IJCCS (Indonesian Journal of Computing and Cybernetics Systems)
Vol.15, No.1, January 2021, pp. 65~74
ISSN (print): 1978-1520, ISSN (online): 2460-7258
DOI: https://doi.org/10.22146/ijccs.62716

Recommendation System for Thesis Topics Using Content-based Filtering

Hans Satria Kusuma*[1], Aina Musdholifah[2]
[1]Bachelor Program of Computer Science, FMIPA UGM, Yogyakarta, Indonesia
[2]Department of Computer Science and Electronics, FMIPA UGM, Yogyakarta, Indonesia
e-mail: ***[1]hans.satria.k@mail.ugm.ac.id**, [2]aina_m@ugm.ac.id

Abstrak

Dalam menempuh pendidikan jenjang S1, setiap mahasiswa diharuskan untuk mengejarkan skripsi agar dapat lulus dari program studi yang ditempuhnya. Akan tetapi dalam prosesnya mahasiswa memiliki beberapa kendala diantaranya kendala dalam menentuk topik skripsi mereka. Oleh karena itu, dibutuhkan suatu sistem rekomendasi yang dapat mengelompokkan topik skripsi berdasarkan minat dan kemampuan mahasiswa tersebut. Penelitian ini mengembangkan suatu sistem rekomendasi topik tugas akhir dengan menggunakan content-based filtering dimana nantinya mahasiswa diminta untuk memilih mata kuliah yang diminati beserta dengan nilainya. Setelah mendapatkan data yang dibutuhkan, maka sistem rekomendasi akan mengolah data tersebut lalu kemudian menampilkan judul dan abstrak publikasi yang paling sesuai dengan data yang dimasukkan.

Pada penelitian kali ini akan digunakan 2 buah dataset yaitu dataset publikasi dosen dalam kurun waktu 3 tahun terakhir dan silabus mata kuliah Ilmu Komputer UGM. Setelah menjalankan penelitian ini, didapatkan bahwa sistem rekomendasi memiliki running time rata - rata sebesar 7.46 detik. Didapatkan pula bahwa sistem ini mendapatkan persentase rata - rata pencapaian tujuan sistem rekomendasi yaitu relevance, novelty, serendipity dan increasing recommendation diversity sebesar 83%.

Kata kunci *— Sistem Rekomendasi, Content-based Filtering, Euclidean Distance*

Abstract

When pursuing their bachelor degree, every students are required to pursue a thesis in order to graduate from the major that they take. However, during the process, students got several difficulty regarding chosing their thesis topics. Therefore, a recommendation system is needed to classify thesis topics based on the students' interest and abilities. This study developed a recommendation system for thesis topics using content-based filtering where the students will be asked to choose the course that they interested in along with their grades. After getting all the required data, the recommendation system will process the data and then it'll show the title and the abstract of publication that fits the criteria.

In this research, there are 2 datasets that is used, there are lecturer publication within 3 years and syllabus data of Computer Science UGM course. After running this research, it was found that the recommendation system has an average 7.46 seconds running time. It was also found that the recommendation system got an average 83% of the recommendation system objectives. The recommendation system objectives consist of relevance, novelty, serendipity, and increasing recommendation diversity.

Keywords—*Recommendation System, Content-based Filtering, Euclidean Distance*

Received December 30th,2020; Revised January 29th, 2021; Accepted January 30th, 2021

1. INTRODUCTION

Thesis is a scientific essay that must be done by the students as a part of the final requirements of their academic study [1]. Therefore, thesis is the student's duty especially in this research is undergraduate students. Thesis should be done in order for the students to graduate from the major that they take.

Before getting hand onto the thesis, every student needs to find a topic that will be brought into their thesis topics. However there are still a lot of UGM students, specifically final year Computer Science UGM students, that's confused in finding their thesis topics based on their interests. One of the reason that makes the students confused is that there are so many thesis topics available, but they're not grouped based on the subject that they interested in and they find it hard to determine which subject that is needed to take the topics.

There are some research that have been conducted regarding the recommendation system for thesis topics. The approaches itself are vary between one researcher and another researcher. For example K-Means Clustering and Simple Additive Weighting by Daniati [2] is used in order to get the thesis topics for each student, but the topics recommendation that will be given will be based on the lab that has been predetermined. Another example of method that is used is Naïve Bayesian Classifier [3]. The research that has been conducted giving the thesis topics only based on the lab that has been predetermined. It won't give any fleixibilites for the students when they want to take more than 2 labs or more.

With the problems that have been mentioned before and by moving on from the previous research, we need a system so that can help students group the thesis topics based on the subjects of their interest. Therefore this research built a recommendation system to help students determine their thesis topics based on their interests and grades. However the main focus of this research will be only for computer science major in UGM.

The recommendation system that has been built is based on the courses that is interested in by the student and their grades upon the courses. After getting all of the grades that is needed, the system will compare between the publication's abstract and the courses' syllabus . Then the system will be displaying the list of recommendation publication title based on the smallest distance between those document. By having the course and it's grade, we hope that the system will group up the topics more broadly and will help the students to get the thesis topics based on their not only interest but also abilities.

2. METHODS

2.1 Text preprocessing

Text preprocessing is a series of process to clean the dataset (text) before the dataset being processed or being used for further preprocessing. Text preprocessing is the most important part in Natural Language Processing [4]. By having text preprocessing flow, it's hoped that it can get rid unnecessary words and maximize the data processing that will be carried out. The purpose of text preprocessing beside improving classification performance, it can also speed up time in data processing.

There will be 2 flow for the system that has been built. The first flow shown in figure 1.

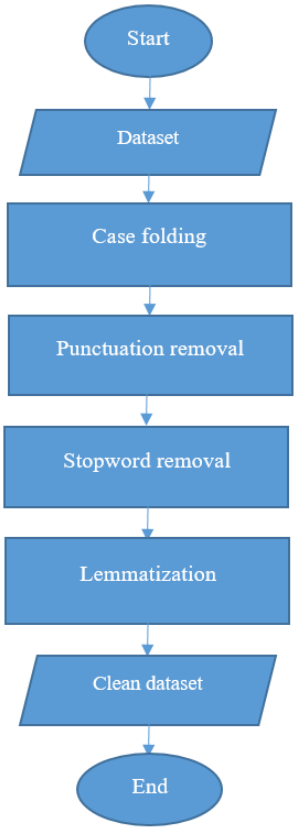

Figure 1 Text preprocessing flow

2.1.1 Case folding & punctuation removal

A text document must have contain various characters including letters, punctuation marks, numbers and symbols. Therefore, casefolding is needed to make uniform letters only (to lower case). For Punctuation removal, every characters other than letters will be removed and will be treated as delimiters.

2.1.2 Stopword removal

Stopword removal is needed during the text preprocessing. The definition of the stop word itself is defined as a set of words that are not related to the main subject, even though these words are often used in the text [5]. Stopword removal will decrease 20-30% of the total words inside a document [4].

2.1.3 Lemmatization

Lemmatization is a process in text preprocessing that determines the shape of a word and change it into a root word or finding the root of each word based on the context of the sentence [6]. The purpose of the lemmatization is to optimize the text mining process. Lemmatization has the same goal as stemming process – to get the root of each word. However,

the difference between the stemmind and lemmatization lies on how the process find the root of a word.

The stemming process looks for the root of a word by cutting the prefix and suffix of the word without paying attention to the context of the word that is used in a sentence. Meanwhile, lemmatization pay attention to the context and morphology of each word [7].

2. 2 Recommendation system

Recommendation system in general is that there are one / several users who provide recommendation as an input, then the input by the system will be collected and directed towards the best ouput according to the system [8]. However, in other cases, the value of a system lies in it's ability to provide the best recommendations to users who use the system. The flow of the system will be shown as in figure 2.

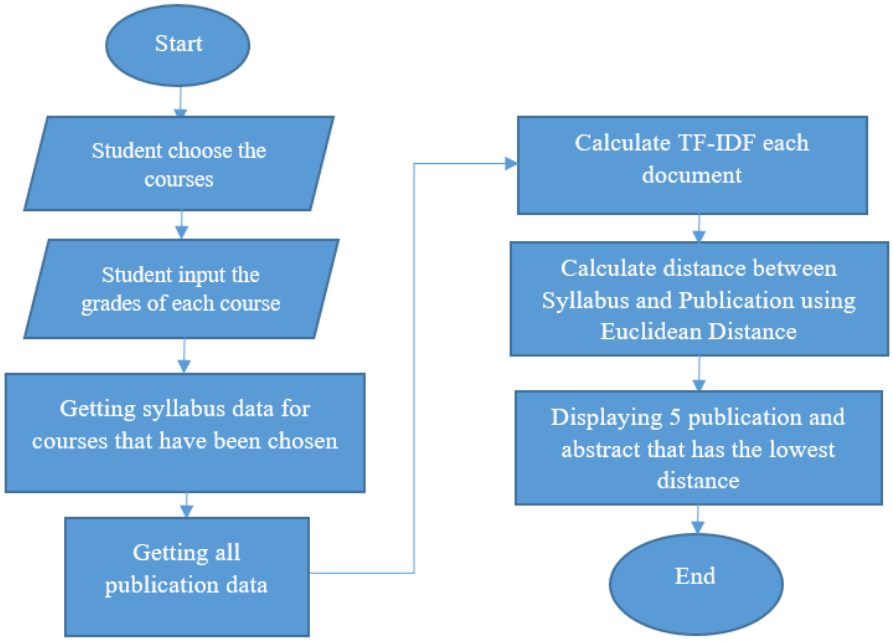

Figure 2 Recommendation system flow

2. 2. 1 Content-based Filtering

Content-based filtering is one of the method that is usually implemented to build a recommendation system. The output of this method really depends on the input from the user. Content-based filtering is one of the best method to build a recommendation system with text as their datasets [9].

2. 2. 2 Term Frequency – Inverse Document Frequency

Term frequency is the total frequency of a term appear in a document [10]. For getting the Term Frequency data, we also need to count the total number of term appear in a document as shown in equation (1). With Count(word,docs) is appearance of term *word* in document *docs* and sigma of Count(word,docs) means the total number of all term appear inside a document.

$$TF(word, docs) = \frac{Count(word, docs)}{\sum_{i=0}^{n} Count(word, docs)} \quad (1)$$

After getting the term frequency, we need to find the inverse document frequency of each term. Inverse document frequency is a value to check how often a term appear inside a corpus. If the term often appear inside a corpus, it'll be considered as a less important term. The formula to get the inverse document frequency [10] of a term will be shown in equation (2). With Count(docs) represent the total number of document and Count(words,docs) represent the total number of document that contain term *words*

$$IDF(word) = \log(\frac{Count(docs)}{Count(words, docs)} + 0.01) \quad (2)$$

After getting the term frequency and inverse document frequency, to get the value of TF-IDF just multiplying the TF and IDF of the words as shown in equation (3).

$$TF - IDF(word, docs) = TF(word, docs) \times IDF(word) \quad (3)$$

2. 2. 3 Euclidean Distance

For comparing the distance between 2 document, this research will use the euclidean distance. The data that will be calculated should be in vector format so that it can suits the formula on equation 4 [11].

$$Distance(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \quad (4)$$

After we got the distance between each document, we need to find the weighted distance (by using the course grade) by using the formula on equation 5. Where, Grade(Si) is the grade for course with syllabus Si and Distance(Aj,Si) is the euclidean distance between Abstract j and Syllabus i.

$$DistanceWeight(Aj, Si) = \frac{1}{Grade(Si)} * Distance(Aj, Si) \quad (5)$$

The last step after getting the DistanceWeight between each document, we need to find the average distance between each abstract and each syllabus. The formula to get average distance between each abstract and each syllabus can be seen on equation 6.

$$AverageDistance(Aj, S) = \frac{\sum_{i=0}^{n} DistanceWeight(Aj, Si)}{Count(Si)} \quad (6)$$

Where, AverageDistance(Aj,S) is the average distance between abstract j and syllabus, DistanceWeight(Aj,Si) is the weighted distance between abstract j and syllabus i, and Count(Si) represent the number of Syllabus that is taken by the students.

2. 3 Evaluation

There will be 2 evaluation that will be measured in this research, there are :

1. Running time : This evaluation will check the average time that is needed to run the system.
2. System objective : This evaluation will check relevance, novelty, serendipity and increasing recommendation diversity.

3. RESULTS AND DISCUSSION

3.1 Dataset

There will be 2 datasets that will be used there are computer science UGM course's syllabus and lecturer of computer science UGM publication within 3 years. The dataset for syllabus consists of 48 courses (have been filtered out for courses above 3rd semester only) and 160 publication including their abstract. For the syllabus datasets consist of No, Mata kuliah and Silabus. But after being cleaned in text preprocessing flow, there will be additional column pp_silabus. For the syllabus dataset the Mata Kuliah and pp_silabus column will be used to get the syllabus that have been chosen by the students. For the publication datasets consist of No, Judul and Abstract. But also after being cleaned in text preprocessing flow, there will be additional column pp_abstract. The pp_abstract column will be used to calculate the distance between publication and syllabus that have been chosen by students, and then after getting the smaller distance, the Judul and Abstract column will be used to show the output of the system.

3.2 Experimental Environment

The experiment is conducted in local computer and virtual environment (Google Colab). For the local computer specification that is used are Intel Quad Core @2.50 GHz, 8GB RAM and Nvidia GeForce 940 MX. For the virtual environment that is used are Google Colab Notebook with Virtual RAM 12.7 GB.

3.3 System Display

The recommendation system are implemented in web-based. The language that is used for the front-end is javascript (native), and for the back-end python with flask framework. The first step that will be seen by the students are checkbox of list courses as shown in figure 3.

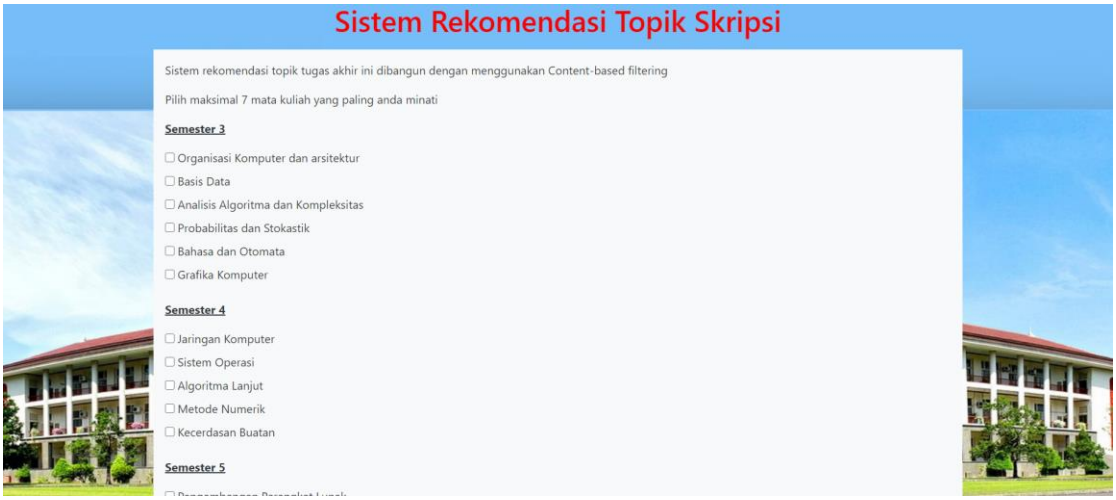

Figure 3 List of courses

After getting all of the courses that is chosen by the students, then there will be a modal to get the grade of each course as shown in figure 4.

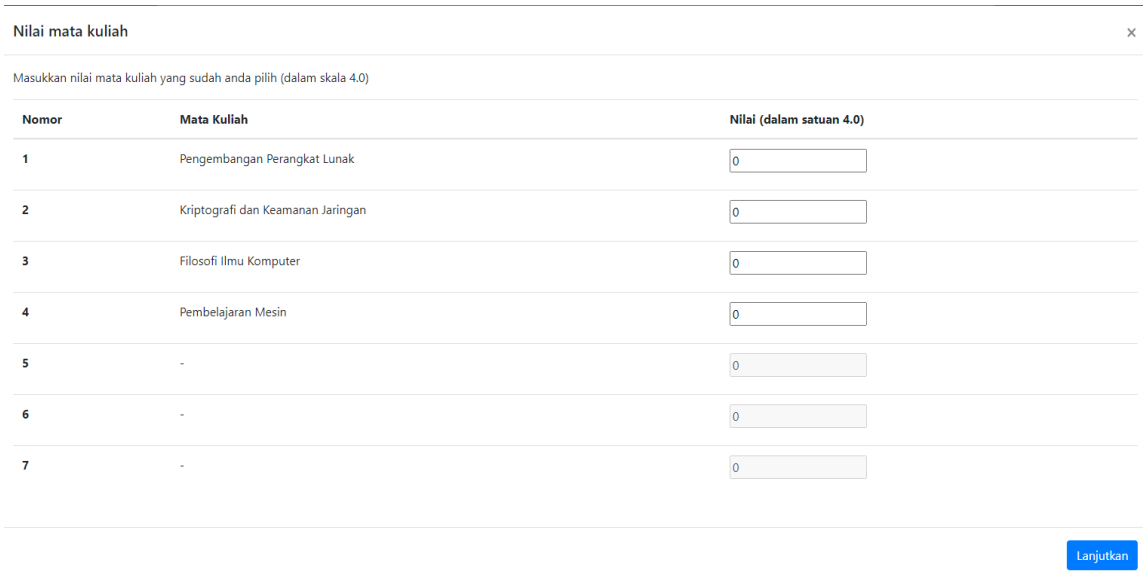

Figure 4 Input Grade

After getting all the data that is needed the system then will show the recommended title publication and it's abstract based on the user input. The display will be shown in modal shown in figure 5.

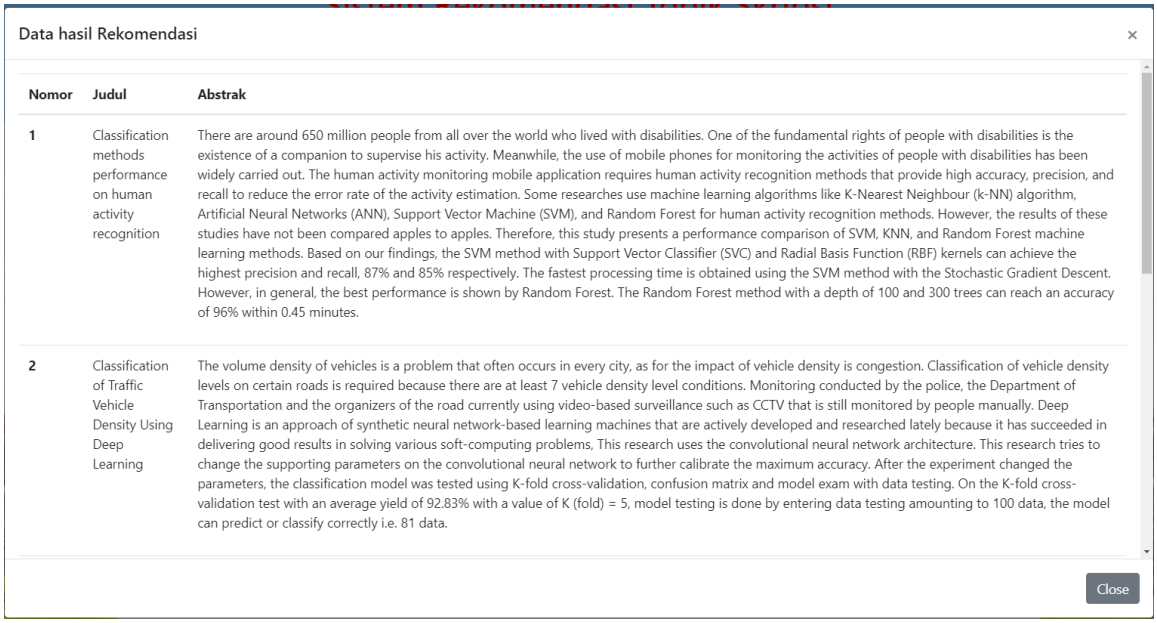

Figure 5 Recommendation display

3.4 System objective result

Evaluating objective of the system is done by interviewing 30 random computer science student. Before interviewing the student, they're asked to test the system by choosing the course that they like and it's grade.

There will be 4 questions regarding the system objective result. The question that will be given will be about the relevance, novelty, serendipity, and increasing recommendation diversity of the system. The result of the interview will be shown in the table below.

Table 1 Average running time system

Student	Question about			
	Relevance	Novelty	Serendipity	Diversity
1	4	2	1	3
2	4	4	5	3
3	3	4	5	4
4	4	5	3	4
5	5	4	3	4
6	5	4	5	4
7	3	4	4	5
8	4	4	3	4
9	5	3	5	5
10	5	5	5	5
11	1	5	1	5
12	4	5	4	5
13	4	5	5	5
14	4	5	4	4
15	4	5	5	4
16	4	4	4	5
17	4	5	4	5
18	4	5	4	5
19	4	5	4	3
20	2	5	3	5
21	5	4	4	3
22	3	5	4	3
23	4	5	5	4
24	4	5	4	4
25	4	5	5	5
26	3	5	4	4
27	4	5	4	4
28	4	4	4	4
29	4	5	4	3
30	4	5	5	5
Total Score	116	136	120	126
Time average	7.26	150	150	150
Percentage	77.33%	90.67%	80%	84%

3.5 Running time evaluation

The average running time that is needed to operate the system is 7.26 seconds. The average result is obtained from adding all the time that is needed for each student and then divide the result by 30 (total students)

Table 1 Average running time system

Student	Processing Time
1	4.88
2	7.62
3	9.62
4	7.16
5	9.25
6	6
7	11.40
8	10.08
9	8.06
10	7.44
11	5.52
12	8.79
13	5.65
14	5.94
15	6.99
16	6.67
17	7.75
18	6.84
19	8.27
20	5.28
21	9.17
22	9.35
23	5.96
24	6.8
25	6.23
26	4.95
27	9.49
28	5.2
29	5.67
30	5.79
Total Time	217.82
Time average	7.26

4. CONCLUSIONS

As can be seen in the result and discussion, the system objective have been achieved with a good score with relevance 77.33%, novelty 90.67%, serendipity 80% and increasing recommendation diversity 84%. The system has an average 83% of the system objective. For the running time that is needed by the system is 7.26 seconds average so it's pretty fast to get the recommendation from the system.

REFERENCES

[1] E. Setiawan, "Kamus Besar Bahasa Indonesia (Online)," 2020. [Online]. Available: http://kbbi.web.id/skripsi . [Accessed: 20-Apr-2020].

[2] T. Suwartiningsih, B. S. Rintyarna, and D. Arifianto, "Rekomendasi Topik Tugas Akhir Mahasiswa Teknik Informatika Universitas Muhammadiyah Jember Menggunakan Metode Naïve Bayesian Classifier," Universitas Muhammadiyah Jember, 2016. Available: http://digilib.unmuhjember.ac.id/download.php?id=1915. [Accessed: 12-Mar-2020]

[3] E. Daniati, "Decision support system to deciding thesis topic," *Proc. - 2017 Int. Semin. Appl. Technol. Inf. Commun. Empower. Technol. a Better Hum. Life, iSemantic 2017*, vol. 2018-Janua, pp. 52–57, 2017.

[4] S. Kannan *et al.*, "Preprocessing Techniques for Text Mining," *Int. J. Comput. Sci. Commun. Networks*, vol. 5, no. 1, pp. 7–16, 2015.

[5] A. Setiawan, E. Kurniawan, and W. Handiwidjojo, "Implementasi Stop Word Removal Untuk Pembangunan Aplikasi Alkitab Berbasis Windows 8," *J. EKSIS*, vol. 6, no. 2, pp. 1–11, 2013.

[6] J. Kanerva, F. Ginter, and T. Salakoski, "Universal Lemmatizer: A sequence-to-sequence model for lemmatizing Universal Dependencies treebanks," *Nat. Lang. Eng.*, 2020.

[7] D. Worth, "Introduction to Modern Information Retrieval, 3rd Edition," *Aust. Acad. Res. Libr.*, vol. 41, no. 4, pp. 305–306, 2010.

[8] Resnick, P. and Varian, H. R., 1997. Recommender systems. *Commun. ACM*, 40(3), pp. 56-58 [online]. Available : https://dl.acm.org/doi/10.1145/245108.245121.

[9] J. Chen, C. Chen, and Y. Liang, "Optimized TF-IDF Algorithm with the Adaptive Weight of Position of Word," vol. 133, pp. 114–117, 2016.

[10] M. D'Agostino and V. Dardanoni, "What's so special about Euclidean distance? A characterization with applications to mobility and spatial voting," *Soc. Choice Welfare*, vol. 33, no. 2, pp. 211–233, 2009.