IMPLEMENTASI ALGORITMA DIJSKTRA UNTUK PERINGKASAN OTOMATIS ARTIKEL ILMIAH BERBAHASA INGGRIS

Yanuar Budi Prasetyo[1]
ynr.evt92@gmail.com

Gloria Virginia[2]
virginia@ukdw.ac.id

Antonius Rachmat C.[3]
anton@ti.ukdw.ac.id

Abstract

This research implemented Dijkstra algorithm to summarize text documents. To run Dijkstra algorithm, we represented sentence as a vertex and the similarity between sentences as an edge, where the result of the summarization process was the shortest path of Dijkstra calculation. The summarization processes were conducted by utilizing WordNet database and without WordNet database. Our experiment showed that without WordNet the result was better than using WordNet. Using WordNet, the system could obtain 80% relevant sentences. While without WordNet, the system could obtain about 90% of relevant sentences.

Keywords : *Dijkstra Algorithm, WordNet, Text Document, Summarization, Shortest Path.*

1. Pendahuluan

Pada masa sekarang ini, sangatlah mudah untuk mendapatkan informasi, baik melalui media cetak maupun media elektronik. Akan tetapi, banyaknya informasi yang ada belum tentu bersifat spesifik seperti yang dibutuhkan. Keterbatasan waktu untuk membaca semua informasi yang ada juga menjadi salah satu kendala untuk memperoleh informasi utama yang diperlukan.

Melihat masalah di atas, penulis membangun sebuah aplikasi berbasis algoritma *Dijkstra* untuk menemukan intisari dari dokumen teks. Peringkasan dokumen teks ini diawali dengan memodelkan dokumen teks menjadi sebuah *graph* (*Graph Theoritic Approach*). Dimana setiap kalimat pada dokumen teks direpresentasikan sebagai *vertex* dan hubungan antara dua kalimat direpresentasikan sebagai *edge.* Setelah memodelkan dokumen teks sebagai *graph*, algoritma *Dijkstra* digunakan untuk memperoleh jalur terpendek dari setiap kalimat yang ada. Hasil ringkasan merupakan semua kalimat yang dilalui oleh jalur terpendek yang diperoleh dari algoritma *Dijkstra.* Penggunaan *WordNet* juga diimplementasikan dengan harapan dapat meningkatkan kualitas peringkasan teks.

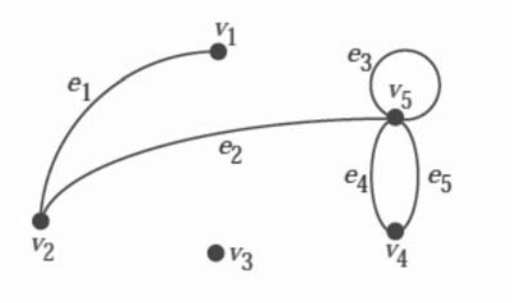
Sistem yang akan dibangun merupakan aplikasi berbasis desktop, dan menggunakan dokumen berupa *plain text (ASCII)* berbahasa inggris. Untuk mengetahui performa sistem, sistem ini akan dievaluasi menggunakan perhitungan *f-measure* dengan relevan dokumen yang merupakan abstrak asli dari setiap artikel.

[1]*Program Studi Teknik Informatika, Fakultas Teknologi Informasi, Universitas Kristen Duta Wacana.*
[2]*Program Studi Teknik Informatika, Fakultas Teknologi Informasi, Universitas Kristen Duta Wacana.*
[3]*Program Studi Teknik Informatika, Fakultas Teknologi Informasi, Universitas Kristen Duta Wacana.*

2. Landasan Teori

2.1 Graph

Secara umum, *graph* adalah kumpulan pasangan *(V,E),* dimana *V* adalah kumpulan *vertex* dan *E* adalah kumpulan *edge* yang menghubungkan *vertex.* (Ruohonen, 2013). Representasi *graph* dapat dilihat pada Gambar 1, dimana *v* adalah *vertex* dan *e* adalah *edge* yang menghubungkan dua *vertex.*



Gambar 1. Contoh Representasi Graph

2.2 Dijkstra Algorithm

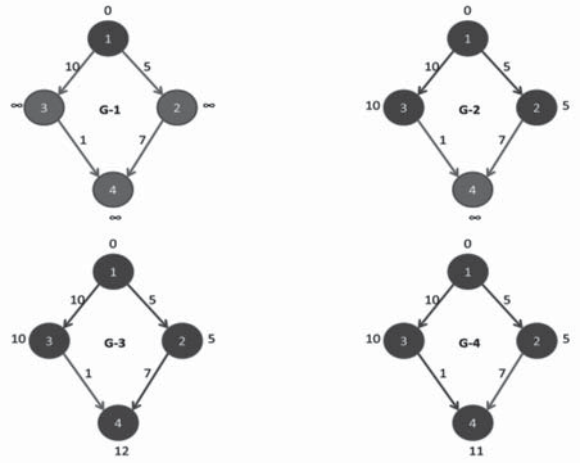
Dijkstra Algorithm dapat menyelesaikan masalah pencarian rute terpendek dengan cepat. . *Dijkstra Algorithm* akan menemukan rute dari *vertex u* ke *vertex z* lainnya secara urut pada *d(u,z)*. Jarak *d(u,z)* dalam *weighted graph* adalah jumlah minimum semua *edge* pada rute dari vertex *u* ke *vertex z.* (West, 2001)

Tahapan untuk melakukan *Dijkstra algorithm* adalah sebagai berikut :

1. Input
   Sebuah *graph* dengan semua bobot *edge* postif dan dimulai dari *vertex u.* Bobot dari *edge xy* adalah *w(x,y)*
2. Inisialisasi
   S adalah kumpulan *vertex* dimulai dari *vertex u*, *t(z)* adalah jarak sementara dari *u* ke setiap $z \notin S$. $S = \{u\}$; $t(u) = 0$; $t(z) = w(uz)$ untuk $z \neq u$
3. Perulangan

Ambil *vertex v* diluar *S* sehingga $t(v) = min_{z \notin S} t(z)$. Tambahkan v ke *S. Relax edges* dari *v* untuk memperbarui data jarak sementara: untuk setiap *edge vz*, *update t(z)* menjadi $\min\{t(z), t(v) + w(vz)\}$.

Perulangan akan terus berjalan hingga $S = V(G)$ atau ketika $t(z) = \infty$ untuk semua $z \notin S$. Setelah perulangan selesai, maka $d(u,v) = t(v)$ untuk semua *v.* Perulangan algoritma *Dijkstra* dapat dilihat pada gambar berikut :



Gambar 2. Simulasi Algoritma Dijkstra.

G-1. Inisialisasi $S = \{1\}, t(1) = 0, t(2) = \infty, t(3) = \infty, t(4) = \infty$.
G-2. *Relax edge vertex* 1 dan menghasilkan $t(2) = min\{t(2), t(1) + w(1,2)\} = 5$, $t(3) = min\{t(3), t(1) + w(1,3)\} = 10$
G-3. *Relax edge vertex 2* dan menghasilkan $t(4) = min\{t(4), t(2) + w(2,4)\} = 12$
G-4. *Relax edge vertex 3* dan menghasilkan $t(4) = min\{t(4), t(3) + w(3,4)\} = 11$

Pada Gambar 2, algoritma *Dijkstra* dari *vertex 1* ke semua *vertex.* Setelah algoritma *Dijkstra* selesai maka jalur terpendek dari *vertex 1* ke semua *vertex* adalah $d(1,2) = 5$, $d(1,3) = 10$, dan $d(1,4) = 11$ dimana $d(x, y)$ adalah jalur terpendek dari *vertex x* ke *vertex y.*

2.3 TF - ISF

Tf-isf (term frequency – inverse sentences frequency) merupakan adaptasi dari *tf-idf. Tf-isf* memiliki konsep yang mirip dengan *tf-idf, j*ika *tf-idf* digunakan untuk pembobotan pada dokumen, maka *tf-isf* digunakan melakukan pembobotan pada kalimat. Formula untuk menghitung *isf* dapat dilihat pada formula [1] :

$$\boldsymbol{isf_{(t)}} = \log\left(\frac{N}{\boldsymbol{sf_{(t)}}}\right) \quad [1]$$

Dimana *N* adalah jumlah kalimat dalam dokumen dan $\boldsymbol{sf_{(t)}}$ adalah jumlah kalimat dimana kata $\boldsymbol{t}$ muncul.

Nilai *isf* digunakan dalam perhitungan nilai *tf-isf* seperti tampak pada formula [2] :

$$\boldsymbol{tf - isf_{(t,s)}} = \boldsymbol{tf_{(t,s)}} * \boldsymbol{isf_{(t)}} \quad [2]$$

Dimana $\boldsymbol{tf_{(t,s)}}$adalah banyaknya kata $\boldsymbol{t}$ yang muncul dalam kalimat s. Pembototan *tf-isf* digunakan untuk memberi nilai setiap kata *t* pada dokumen *d*.

2.4 F-Measure

F-Measure adalah *harmonic mean* dari *precision* dan *recall. Precision* merupakan proporsi dari dokumen yang didapat oleh sistem dengan dokumen yang dianggap relevan. Sedangkan *recall* merupakan proporsi dari dokumen yang dianggap relevan dengan dokumen yang didapat oleh sistem. (Manning et al., 2009). Sehingga *precision* dan *recall* dapat dirumuskan sebagai berikut :

$$\boldsymbol{Precision} = \frac{\#(\text{relevant items retrieved})}{\#(\text{retrieved items})} = \boldsymbol{P}(\text{relevant}|\text{retrieved}) \quad [3]$$

$$\boldsymbol{Recall} = \frac{\#(\text{relevant items retrieved})}{\#(\text{relevant items})} = \boldsymbol{P}(\text{retrieved}|\text{relevant}) \quad [4]$$

Rumus diatas dapat diperjelas dengan gambar berikut:

	Relevant	Nonrelevant
Retrieved	true positives (tp)	false positives (fp)
Not retrieved	false negatives (fn)	true negatives (tn)

Gambar 3. Tabel Kontingensi.

(Manning et al. (2009) An Introduction to Information Retrieval)

Sehingga rumus *precision* dan *recall* dapat disimpulkan seperti dibawah ini:

$$P = \frac{\text{tp}}{(\text{tp}+\text{fp})} \quad [5]$$

$$R = \frac{\text{tp}}{(\text{tp}+\text{fn})} \quad [6]$$

Nilai *precision* dan *recall* dapat digunakan untuk menghitung nilai *f-measure* dengan rumus:

$$F = \frac{1}{\alpha\frac{1}{\text{p}} + (1-\alpha)\frac{1}{\text{R}}} = \frac{(\beta^2+1)PR}{\beta^2 P+R} \quad [7]$$

Dimana $\beta^2 = \frac{1-\alpha}{\alpha}$, α merupakan himpunan dari 0 sampai 1 dan β^2 merupakan himpunan dari 0 sampai tak terhingga.

Balanced f-measure memiliki α = ½ atau β = 1. Jika menggunakan *balanced f-measure* maka formula [7] akan menjadi formula [8]:

$$F_{\beta=1} = \frac{2PR}{\text{P}+\text{R}} \quad [8]$$

Meskipun demikian, cara di atas bukan hanya satu-satunya cara untuk menggunakan *f-measure*. Nilai dari β < 1 lebih menekankan *precision,* sedangkan β > 1 menekankan *recall. Precision, recall,* dan *f-measure* akan bernilai antara 0 dan 1.

3. Hasil dan Pembahasan

3.1 Contoh Perhitungan Manual Sistem

Untuk contoh perhitungan manual, input teks yang digunakan adalah sebagai berikut:

"Most students like the freedom they have in college. Usually college students live on their own, in the dormitory or in an apartment. This means they are free to come and go as they like. Their parents can't tell them when to get up, when to go to school, and when to come home. It also means that they are free to wear what they want. There are no parents to comment about their hair styles or their dirty jeans. Finally, they are free to listen to their favorite music without interference from parents"

Teks input terdiri dari tujuh kalimat yaitu :

S1. Most students like the freedom they have in college.
S2. Usually college students live on their own, in the dormitory or in an apartment.
S3. This means they are free to come and go as they like.
S4. Their parents can't tell them when to get up, when to go to school, and when to come home.
S5. It also means that they are free to wear what they want.
S6. There are no parents to comment about their hair styles or their dirty jeans.
S7. Finally, they are free to listen to their favorite music without interference from parents.

Bobot setiap kalimat dapat dihitung dengan cara sebagai berikut :

S1 :

TF-ISF(most, S1) = TF(most,S1) * ISF(most) = 1 * 0.84509 = 0.84509

ISF(most) = $\log(\frac{|S|}{SF(most)}) = \log\left(\frac{7}{1}\right)$ = 0.84509

TF-ISF(students,S1) = 0.54406

TF-ISF(like,S1) = 0.54406

TF-ISF(freedom,S1) = 0.84509

TF-ISF(college,S1) = 0.54406

Bobot Kalimat (AVG-TF-ISF(S1)) = $\frac{\sum_{i=1}^{5} TS-ISF(w_i,S1)}{5} = \frac{4.16745}{5} =$ **0.83349**

Perhitungan bobot dengan cara yang sama dilakukan untuk kalimat S2 sampai S7, dimana hasilnya tampak pada Tabel 1:

Tabel 1.
Tabel Bobot Kalimat S1-S7

Kalimat	Bobot Kalimat
S1	0.83349
S2	0.74474
S3	0.50753
S4	0.73331
S5	0.65805
S6	0.77057
S7	0.71734

Setelah bobot setiap kalimat dihitung, bobot setiap kalimat dapat digunakan untuk menghitung nilai kemiripan antar kalimat dengan cara sebagai berikut:

$$Cost_{1,2} = \frac{(1-2)^2}{overlap_{1,2}\ x\ weight_2} = \frac{1}{4\ x\ 0.74474} = 0.35568$$

$$Cost_{1,3} = \frac{(1-3)^2}{overlap_{1,3}\ x\ weight_3} = \frac{4}{2\ x\ 0.50753} = 3.94065$$

$$Cost_{1,4} = \frac{(1-4)^2}{overlap_{1,4}\ x\ weight_4} = \frac{9}{0\ x\ 0.73331} = \infty$$

$$Cost_{1,5} = \frac{(1-5)^2}{overlap_{1,5}\ x\ weight_5} = \frac{16}{1\ x\ 0.65805} = 24.31426$$

$$Cost_{1,6} = \frac{(1-6)^2}{overlap_{1,6}\ x\ weight_6} = \frac{25}{0\ x\ 0.77057} = \infty$$

$$Cost_{1,7} = \frac{(1-7)^2}{overlap_{1,7}\ x\ weight_7} = \frac{36}{1\ x\ 0.71734} = 50.18540$$

Dimana $Cost_{1,2}$ adalah nilai kemiripan antara kalimat ke-1 dan ke-2.

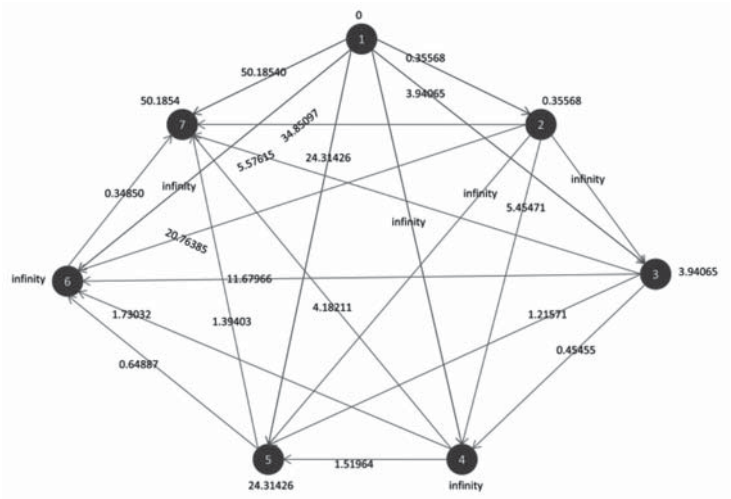
Perhitungan diatas dilakukan hingga mendapatkan nilai $Cost_{6,7}$. Setelah semua perhitungan nilai kemiripan selesai, maka akan diperoleh Tabel 2.

Tabel 2.
Tabel Nilai Kemiripan Antar Kalimat

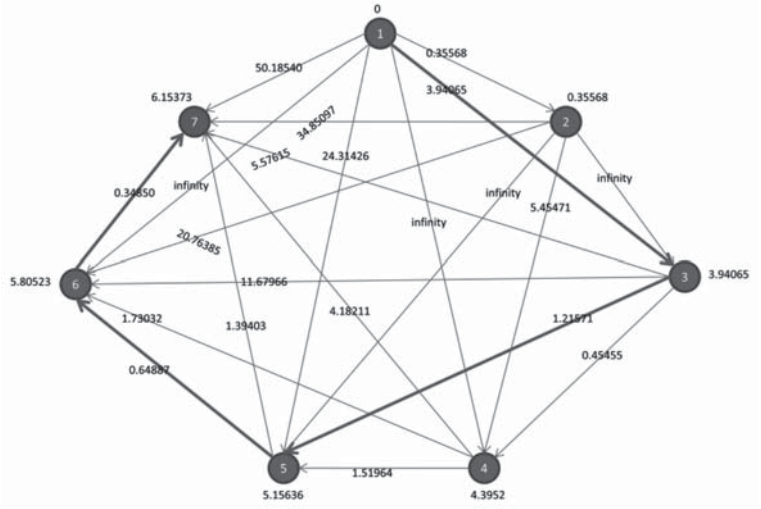
	1	2	3	4	5	6	7
1	X	0.35568	3.94065	∞	24.31426	∞	50.18540
2	X	X	∞	5.45471	∞	20.76385	34.85097
3	X	X	X	0.45455	1.21571	11.67966	5.57615
4	X	X	X	X	1.51964	1.73032	4.18211
5	X	X	X	X	X	0.64887	1.39403
6	X	X	X	X	X	X	0.34850
7	X	X	X	X	X	X	X

Baris 1 kolom 2 menunjukan nilai kemiripan antara kalimat pertama dan kalimat kedua. Tanda X pada tabel menunjukan bahwa tidak ada hubungan antar dua kalimat.

Setelah nilai kemiripan diperoleh, maka algoritma *Dijkstra* dapat dijalankan. *Graph* awal dapat dilihat pada Gambar 4, dan *graph* hasil algoritma *Dijkstra* dapat dilihat pada Gambar 5:



Gambar 4. Graph awal



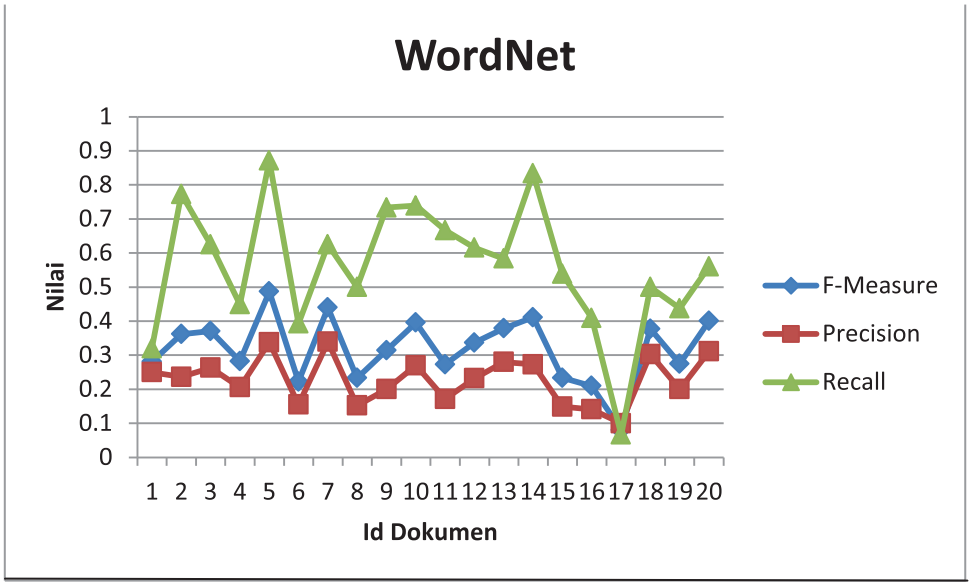
Gambar 5 . Graph hasil menggunakan algoritma *Dijkstra*.

Jalur terpendek yang dihasilkan adalah 1 → 3 → 5 → 6 → 7 dengan total jarak 6.15373. Sehingga kalimat yang dipilih oleh algoritma *Dijkstra* sebagai hasil ringkasan adalah kalimat ke-1, kalimat ke-3, kalimat ke-5, kalimat ke-6, dan kalimat ke-7 secara berurutan. Dengan demikian hasil peringkasan teks dari contoh diatas adalah sebagai berikut:

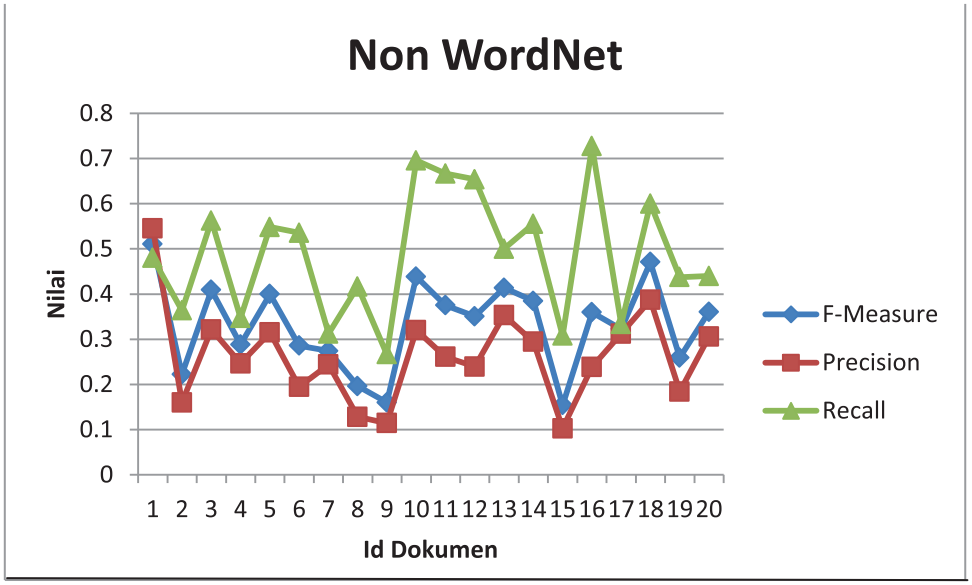
"Most students like the freedom they have in college. This means they are free to come and go as they like. It also means that they are free to wear what they want. There are no parents to comment about their hair styles or their dirty jeans. Finally, they are free to listen to their favorite music without interference from parents."

3.2 Pengujian Sistem

Penguijan dilakukan dengan menggunakan 20 dokumen *plain text* dengan jumlah total kalimat sebanyak 1750 kalimat. Pengujian dilakukan untuk proses menggunakan *WordNet* dan tanpa menggunakan *WordNet*. Hasil perhitungan *F-Measure*, *Precision* dan *Recall* dapat dilihat pada Gambar 6 dan Gambar 7 :



Gambar 6. Grafik Nilai Performa Sistem dengan WordNet.



Gambar 7. Grafik Nilai Performa Sistem Non-*WordNet*

Tabel 3.
Tabel Perbandingan Nilai Rata-Rata *F-Measure, Recall, dan Precision*

	WordNet	Non WordNet
Mean Average F-Measure	0.317953	0.331726
Mean Average Precision	0.228385	0.263364
Mean Average Recall	0.561919	0.487523

Berdasarkan Tabel 3, dapat dilihat bahwa performa sistem menggunakan *WordNet* dan tidak menggunakan *WordNet* memiliki nilai rata-rata *f-measure, precision, dan recall* yang tidak jauh berbeda.

Untuk evaluasi secara manual, hasil ringkasan teks akan dibandingkan dengan abstrak asli dari setiap dokumen uji. Tabel 4 menunjukan hasil evaluasi yang dilakukan secara manual pada hasil ringkasan menggunakan *wordnet* dan tidak menggunakan

wordnet. Kolom *relevan* akan bernilai 1 jika hasil ringkasan dianggap relevan dan 0 jika tidak.

Tabel 4.
Tabel Hasil Evaluasi Secara Manual

Id Dokumen	Jumlah Kalimat Dalam Dokumen	WordNet		Non WordNet	
		Relevansi	Jumlah Kalimat yang Relevan	Relevansi	Jumlah Kalimat yang Relevan
1	81	Ya	32	Ya	22
2	91	Ya	72	Ya	50
3	57	Ya	38	Ya	28
4	168	Tidak	107	Ya	69
5	104	Tidak	80	Tidak	54
6	132	Ya	71	Ya	77
7	105	Ya	59	Tidak	41
8	123	Ya	79	Ya	78
9	78	Ya	59	Ya	36
10	83	Ya	63	Ya	50
11	46	Ya	35	Ya	23
12	125	Ya	69	Ya	71
13	45	Ya	25	Ya	17
14	66	Ya	55	Ya	34
15	74	Tidak	47	Ya	39
16	129	Ya	64	Ya	67
17	34	Ya	10	Ya	16
18	72	Ya	33	Ya	31
19	69	Ya	35	Ya	38
20	68	Tidak	45	Ya	36

Tabel 4 menunjukan bahwa menggunakan *database WordNet* pada peringkasan teks dengan algoritma *Dijkstra* tidak selalu relevan. Dari 20 dokumen uji, proses peringkasan menggunakan *WordNet* menghasilkan 16 dokumen yang relevan, sedangkan proses peringkasan tanpa menggunakan *WordNet* adalah 18 dokumen yang relevan. Hal ini menunjukan bahwa proses peringkasan menggunakan *database WordNet* tidak menjamin hasil ringkasan yang relevan, bahkan penggunaanya menurunkan kualitas performa sistem. Hal ini mungkin dapat disebabkan oleh dua hal yaitu implementasi penggunaan *WordNet* yang kurang tepat atau penggunaan formula perhitungan nilai kemiripan yang tidak sesuai.

4. Kesimpulan dan Saran

Berdasarkan analisis dan implementasi sistem, maka diperoleh kesimpulan sebagai berikut:

1. Dalam analisis secara manual, dapat disimpulkan bahwa sistem peringkasan teks tanpa menggunakan *database WordNet* menghasilkan ringkasan yang lebih baik. Pada peringkasan tanpa menggunakan *database WordNet,* dari 20 dokumen uji, 18 diantaranya menghasilkan hasil yang relevan dengan abstrak. Sedangkan dengan menggunakan *database WordNet,* hanya 16 dokumen yang relevan dengan abstrak.
2. Dalam analisis performa sistem, peringkasan dengan menggunakan *WordNet* memiliki nilai rata-rata *f-measure, precision,* dan *recall* yaitu 0.317953, 0.228385,

0.561919. Sedangkan peringkasan teks tanpa menggunakan *database WordNet* memiliki nilai rata-rata *f-measure, precision,* dan *recall* sebesar 0.331726, 0.263364, 0.487523. Dari data tersebut, dapat disimpulkan performa sistem lebih baik saat tidak menggunakan *database WordNet.*

Saran untuk pengembangan dan perbaikan sistem adalah:

1. Mengoptimalkan cara mengambil kata-kata yang berhubungan dalam *database wordnet.* Pada penelitian ini, dilakukan pre-proses untuk mengambil kata yang berhubungan dan memakan waktu kurang lebih 36 jam.
2. Dalam penelitian ini, untuk mendapatkan hubungan antar kalimat hanya memanfaatkan 3 dari 18 tabel yang terdapat pada *database WordNet* yaitu tabel *words, synsets,* dan *senses.* Pada penelitian selanjutnya, perlu dicoba untuk memanfaatkan lebih banyak tabel *WordNet* seperti tabel *semlink, lexlinks,* dan lainnya.
3. Menggunakan formula yang lebih optimal untuk memanfaatkan *database wordnet.*
4. Dapat mencoba melakukan *stemming* dan *lemmatization* terlebih dahulu sebelum dilakukan peringkasan teks.

Daftar Pustaka

Budhi Gregorius S.,Intan Rolly , Silvia R.,Stevanus R. R. (2007). Indonesian Automated Text Summarization. *Proceeding ICSIIT* .

Halim Steven, Halim Felix. (2013). *Competitive Programming 3.* Singapore.

Manning Christopher D., Raghavan Prabhakar, Schutza Hinrich . (2009). *An Introduction to Information Retrieval.*

Ruohonen, K. (2013). *Graph Theory.*

Sankar K, Sobha L. (2009). An Approach to Text Summarization. *Association for Computational Linguistics* , 53-60.

Sjobergh Jonas , Araki Kenji . (2006). Extraction Based Summarization Using a Shortest Path Algorithm. *Proceedings of 12th Annual Language Processing Conference* .

Weiss Sholom M., Indurkhya Nitin, Zhang Tong, Damerau Fred J. (2005). *Text Mining Predictive Methods for Analyzing Unstructured Information.* Springer.

West, D. B. (2001). *Introduction to Graph Theory Second Edition.* United States of America: Prentice-Hall.