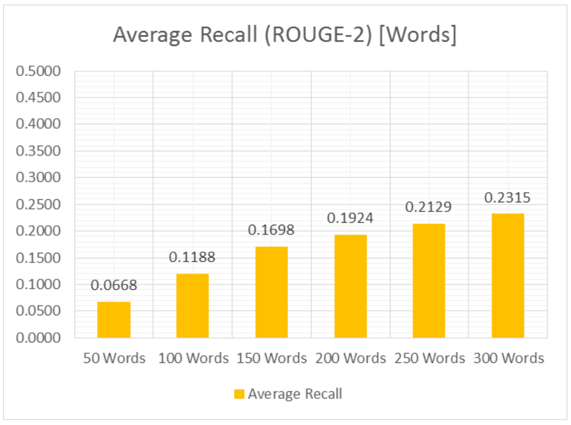


Fig.4. Recall Average Length

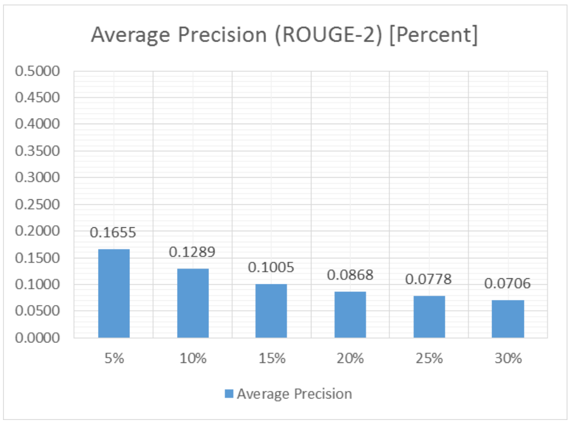


Fig.5. Average Precision (Percentage)

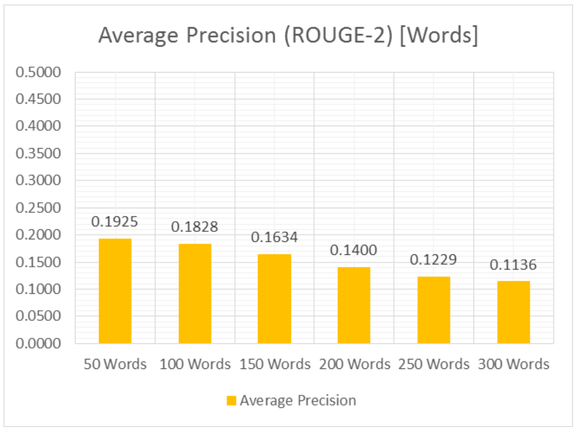


Fig.6. Average Precision (Words)

Fig.7 and Fig.8, shows the average value of the F-Score achieved when the maximum length of the summary is set at 10% of the total words on the document or set at 150 words. This shows that on average the highest proximity to the system summary manmade ideal is achieved when the length of the summary is set at 10% of the total words on the document or set at 150 words. In addition to a long summary, F-Score of the summary is also influenced by the content of the document to be summarized.

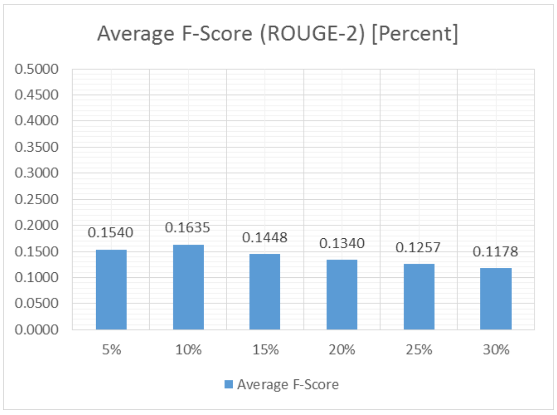


Fig.7. Average F-Score (percentage)

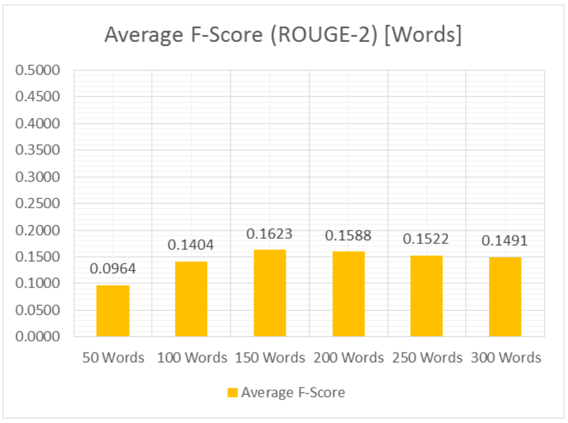


Fig.8. Average F-Score (Words)

VI. Conclusion

Research has been successfully implemented in a system using TextRank algorithm, Semantic Network and Corpus Statistics. The results showed that the value of F-Score Low summary influenced by the style of writing the source document that gives many examples of cases so as to provide a significant impact on the value of the F-Score summary as previously discussed. The result applies to all documents that use this style of writing such works by looking at remembering TextRank majority voting so that the topic with major supporters sentence that will go into a sentence in the summary. Thus, the style of writing is one of the factors that affect the quality of the output summary. The use of symbols and choice of words that are not standard will also affect the results of the summary given on Sentence Similarity module as described earlier, words or symbols that are not on WordNet will be assigned a value of 0 for the sentence.

The results also showed that the value of F-Score on summary influenced by the length of the summary. This is because with the increased length of the summary, increased many words that are not relevant to the summary should ideally lead to the falling value of precision. The fall in the value of precision will significantly impact the value of the F-Score although recall value will continue to increase, given the value of the F-Score has obtained by calculating the average harmonic of recall and precision so that these values must be in the balance to achieve the F-Score high.

Based on test results, obtained the optimal length of the output summary which has an average value of the highest F-Score system that is 10% with a value of 0.1635 if the F-Score lengthy summaries using percentages, and 150 words with a value of 0.1623 if the F-Score long summary uses the word count. The F-Score value generated by the system is still limited to be used as a summary of the document. This is because the summary of the results of the document summarize system can only extract from the sentence contained in the document, unlike the man-made abstract whose contents may have the same meaning but use different words (paraphrasing) so that the ROUGE test can not detect this .

The future work of the reseaech, It should be further investigated the use of other algorithm models or the incorporation of several other algorithm models to calculate the degree of similarity between words. It should be further investigated how the influence and use of other ontology models in addition to WordNet in finding relations between words. Implemented word matching based on HAS-A relation on WordNet and combine it with search based on IS-A relation.

Acknowledgment

Cloud & Grid Technology Working Group has supported the research in facilities and technical assistance. The research has supported by Postgraduate of Computer Science Program, Faculty of Mathematics and Natural Sciences, Universitas Gadjah Mada. The research give acknowledgement to Nurdian Nugroho for his assistance and technical supports.

References

[1] Bond, F., Lim, L.T., Tang, E.K. and Riza, H., 2014, The Combined Wordnet Bahasa, *NUSA: Linguistic studies of languages in and around Indonesia*, 57, 83–100.

[2] Brin, S. and Page, L., 1998, The Anatomy of a Large-Scale Hypertextual Web Search Engine, *Computer Networks and ISDN Systems*, 1-7, 30, 107–117.

[3] Dinakaramani, A., Rashel, F., Luthfi, A. and Manurung, R., 2014, Designing an Indonesian Part of Speech Tagset and Manually Tagged Indonesian Corpus, *2014 International Conference on Asian Language Processing (IALP)*, Kuching.

[4] Li, Y., McLean, D., Bandar, Z., O'Shea, J.D. and Crockett, K., 2006, Sentence Similarity Based on Semantic Nets and Corpus Statistics, *IEEE Transactions on Knowledge and Data Engineering*, 8, 18, 1138–1150

[5] Lin, C.Y., 2004, Rouge: A Package for Automatic Evaluation of Summaries, *Text Summarization Branches Out: Proceedings of the ACL-04 Workshop*, Barcelona.

[6] Aliguliyev, R.M., 2007, Automatic Document Summarization by Sentence Extraction, *Вычислительные технологии*, 5, 12, 5–15

[7] Radev, D.R., Hovy, E. and McKeown, K., 2002, Introduction to the Special Issue on Summarization, *Computational Linguistics*, 4, 28, 399–408.

[8] Mihalcea, R. and Tarau, P., 2004, TextRank: Bringing Order into Texts, Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing, Barcelona.

[9] Li, Y., Bandar, Z.A. and McLean, D., 2003, An Approach for Measuring Semantic Similarity between Words Using Multiple Information Sources, *IEEE Transactions on Knowledge and Data Engineering*, 4, 15, 871–882

[10] Zikra, H., 2009, Sistem Peringkas Teks Otomatis Menggunakan Algoritme Page Rank, *Tesis*, Jurusan Ilmu Komputer FMIPA IPB, Bogor

[11] Hoffmann, A. and Pham, S.B., 2003, Towards Topic-Based Summarization for Interactive Document Viewing, *K-CAP 2003 - Proceedings of the 2nd International Conference on Knowledge Capture*, Sanibel Island.

[12] Park, S., 2009, User-focused Automatic Document Summarization using Non-negative Matrix Factorization and Pseudo Relevance Feedback, *Proceedings of 2009 International Conference on Computer Engineering and Applications (ICCEA 2009)*, Manila.

[13] Aji, S. and Kaimal, R., 2012, Document Summarization Using Positive Pointwise Mutual Information, *International Journal of Computer Science & Information Technology*, 2, 4, 47–55.

[14] Miller, G.A., 1995, WordNet: A Lexical Database for English, *Communications of the ACM*, 11, 38, 39–41

[15] Noor, N.H.M., Sapuan, S. and Bond, F., 2011, Creating the Open Wordnet Bahasa., Proceedings of the 25th Pacific Asia Conference on Language, Information and Computation (PACLIC 25), Singapore.

[16] Fang, C., Mu, D., Deng, Z., Wu, Z., 2016, Word-Sentence co-ranking for automatic extractive text summarization, Expert System with Appliactions, 72,, 189-195.

[17] Abdi, A., Idris, N., Alguliyev, R.M., 2017, Query-based multi-dicuments summarization using linguistic knowledge and content word expansion, Soft Computing, 21(7), 1785-1801.

[18] Di Sciascio, C., Mayr, L., Veas, E., 2017, The 2017 ACM Workshop on Exploratory Search and Intreactive Data Aanalytics, 41-48.

[19] Kumar, Y.J., Kang, F.J., Goh, O.S., Khan, A., 2017, Text Summarization Besd on Classficiation Using ANFIS, Studies in Computational Intelligence, 405-417.

Authors' Profiles

Ahmad Ashari, Dr.techn is senior researcher in Department of Computer Sciences and Electronics, Faculty of Mathematics and Natural Sciences, Universitas Gadjah Mada. His work has been published in several international journals for Computer and Network architecture area. The research focuses are computer network, security and high performance computing.

Mardhani Riasetiawan is researcher in Department of Computer Sciences and Electronics, Faculty of Mathematics and Natural Sciences Universitas Gadjah Mada, Indonesia. Mardhani research focus is in Cloud, Grid and Cluster Infrastructure, Enterprise Data Center, and Big Data.

How to cite this paper: Ahmad Ashari, Mardhani Riasetiawan, "Document Summarization using TextRank and Semantic Network", International Journal of Intelligent Systems and Applications(IJISA), Vol.9, No.11, pp.26-33, 2017. DOI: 10.5815/ijisa.2017.11.04