Information Retrieval in Text-based Document using Fuzzy Logic Approach

Adhiguna Surya - 13509077
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13509077@std.stei.itb.ac.id

***Abstract*—This paper explains the use of fuzzy logic approach in information retrieval. The first method is using fuzzy pattern rule induction (FRIS) to generate rules and patterns for information retrieval. This paper also analyses its performance compared to other information retrieval methods. The second part of the paper covers the application of fuzzy logic approach to text summarisation, implemented using Matlab's fuzzy inference system (FIS). This paper also covers the analysis of fuzzy-based text summarisation compared to other text summarisation methods.**

***Index Terms*—Information retrieval, text summarisation, fuzzy logic, and fuzzy pattern rule induction,**

I. INTRODUCTION

1.1 Information Retrieval

Information retrieval is a field that focuses on finding information within documents, database storage, and even the world wide web. This paper specialises in information retrieval in text-based documents, such as microsoft word document and websites with textual content.

Information retrieval requires inter-disciplinary approach, including computer science, mathematics, information science, information architecture, and even statistics. This paper is concerned with automated information retrieval; that is using specific algorithms to extract information from a certain, predefined, type of document.

Consequently, automated information retrieval makes use of the principles of Natural Language Processing (NLP), a subfield of artificial intelligence that deals with communication between computers and humans using natural language, as opposed to harder-to-comprehend formal language or regular expression. However, complete natural language understanding is somewhat impossible to achieve and therefore classified as AI-complete problem, because it requires extensive knowledge about the outside world.

1.2. Performance and Correctness Measures of Information Retrieval

Because of the imprecise, uncertain nature of natural language processing itself, measuring the performance and correctness of information retrieval algorithm is not a trivial matter. Despite that, some different measures have been proposed to determine the correctness of information retrieval results. All these measures share something in common: relevancy forms the basis of every measurement. That is, every document is either marked relevant or irrelevant to a particular query. Two important measures will be discussed in this paper, precision and recall.

Precision is the fraction of the documents retrieved that are relevant to the user's information need. Below is the formula to calculate the precision of a particular document:

$$\text{precision} = \frac{|\{\text{relevant documents}\} \cap \{\text{retrieved documents}\}|}{|\{\text{retrieved documents}\}|}$$

The second measure is recall. Recall is the fraction of the documents that are relevant to the query that are successfully retrieved. Below is the formula to calculate the recall of a particular document:

$$\text{recall} = \frac{|\{\text{relevant documents}\} \cap \{\text{retrieved documents}\}|}{|\{\text{relevant documents}\}|}$$

However, it is important to note that recall alone is not enough. It is trivial to achieve recall of 100% by returning all documents in response to any query. Therefore, it is important to assure that the recall measure is used alongside other measures, such as precision. These two measures are the standards that will be used to evaluate the relevancy of information retrieval methods being used.

1.3. Fuzzy Logic

Fuzzy logic is a type of logic that deals with approximate reasoning, in contrast with conventional, more widely-used logic that is fixed and exact. Fuzzy logic uses either many-valued logic or probabilistic logic. While traditional logic uses either 0 or 1 (binary value) as its truth value, fuzzy logic variables may have a truth value between 0 and 1, with 0 representing completely false, and 1 representing completely true.

Fuzzy logic was proposed by Lotfi Zadeh, an Iranian-American computer scientist, in 1965.

Figure 1 - Lotfi Zadeh, one of the "founding fathers" of fuzzy logic"

The perplexing thing is that, while fuzzy logic was founded by Zadeh in the United States, its application is more widely-used in Asia, especially Japan. One of the reasons is the western culture that tends to regard things as black or white, true or false, while Asian culture is more receptive to the concept of "grey", a value between black and white.

In this topic, fuzzy logic is more useful than traditional logic, because fuzzy logic has the ability to handle natural language better. Consider the following example: a fuzzy variable, temperature, has four fuzzy sets, cold, cool, warm, and hot. As a comparison, for the same variable there are four crisp sets, namely temperature <273 Kelvin, between 273 and 323 Kelvin, between 323 dan 373 Kelvin, and above 373 Kelvin. From the two examples, we can safely conclude that the fuzzy approach is the more suitable one to use in information extraction, because it is more closely related with natural language; that is, a text is more likely to contain the word hot, warm, and cold, rather than mentioning the exact temperature.

For the reason above, fuzzy approach is the approach that will be explored further in this paper.

II. Fuzzy Pattern Rule Induction

The first method covered in this paper is fuzzy pattern rule induction. It is important to note that pattern matching is one of the most common methods to extract information from a text-based document. Fuzzy pattern, however, requires different methods than traditional pattern recognition.

One of the methods that is most commonly used is pattern rule induction. There have been some previous works in the field. Autoslog, for instance, uses a set of linguistic extraction patterns to build a dictionary for the terrorism domain. Another example, Crystal, begins with the most specific rule, and then proceeds with generalisation by merging similar rules.

From the example above, it is obvious that rules form the basis of fuzzy pattern rule induction. To obtain the rules, this method mainly uses Fuzzy pattern Rule Induction System (FRIS). Basically, FRIS is a rule induction algorithm that makes use of machine learning principles, requiring a training corpus and classified as supervised learning.

Like all learning algorithms, FRIS requires both training set and test set, both in the form of documents. First of All, FRIS requires a pre-processing of the documents, which will be explained in the next chapter.

2.1. Pre-processing of Training and Test Documents

Prior to the learning process, both the learning and testing documents will be pre-processed by the standard NLP modules, including tokenisation, shallow parsing, and sentence splitter. For experimental purpose, we can use NLProcessor (a shallow parser) to generate the part-of-speech. Consider the statement "A bomb was thrown near the house". A shallow parsing using NLProcessor would result in "[A_DT bomb_NN] <was_VBD thrown_VBN> near_IN [the_DT house_NN]", in which the "[ ]"s are noun phrases and "( )"s are verb phrases. DT, NN VBD, VBN and IN are the PoS tags for delimiter, noun, verb past, verb past participle and preposition respectively.

Every [] and () and the words between the bracket is the unit that will be used as the chunk in the FRIS process. The system also requires that the users **tag** the sentences containing particular information of specific slot type. For every slot type, the tagged instances are regarded as positive examples, while other sentences are regarded as negative examples. The tag and slot type are essential in the FRIS, because they form the basis for the FRIS algorithm, which will be explained in detail later in this paper.

2.2 The Context Feature Vector

It is important to note that the tagged instances is used as the training data, so that the algorithm will be able to predict correctly the tag of future, unknown instances (test data). For every tagged training instance, a context feature vector centred around the tagged slot is generated by the FRIS. The context feature vector forms the basis where the fuzzy pattern rule is generated from. Below is the general form of the vector:

$$<c_{-k}> ...<c_{-2}> <c_{-1}> <c_0> (tagged\_slot) <c_{+1}> <c_{+2}> ... <c_{+k}> \quad (1)$$

Here $<ci>$ $\{i=-k$ to $+k;\ i \neq 0\}$ represents the context units of the tagged slot, and k is the number of context units considered. $<c0>$ represents the central tagged slot itself. $<ci>$ can be a token, a noun or a verb phrase or even a syntactic unit such as subject or object and it can be of various feature types, including: words, PoS (if it's a single token), various types of verbs and noun chunks, and semantic classes[1].

For a single tagged instance, the above (1) formula can be transformed to the formula as follows:

$$<(-k,f_{-k}^{1}), ..., (-k,f_{-k}^{m}),..., (-1, f_{-1}^{1}), ..., (-1, f_{-1}^{m}), (0,f_0^{1}),..., (0,f_0^{m}), (1,f_1^{1}), ..., (1,f_1^{m}), ..., (k,f_k^{1}), ..., (k,f_k^{m})> \quad (2)$$

In the formula above, m is defined as the number of linguistic features of each element. As seen above, every element takes the form of a tuple (g,fg^i). The first element of the tuple (g in the example) represents the element's position within the tagged instance. $g=0$ gives the position

of tagged slot, and positive *g* (or negative *g*) gives the *gth* right (or left) hand context element from the tagged slot.

Meanwhile, the second element of the tuple, fg^i represents the possibilities of germane linguistic interpretation of each element. The set of possible linguistic interpretation is comprised of twelve syntactic, lexical, and semantic features described in the table below:

Feature	Description	Feature	Description
f_g^1	Lex. String	f_g^2	PoS
f_g^3	NP_Person	f_g^4	NP_Org.
f_g^5	NP_Loc	f_g^6	NP_Date
f_g^7	NP_Time	f_g^8	NP_Perc.
f_g^9	NP_Mon.	f_g^{10}	NP_Num.
f_g^{11}	VP_Pass.	f_g^{12}	VP_Act.

Table 1 - Linguistic Features used by FRIS

In the table above, The first two representations (Lex. String and PoS) respectively give the original lexical form, and the Part-of-Speech information of the element if it is a single token. On the other hand, the next 8 representations, (NP_Person, NP_Org., NP_Loc., NP_Date, NP_Time, NP_Perc., NP_Mon., and NP_Num.) cover the general named entities (NE) of type Person, Organization, Location, Date, Time, Percentage, Money and Number ("NP" stands for "Noun Phrase" and "VP" stands for "Verb Phrase"). The last 2 representations (VP_Act. and VP_Pass.) indicate the active and passive voice of VP.

2.3. Distribution of Training Examples

To make sure that the training examples are fully utilised, FRIS uses a global approach. That means that rules are not generalised from one single instance, but thoroughly includes every information in positive training examples and selects the most prominent feature to construct the rule.

The purpose of distributing the training example is to make sure that the FRIS algorithm fully represents the training set. Otherwise, there is a tendency that for the FRIS to over-generalise from the training set. The algorithm to distribute the training examples, however, is rather complex and is beyond the scope of this paper. It is sufficient for readers to note that proper distribution of training examples is very important to assure the accuracy of the FRIS algorithm.

2.4. Rule-Induction Algorithm

Now we have come to the most essential part: the rule-induction algorithm itself. This process proves to be crucial because correct rules mean correct patterns, and correct patterns result in correct information extraction. The overall algorithm is as follows:

a) Group tagged instances of the same slot type into one cluster.
b) Generate context feature vectors for all positive instances in every cluster. The
resulting *kth* cluster is *Ck*, with the positive instance set *Pk* and negative instance set *Nk*.
Let *rk* be the set of rules extracted so far to cover *Pk*; and set *rk* = *null*.
c) For every cluster *Ck*, perform the followings:

```
(c1) Loop-1: // to generate new rules
Let fc=null be the current feature set;
rc(fc) be the current rule; and
Pc, Nc be the set of instances covered by
rc(fc).
Initially, set: Pc = Pk, Nc = Nk
RuleAttempt = 0;
(c2) Loop-2: // to refine current rule
rc(fc)
Find top w element features {fg
i} (based on βgi egi values) that
covers at least
one instance in Pc;
Select the fi
j that minimizes the Laplacian measure of
the current rule rc(fc
∪ fi
j);
Add fi
j to fc, i.e. fc = fc ∪ fi
j
RuleAttempt++;
(c3) IF Laplacian(rc(fc)) < σ (error
tolerance)
THEN // the quality of resulting rule is
good
Add rule rc to rule set rk; or rk = rk ∪
rc;
Update Pk = Pk-{all instances covered by
rule rc};
Go to Loop-1 to generate another rule.
ELSE // more work is needed to constraint
rule rc
Update Pc by removing those instances that
are not covered by rc;
IF RuleAttempt ≥ λ (max. rule attempt for
constraining rules)
THEN // relaxing error tolerance;
Increase σ;
Go to Loop-1 to generate new rule with
bigger error tolerance;
ELSE
Go to Loop-2 to find new feature f'i
j to refine rule rc.
  Repeat until Pk is empty.
```

In short, first of all, the algorithm puts all tagged instances of the same slot type into the same cluster. After that, context feature vectors (as described on the previous section) is generated, resulting in both positive and negative instance set. Then, for every cluster, generate the rules. The next step is refining the rules, by selecting the function that minimises the Laplacian measure of the current rule (the smaller the Laplacian measure, the better). If the Laplacian value of the current rule is smaller than the error tolerance, then repeat the previous steps to obtain other rules. If not, either set a bigger error tolerance, or add another feature to refine the rule. The tendency is that the more features added to the rule, the lower its Laplacian value will be (and therefore, the more accurate). However, the approach has its own drawbacks, as the extra features might result in more complexity. In the end, every final rule must have its Laplacian value

below the error tolerance. The process is then repeated until all the positive instance set is empty.

2.5. Experimental Result

For testing purposes, some text-based documents, such as semi-strucutred web page corpora, and free text corpus was used as both training set and data set. In the experiments, the FRIS algorithm, as described above, is given a subset of a text as its training set, and the rest of the text as its data test. Both datasets are preprocessed using the predefined preprocessing algorithm. The number of texts used were 1,500 texts. Of all the texts, approximately 50% are relevant with their associated answer keys given in the MUC-4 corpus. The target slots are perpetrators, victims, and physical targets.

In the testing process, 100 texts are used, with breakdown as follows: 25 relevant texts and 25 irrelevant texts from the TST3 data test, and also 25 relevant texts and 25 irrelevant texts from the TST4 data test. The result is summarised below:

TST3	Rec.	Pre.	F_1	TST4	Rec.	Pre.	F_1
GE	58	54	56	GE	62	53	57
GE-CMU	48	55	51	GE-CMU	53	53	53
UMASS	45	56	50	SRI	44	51	47
FRIS	45	53	49	Alice	46	46	46
Alice	46	51	48	*FRIS*	45	47	46
SRI	43	54	48	NYU	46	46	46

Table 2 - Comparison among various information retrieval algorithms

The table above shows the comparison among various algorithms, FRIS obviously being one of them. Results are ranked from the most accurate to the least accurate algorithms. From the experiment result, we can see that FRIS is hardly the most accurate algorithm. GE and GE-CMU constantly rank as the two most accurate algorithms being compared. However, it is important to note that both GE and GE-CMU are not fully automated, requiring at least 10,5 person month to perform correctly,in contrast with FRIS which is fully automated and requires no manual effort at all.

Among the fully automated algorithms, FRIS ranks nearly on par with Alice, with FRIS having the upper hand in TST3 data test, and performing equally for the TST4 data test. **Among fully automated information retrieval algorithms, FRIS performs well.** The key to FRIS' success is that FRIS learns the rule from a whole set of training instances, at the lexical, syntactic, and semantic levels. In contrast, other algorithms learn from only one instance.

III. TEXT SUMMARISATION USING FUZZY LOGIC APPROACH

3.1. Text Summarisation

Automatic text summarisation is a process done by computers, with the goal of creating a shorter document, while preserving the content of the original document. However, automatic summarisation is very different from summarisation done by humans. The summarisation done by humans require deep and thorough understanding of natural language, something that computers are unlikely to do given the immense complexity of the matter.

Automatic text summarisation can be classified into two categories: based on abstraction and based on extraction. This paper covers automatic text summarisation based on extraction. Overall, text summarisation is a subfield and inseparable from information retrieval, since text summarisation requires a distinction between important and unimportant information.

3.2. Architecture of the System

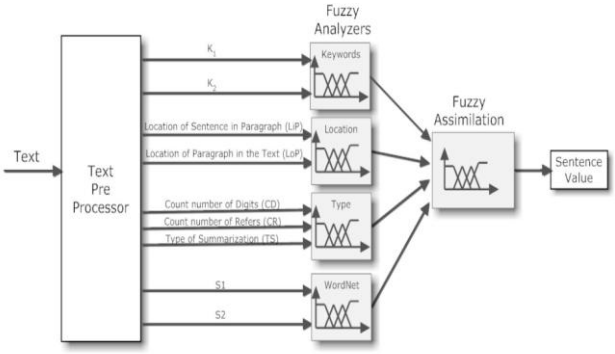

Figure 2 - The overall architecture of the proposed system

First of all, a text is pre-processed using a predefined algorithm. The pre-processing consists of separating the text into different parts that is later analysed using; each analyser is a fuzzy function and has its own distinct characteristic to be analysed. As shown in the picture above, the first fuzzy variable analyses the keywords of the text. The second fuzzy variable analyses the location of a particular sentence or paragraph in the text. This part is crucial because an opening sentence in a paragraph might carry more weight than senteces located in the middle of a paragraph, since there is more likelihood that the opening paragraph is also the topic sentence.

The third part deals with the numbers. Factors weighed in this fuzzy analyser include the number of digits, number of refers, and even type of summarisation, as different summarisation types yield different results. Last but not least, the WordNet. WordNet is an English language lexical database. It groups words that have similar meanings, into a particular cluster, and provide a simple definition for the word.

The last process is fuzzy assimilation. The results of fuzzy analysers are then assimilated in order to extract the most essential things from the text. Only the most essential information is included in the final summary. The result from the assimilation is then de-fuzzified, resulting in a particular sentence value. A higher sentence value means that the particular sentence contains essential information, and is more likely to be included in the final summary.

While some people argue that text summarisation should take the readers into consideration, such as whether the readers are experts or not, linguists and computer scientists alike have regarded this as nearly impossible to achieve. The common approach is to create a text summarisation that is generally clear enough for every reader, not only suited for a subset of readers.

3.3. The Algorithm

From the algorithm above, we can see that there are two steps involved, the first one is pre-processing, and the second is fuzzy analysis. A program was then developed, using MATLAB/SimuLink, that parses the sentence into the following components:
(1) The number of title words in the sentence,
(2) Whether it is the first sentence in the paragraph,
(3) Whether it is the last sentence in the paragraph,
(4) The number of words in the sentence,
(5) The number of thematic words and keyword synonyms in the sentence.

From the references, it is clear that summaries comprised of leading sentences have better performance than summaries using other methods. Therefore, it is of utmost importance that the algorithm identify which sentences are the most important ones. This can be done by taking the five factors above into consideration, such as considering the number of title words, the more title words there are in a sentence, the more probable that the sentence is an important one.

Additionally, the first and last sentence in a paragraph are more likely to be concluding statements, carrying more weight than other sentences in the paragraph. From experiments, it is also concluded that short sentences are less likely to appear in the summary. Once the five information needed above have been obtained from a sentence, those five characteristics then serve as the input for a fuzzy inference system. The fuzzy shape used is triangular, mainly for simplicity and efficiency reason.

3.4. Fuzzy Analyser

Due to the uncertain and often imprecise nature of natural language, risks, uncertainty, and ambiguity require flexibility beyond the scope of traditional, binary logic.

The main part of this system is the fuzzy analyser. The analyser processes every sentence in accordance with the five characteristics above, and then ranks every sentence according to its importance. For instance, a sentence that contains more keywords and more title words are likely to be ranked higher than other sentences.The summary includes only a handful of most important sentences according to the rank generated by the fuzzy analyser, preserving the overall content of the text with significantly less sentences.

The analyser uses Mamdani fuzzy inference system, which is more suitable in obtaining knowledge from a text. Using Mamdani, the summarisation result is also more likely to be more readable and human-friendly than the Sugeno method.

Nevertheless, the Mamdani method has a main weakness: it requires more computational resource than its simpler Sugeno counterpart. This problem can be rectified by using only the triangular-shaped functions, since they require considerably less computational resource than other, more complex-shaped functions.

Overall, the analyser follows six steps to compute the output:
(1) Determining a set of fuzzy rules,
(2) Fuzzifying the inputs using the input membership functions,
(3) Combining the fuzzified inputs according to the fuzzy rules to establish rule
strength,
(4) Combining the fuzzified inputs according to the fuzzy rules to establish rule
strength,
(5) Finding the consequence of the rule by combining the rule strength and the
output membership function,
(6) Defuzzifying the output distribution to get a crisp output

The fuzzy sets and variable for each of the five analyser criteria defined above are summarised by the following table:

The fuzzy analyzer	I/O	Variable name	Fuzzy term sets
Keyword	input	*K1*	Zero, Low, Medium, High
Keyword	input	*K2*	Zero, Low, Medium, High
Keyword	Output	*Ko*	Zero, Low, Medium, High and Very High
Location	input	*Location-in-Paragraph (LiP)*	First, Middle and End
Location	input	*Location-of-Paragraph (LoP)*	Top, Middle and Down
Location	Output	*Lo*	Zero, Low, Medium, High and Very High
Summary Type	Input	*Number-of-Digits (CD)*	L (Low), Medium (M) and High (H)
Summary Type	Input	*Number-of-Refers (CR)*	L (Low), Medium (M) and High (H)
Summary Type	Input	*Type-of-Summarization (TS)*	Statistical, Normal and Journal
Summary Type	Output	*To*	Zero, Low, Medium, High and Very High
Wordnet	Input	*S1*	Zero, Low, Medium, High

Table 3 - Functions, Variable names, and Fuzzy term sets for each fuzzy analyser criterion

To see the correlation between the analysers keyword and summary type and the overall result, we can see the 3D figure below:

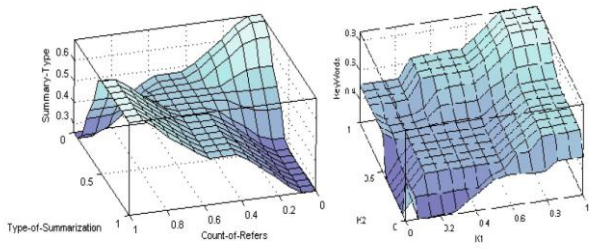

Figure 3 - The Effect of Type-of-Summarisation and Keywords Analysers on the Result

The figure above shows the simulation result of the fuzzy inference system using MATLAB. In addition, it is also important to note that MATLAB is the main tool used. All fuzzy sets and rules are put into MATLAB, and the inference system eventually generates the result.

3.5. Experimental Result

Evaluating the performance of a result summarisation is rather complex. Previously, it required human efforts to evaluate a computer-generated summary, with correctness, grammar, coherence, and other criteria as the measurement standards. That process, however, is rather expensive and even ostensibly simple summaries might require hundreds or even thousands of human hours to evaluate.

Fortunately, however, some methods to evaluate summaries automatically have been developed. The standard that we use in this paper is the Rouge evaluation. Details about the Rouge evaluation can be seen in detail in the reference.

The models were trained on English data, and eventually tested on Duc 2003 data. Some other automatic summarisers were also used in comparison. The result is shown in the figure below:

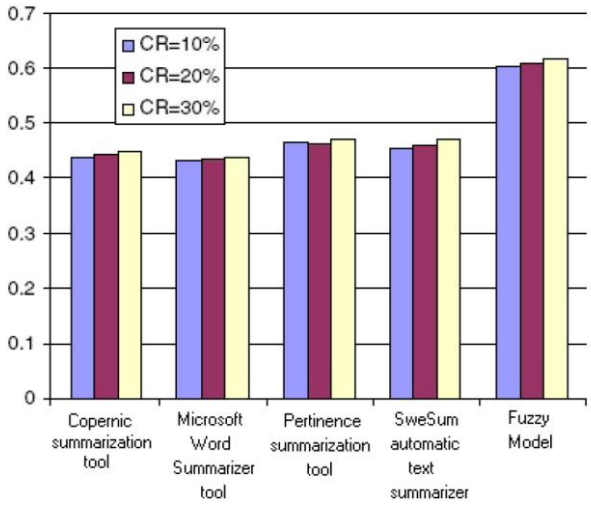

Figure 4 - Comparison between different automatic text summarisation methods

CR means the compression rate, and the Y-Axis represents the completeness of the summary using the Rouge evaluation. The chart above shows, obviously, that the higher the compression rate, the higher the accuracy.

The interesting thing, is that for every compression rate, **fuzzy model performs best compared with other methods**.

Fuzzy model yields higher results mainly because linguistic variables and and human perceptions are also taken into account. Another advantage is that fuzzy model is more suited to natural language that is more prone to errors and uncertainty. However, the downside is that designing the fuzzy rules is not a trivial task, and all relationships among the parameters have to be taken into account.

V. CONCLUSION

Information retrieval is the process of finding valuable information, mainly in text-based documents. Fuzzy approach can be used as one of the methods of information retrieval. The first point covered is using fuzzy pattern rule induction system (FRIS). To efficiently retrieve information from a text document, we need to define some patterns, or rules. The steps include pre-processing, distributing the training documents, and processing using the algorithm itself. Overall, FRIS compares favourably among fully automated information retrieval algorithms. The second point is using fuzzy analysis with fuzzy inference system (FIS) in MATLAB to summarise texts. There are five aspects of every sentence that is analysed, with each aspect having its own fuzzy sets. All the five analysers is eventually assimilated to produce a value. The value decides whether or not a sentence is put in the summary; only the most important sentences are put in the summary. This fuzzy model for text summarisation also compares very favourably among other text summarisation algorithms

VII. ACKNOWLEDGMENT

The author would like to thank Bapak Rinaldi Munir, as the lecturer of "Metode Numerik dan Fuzzy logic". It is due to his unwavering dedication to share knowledge that this paper is able to be finished. Also, the author would like to thank kak Dimas as the course assistant. Last but not least, the author thanks all family and friends for their supports.

REFERENCES

[1] Dipanjan Das, A. F. (2007). A Survey on Automatic Text Summarization.

[2] F. KYOOMARSI, H. K. (2010). EXTRACTION-BASED TEXT SUMMARIZATION USING FUZZY. *Iranian Journal of Fuzzy Systems, Vol. 7, No. 3* .

[3] Ladda Suanmali, N. S. (2008). AUTOMATIC TEXT SUMMARIZATION USING FEATURE-BASED FUZZY EXTRACTION. *Jurnal Teknologi Maklumat, Jilid 20, Bil. 2* .

[4] Ladda Suanmali, N. S. (2009). Fuzzy Logic Based Method for Improving Text. *(IJCSIS) International Journal of Computer Science and Information Security,* .

[5] Xiao, J. (n.d.). Fuzzy Pattern Rule Induction for Information Extraction.

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 15 May 2011

Adhiguna Surya
13509077