KOMBINASI TEKNIK CHI SQUARE DAN SINGULAR VALUE DECOMPOSITION UNTUK REDUKSI FITUR PADA PENGELOMPOKAN DOKUMEN

Catur Supriyanto[1], Affandy[2]
[1,2]*Fakulti Teknologi Maklumat dan Komunikasi, Universiti Teknikal Malaysia, Melaka*
E-mail : [1]supriyanto_catur@yahoo.co.id, [2]affandy_ra@utem.edu.my

ABSTRAK

Clustering dokumen adalah proses pengelompokan dokumen yang memiliki kesamaan topik. Metode Vector Space Model (VSM) merepresentasikan sekumpulan dokumen dalam bentuk matrik term-document, dimana setiap kolomnya mewakili dokumen dan setiap barisnya mewakili term (kata) yang terkandung dalam dokumen tersebut. Masalah yang terjadi dalam VSM adalah besarnya dimensi dan banyaknya nilai nol yang dihasilkan pada matrik term-document. Hal ini dapat mengurangi performa dari proses pengelompokan dokumen. Penelitian sebelumnya menunjukkan bahwa Latent Semantic Indexing (LSI) dengan menggunakan Singular Value Decomposition (SVD) mampu mereduksi besarnya dimensi matrik, namun SVD membutuhkan waktu proses komputasi yang relatif lama. Kajian ini mengusulkan penggunaan seleksi fitur untuk mengatasi kelemahan tersebut, dimana seleksi fitur akan menyeleksi term-term yang memiliki kontribusi yang besar untuk penentuan topik sebuah dokumen. Tahap preprocessing yang diusulkan meliputi tokenization, stopword removal dan stemming. Penelitian akan memfokuskan pemanfaatan chi-square sebagai seleksi fitur dan SVD untuk diterapkan dalam k-means clustering. Hasil penelitian menunjukan bahwa penggunaan chi-square mampu meningkatkan performa SVD dalam proses pengelompokan 150 dokumen. Sebanyak 1991 term berhasil diperoleh setelah tahap preprocessing dilakukan. Setelah melalui tahap seleksi fitur, rank-10 SVD dengan menggunakan 10 term dapat meningkatkan nilai F-measure dari 0.92 menjadi 0.97, serta dapat menurunkan waktu komputasi dari SVD hingga 48 persen.

Kata kunci : Clustering, Dokumen, VSM, SVD, Chi-square

1. PENDAHULUAN

Clustering dokumen adalah proses pengelompokan dokumen yang memiliki kesamaan topik. *Clustering* dokumen memudahkan pengguna menemukan dokumen yang diinginkan [1]. *Carrot2*[1] adalah sebuah contoh dari *web search engine* yang menerapkan algoritma *clustering*. Sebagai contoh, ketika pengguna mencari dokumen dengan kata kunci *apple*, maka *carrot2* akan menampilkan *cluster apple* yang bermakna buah dan *cluster apple* yang bermakna perusahaan *computer*. Fitur ini tidak didapati pada Google karena memang Google tidak mengaplikasikan teknik *clustering* pada hasil temuannya.

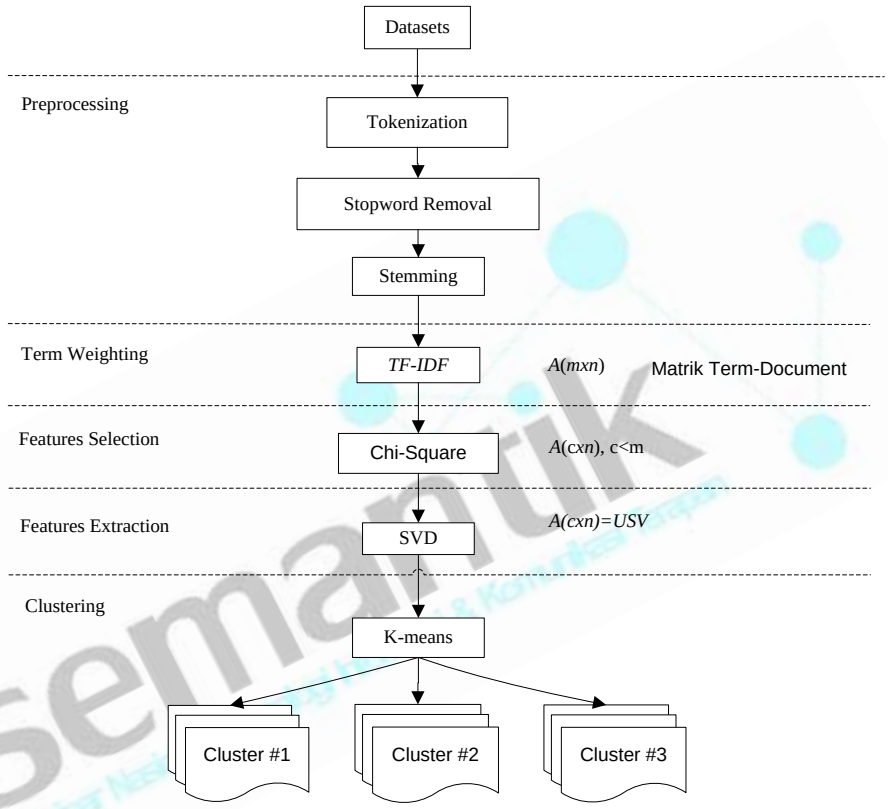
Beberapa tahapan dasar sebelum *clustering* dokumen dilakukan, yaitu *preprocessing*, *term weighting* dan penerapan algoritma *clustering*. Tahapan *preprocessing* akan menghasilkan kumpulan *term* atau kata yang nantinya akan diberikan bobot atau nilai dimana bobot tersebut mengindikasikan pentingnya sebuah *term* terhadap dokumen. Semakin banyak *term* tersebut muncul pada koleksi dokumen, semakin tinggi nilai atau bobot *term* tersebut. Pemberian bobot tersebut dinamakan *term weighting*. Setelah tahapan pemberian bobot selesai, maka akan dihasilkan sebuah matrik *term-document* dengan dimensi *mxn*, dimana *m* adalah jumlah *term* dan *n* adalah jumlah dokumen. Model sistem temu kembali informasi seperti ini disebut model ruang vektor atau *vector space model* (VSM). VSM banyak digunakan untuk algoritma *clustering* [2] namun masalah yang terjadi pada penggunaan VSM adalah besarnya ruang vektor atau besarnya dimensi pada matrik *term-document*. Masalah tersebut dapat mengakibatkan penurunan kinerja dari pengelompokan dokumen [3].

[1] http://search.carrot2.org/stabel/search

Beberapa penelitian dilakukan untuk mengatasi permasalahan tersebut yaitu dengan menggunakan Latent Semantic Indexing (LSI) melalui Singular Value Decomposition (SVD). SVD bekerja dengan mengurai matrik term-document menjadi matrik yang berdimensi lebih kecil. Namun, [4] melaporkan bahwa LSI melalui SVD memerlukan waktu yang relatif lama dalam melakukan proses perhitungan.

Berdasarkan kelemahan yang ada pada SVD tersebut, penelitian ini mengusulkan penggunaan *chi-square* sebagai seleksi fitur dalam *clustering* dokumen. *Chi-square* akan menyeleksi *term-term* yang memiliki kontribusi terhadap penentuan topik sebuah dokumen. Penggunaan *chi-square* sebelum tahapan SVD diharapkan mampu mengurangi waktu perhitungan SVD sehingga dapat meningkatkan kinerja dari *clustering* dokumen. Sebagai bahan evaluasi dari metode yang diusulkan, penelitian ini menggunakan algoritma *clustering* yang sederhana yakni algoritma *k-means*.

2. DESAIN SISTEM



Gambar 1: Desain Sistem

Untuk mengatasi permasalahan di atas penulis mencoba untuk mengusulkan desain sistem yang merupakan modifikasi dari sejumlah sistem yang telah ada. Gambar 1 menunjukkan urutan proses dari desain sistem yang diusulkan:

1. Tahapan pengumpulan *datasets* yang berupa dokumen teks.
2. Tahapan *preprocessing* yang terdiri dari *tokenization*, *stopword removal* dan *stemming*.
3. Tahapan *term weighting* yang akan menghasilkan matrik *term-document* dengan dimensi *m* x *n* (Matrik $A_{m \times n}$) dimana *m* adalah jumlah *term* dan *n* adalah jumlah dokumen.
4. Tahapan fitur seleksi dengan menggunakan *chi-square* yang bertujuan untuk menyeleksi *term-term* hasil dari *preprocessing* untuk proses pengelompokan dokumen, sehingga dimensi matrik *term-document* menjadi *c* x *n* (Matrik $A_{c \times n}$) dimana *c* adalah jumlah *term* yang terpilih untuk clustering dan *n* adalah jumlah dokumen.
5. Tahapan fitur ekstraksi dengan menggunakan SVD. SVD akan mengurai matrik *term-document* menjadi 3 matrik yang berdimensi lebih kecil ($A_{c \times n} = U*S*V^T$).

6. Menggunakan matrik V^T untuk digunakan dalam proses pengelompokan dokumen dengan algoritma *k-means*.

Sistem akan menggunakan *Lucene*[2] sebagai *java library*. *Lucene* menyediakan fungsi untuk *stopword removal* dan *stemming* untuk tahapan *preprocessing*. *Lucene* juga menyediakan perhitungan pembobotan dengan metode *Term Frequency Invers Document Frequency* (TFIDF) dan perhitungan *cosines similarity* untuk menghitung kemiripan antar dokumen.

3. LANDASAN TEORI

3.1 Preprocessing

Preprocessing adalah tahapan mengubah suatu dokumen kedalam format yang sesuai agar dapat diproses oleh algoritma *clustering*. Terdapat 3 tahapan preprocessing dalam penelitian ini, yaitu:

1. *Tokenization*, merupakan tahapan penguraian string teks menjadi *term* atau kata.
2. *Stopword Removal*, merupakan tahapan penghapusan kata-kata yang tidak relevan dalam penentuan topik sebuah dokumen dan yang sering muncul pada sebuah dokumen, misal "*and", "or", "the", "a", "an"* pada dokumen berbahasa inggris.
3. *Stemming*, merupakan tahapan pengubahan suatu kata menjadi akar kata nya dengan menghilangkan imbuhan awal atau akhir pada kata tersebut, misal *eating* → *eat*, *extraction* → *extract*. Penelitian ini menggunakan algoritma *porter stemmer*.

3.2 Vector Space Model

Vector space model banyak digunakan dalam sistem temu kembali dokumen teks [5]. VSM adalah model yang digunakan untuk mengukur kemiripan antar dokumen. VSM mengubah koleksi dokumen kedalam matrik *term-document* [6]. Matrik *term-document* (Gambar 2) tersebut memiliki dimensi *mxn* dimana *m* adalah jumlah *term* dan *n* adalah jumlah dokumen. Terdapat 3 metode pembobotan atau *term weighting* dalam VSM yaitu *Term Frequency* (TF), *Invers Document Frequency* (IDF) dan *Term Frequency Invers Document Frequency* (TFIDF). TF adalah banyaknya kemunculan suatu *term* dalam suatu dokumen, IDF adalah perhitungan logaritma antara pembagian jumlah total dokumen dengan cacah dokumen yang mengandung suatu *term*, dan TFIDF adalah perkalian antara TF dengan *IDF*. Semakin besar bobot TFIDF pada suatu term, semakin penting term tersebut untuk digunakan pada tahapan. Dalam penelitian ini digunakan TFIDF sebagai metode *term weighting* dim[illegible]ng digunakan.

$$A_{mxn} = \begin{array}{c} \begin{matrix} d1 & d2 & \cdots & dn \\ \downarrow & \downarrow & & \downarrow \end{matrix} \\ \begin{bmatrix} \omega_{11} & \omega_{12} & \cdots & \omega_{1n} \\ \omega_{21} & \omega_{22} & \cdots & \omega_{2n} \\ \vdots & & & \vdots \\ \omega_{m1} & \omega_{m2} & \cdots & \omega_{mn} \end{bmatrix} \end{array} \begin{array}{l} \leftarrow t1 \\ \leftarrow t2 \\ \\ \leftarrow tm \end{array}$$

Gambar 2: Matrik Term-Document

$$IDF = \log \frac{D}{DF} \quad (1)$$

$$TFIDF(t) = TF * \log \frac{D}{DF} \quad (2)$$

[2] http://lucene.apache.org

Dokumen dalam VSM direpresentasikan dalam bentuk vektor $d=\{w_1,w_2,w_3,...w_n\}$ dimana *d* adalah dokumen dan *w* adalah nilai atau bobot setiap *term* dalam dokumen tersebut. Untuk menghitung persamaan antar dokumen, penelitian ini menggunakan rumus *cosines similarity*. *Cosines similarity* akan mengukur jarak antara 2 dokumen d_i dan d_j, besarnya nilai *cosines* mengindikasikan bahwa 2 dokumen tersebut memiliki nilai kemiripan yang tinggi. Berikut adalah rumus dari perhitungan *cosines similarity*.

$$similarity(d_i, d_j) = cosines\ \theta = \frac{\overrightarrow{di}.\overrightarrow{dj}}{\|di\|.\|dj\|} \qquad (3)$$

3.3 Seleksi Fitur Chi-Square

Seleksi fitur adalah proses menghilangkan beberapa fitur atau *term* yang kurang relevan untuk penentuan topik suatu dokumen. Terdapat 2 pembagian dalam seleksi fitur yaitu seleksi fitur *supervised* dan *unsupervised*. Seleksi fitur yang termasuk ke dalam kategori *supervised* adalah *Chi-Square* (χ^2), *Information Gain* (IG) dan *Mutual Information* (MI), sedangkan seleksi fitur yang termasuk kedalam kategori *unsupervised* adalah *Term Strength* (TS), *Term Contribution* (TC), *Entropy-based Ranking* (En) dan *document frequency* (DF) [7]. Perbedaan antara keduanya adalah keberadaan informasi awal tentang kategori dari suatu dokumen. Dalam pengklasifikasian sebuah dokumen, *Chi-square* adalah salah satu *supervised* seleksi fitur yang mampu mengilangkan banyak fitur tanpa mengurangi tingkat akurasi [8].

$$\chi^2(t,c) = \frac{N(AxD-BxC)^2}{(A+B)x(C+D)x(A+C)x(B+D)} \qquad (4)$$

Keterangan: A : Banyaknya dokumen dalam kategori *c* yang mengandung *term t*
B : Banyaknya dokumen yang bukan kategori *c* tetapi mengandung *term t*
C : Banyaknya dokumen dalam katgori *c* tetapi tidak mengandung *term t*
D : Banyaknya dokumen yang bukan kategori *c* dan tidak mengandung *term t*
N : Total keseluruhan dokumen

Untuk menghitung rata-rata nilai *chi-square* dari suatu *term*, digunakan rumus di bawah ini.

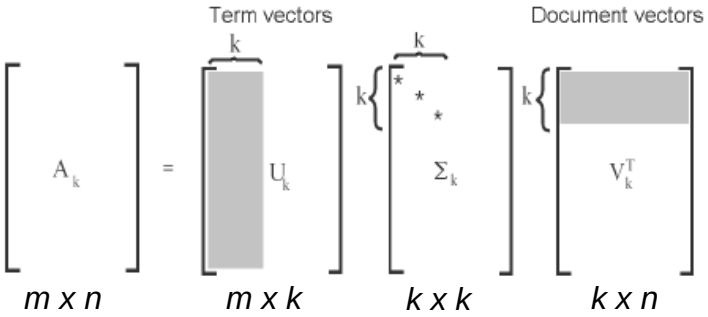
$$\chi^2{}_{avg}(t)=\sum_{i=1}^{m} P(Ci)\,\chi^2(t,Ci) \qquad (5)$$

3.4 Latent Semantic Indexing

Dalam *space vector model*, matrik *term-document* memiliki dimensi *mxn* dimana *m* adalah jumlah *term* dan *n* adalah jumlah dokumen. *Latent Semantic Indexing* (LSI) melalui *Dekomposisi Nilai Singular* (SVD) akan mengurai matrik *term-document* menjadi 3 matrik *U, S* dan *V* yang memiliki dimensi lebih kecil

$$A = USV^T \qquad (6)$$

Matrik *U* adalah matrik *term* yang memiliki dimensi *m* x *k*, *S* adalah matriks diagonal yang berisi *eigen value* yang memiliki dimensi *kxk* dan V^T adalah matrik dokumen yang memiliki dimensi *k* x *n*. Gambar 2 menunjukkan dekomposisi *truncated* SVD.



Gambar 3: *Truncated* SVD

SVD akan menggunakan pendekatan *rank-k* untuk mengurangi dimensi. Pendekatan seperti ini disebut *truncated* SVD [9]. Matrik yang dihasilkan dari pendekatan *rank-k* SVD memiliki tingkat kemiripan dengan matrik *term-document*. Sehubungan dengan *truncated* SVD, [10] menyatakan bahwa biasanya *rank-k* menggunakan angka-angka kecil.

3.5 K-Means Clustering

K-means adalah algoritma *clustering* yang cukup sederhana dan mampu diimplementasikan untuk koleksi data yang besar. K-means mencoba untuk mengelompokan dokumen kedalam beberapa *cluster* [11]. K-means akan memilih beberapa dokumen secara acak untuk dijadikan *centroid* atau pusat *cluster*. Banyaknya *centroid* menentukan jumlah *cluster* yang akan dihasilkan. Berikut adalah *pseudocode* dari algoritma *k-means* [12]:

Algoritma *K-means Clustering*

Input : Koleksi Dokumen $D=\{d_1, d_2, d_3, \ldots d_n\}$;
Jumlah *cluster* (*k*) yang akan dibentuk*;*

Output : *k cluster;*

Proses :
1. Memilih *k* dokumen untuk dijadikan *centroid* (titik pusat *cluster*) awal secara random*;*
2. Hitung jarak setiap dokumen ke masing-masing *centroid* menggunakan persamaan *cosines similarity* (persamaan 3) kemudian jadikan satu *cluster* untuk tiap-tiap dokumen yang memiliki jarak terdekat dengan *centroid*;
3. Tentukan *centroid* baru dengan cara menghitung nilai rata-rata dari data-data yang ada pada *centroid* yang sama;
4. Kembali ke langkah 2 jika posisi *centroid* baru dan centroid lama tidak sama;

4. EVALUASI PERFORMA

Beberapa cara untuk mengukur kualitas performa dari *clustering* yaitu dengan menggunakan mutual information metrix, misclassification index, purity, F-measure, confusion metric dan entropy [13]. Dalam penelitian ini, digunakan F-measure untuk mengukur kinerja *clustering*. F-measure diperoleh dari pengukuran *recall* dan *precision*. *Recall* adalah rasio dokumen yang relevan yang terambil dengan jumlah seluruh dokumen dalam koleksi dokumen, sedangkan *precision* adalah rasio jumlah dokumen relevan terambil dengan seluruh jumlah dokumen terambil. Nilai interval *recall* dan *precision* berada antara 0 dan 1 [14]. Nilai *recall* dan *precision* yang tinggi menunjukan keakuratan dari sebuah clustering [15]. *Recall* dan *precision* kategori *i* dalam *cluster j* diperoleh dari persamaan berikut.

$$Recall\ (i,j) = \frac{n_{ij}}{n_i} \qquad (7)$$

$$\boldsymbol{Precision\ (i,j) = \frac{n_{ij}}{n_j}} \qquad (8)$$

Dimana n_{ij} adalah jumlah dokumen kategori *i* dalam *cluster j*, n_i adalah jumlah dokumen dalam kategori *i* dan n_j adalah jumlah dokumen dalam *cluster j*. Sedangkan untuk menghitung F-measure yang digunakan persamaan berikut.

$$F(i,j) = \frac{2*(Precision*Recall)}{(Precision+Recall)} \qquad (9)$$

Secara keseluruhan, rata-rata dari F-measure dihitung dengan persamaan berikut.

$$F = \sum_i \frac{n_i}{n} \max_{j=1,\ldots,k} F(i,j) \qquad (10)$$

Dimana $\max\{F(i,j)\}$ adalah nilai maksimum F-measure dari kategori *i* dalam *cluster j*. Semakin tinggi nilai F-measure, semakin tinggi tingkat akurasi dari *clustering* [16].

5. HASIL DAN PEMBAHASAN

5.1 Datasets

Datasets dalam penelitian ini berjumlah 150 abstrak yang dikumpulkan dari *journal* dan *conference* IEEE. Datasets terbagi menjadi 3 kategori: *data mining*, *machine learning* dan *semantic web*. Sebanyak 1991 *term* berhasil diperoleh dari tahapan *preprocessing*. Jumlah *cluster* yang akan dibentuk pada penelitian ini sebanyak 3 *cluster* dimana setiap *cluster* adalah representasi dari masing-masing kategori. Tabel 1 menunjukkan jumlah dokumen pada setiap kategori. Datasets disimpan dalam satu file dengan format *Extensible Markup Language* (XML) seperti pada gambar 4.

Tabel 1: Deskripsi Datasets

Kategori	*Data Mining <cluster 1>*	*Machine Learning <cluster 2>*	*Semantic Web <cluster 3>*
Jumlah Dokumen	50	50	50

```
<datasets>
        <datasets category="datamining">
        <docID>DM001</docID>
        <title>Agent Mining: The Synergy of Agents and Data Mining</title>
        <abstract>Autonomous agents and multiagent systems ...</abstract>
        </datasets>
</datasets>
```

Gambar 4: Koleksi dokumen dalam format xml

5.2 Percobaan

Terdapat 2 percobaan dalam penelitian ini. Percobaan pertama adalah penggunaan SVD dalam *clustering*. Dalam percobaan pertama, nilai *rank-k* yang digunakan adalah *rank-k* = 2, 4, 5, 6, 7, 8, 9, 10. [17] melaporkan bahwa nilai *rank-k* yang kecil memiliki performa yang baik untuk menghasilkan F-measure yang tinggi. Percobaan kedua adalah penggunaan *chi-square* dan SVD secara berurutan dalam proses clustering. Tiap nilai *rank-k* pada percobaan pertama akan digunakan untuk diuji menggunakan jumlah *term* yang berbeda-beda dari hasil seleksi fitur *chi-square*. Jumlah *term* yang akan diuji dalam percobaan ke dua ini adalah 10, 50, 100, 500, 1000 dan 1500. Kinerja dari *clustering* akan diukur dari akurasi dan *time taken* yang diambil dari pengelompokan pada percobaan pertama dan kedua. *Time taken* adalah waktu yang diperlukan oleh algoritma untuk menghasilkan *cluster*.

5.2.1 Akurasi

Seperti terlihat pada tabel 2, SVD dapat meningkatkan kinerja *k-means* terutama pada *rank-3 dan rank-4* dimana nilai F-measure mencapai 0.99. Dapat dibandingkan dengan *original k-means* yang memiliki F-measure 0,94. *Original k-means* adalah pengelompokan dokumen oleh *k-means* yang tidak menggunakan seleksi fitur atau ekstraksi fitur didalam proses clusteringnya.

Tabel 2: Nilai F-measure pada penerapan SVD

Rank-k	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*
F-measure	0.73	0.99	0.99	0.98	0.97	0.92	0.92	0.92	0.93

Ketika *chi-square* diterapkan sebelum tahapan SVD, ditemukan bahwa penggunaan jumlah *term* yang sedikit (*term* yang di hasil dari seleksi *chi-square*) dapat meningkatkan kinerja *clustering*. Nilai *chi-square* yang tinggi pada sebuah *term* menunjukkan bahwa *term* tersebut memiliki kontribusi yang besar dalam menentukan topik sebuah dokumen. Pada tabel 3 terdapat 10 *term* dengan nilai *chi-square* tertinggi dan 5 *term* diantaranya dapat mewakili topik sebuah dokumen: *machine*, *learning*, *mining*, *semantic* dan *web*. Tabel 4 dan gambar 5 menunjukkan penilaian F-measure pada *clustering* ketika diterapkan *chi-square* dan

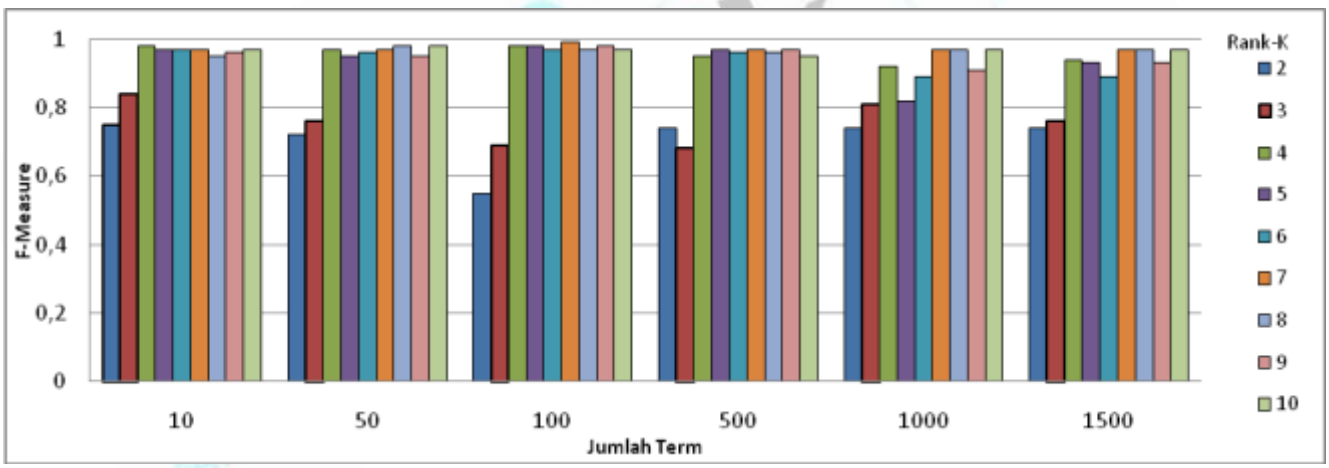
SVD. *Rank-10* memiliki kinerja yang lebih baik dan stabil dibandingkan *rank-k* yang lain karena *rank-10* menggunakan 10 *term* tertinggi yang dihasilkan dari pemilihan fitur oleh *chi-square*, sedangkan *rank-2* dan *rank-3* memiliki nilai F-measure yang lebih rendah karena terjadi proses dekomposisi pada 10 *term* tersebut.

Tabel 3: 10 term yang memiliki nilai chi-square terbesar

No	***Term***	***Chi Square***	***No***	***Term***	***Chi Square***
1	Mining	285124.99385333277	6	Service	137565.62602794694
2	Machine	253881.8622314444	7	Description	101480.9639591654
3	Learning	239990.83062933508	8	Vector	90223.6530743876
4	Web	213782.52757795807	9	Ontology	84486.6196939402
5	Semantic	196078.71595703863	10	Large	78040.51203134487

Tabel 4: Nilai F-measure pada penerapan chi-square dan SVD

Jumlah Term	*Rank-k*								
	2	3	4	5	6	7	8	9	10
10	0.75	0.84	0.98	0.97	0.97	0.97	0.95	0.96	0.97
50	0.72	0.76	0.97	0.95	0.96	0.97	0.98	0.95	0.98
100	0.55	0.69	0.98	0.98	0.97	0.99	0.97	0.98	0.97
500	0.74	0.68	0.95	0.97	0.96	0.97	0.96	0.97	0.95
1000	0.74	0.81	0.92	0.82	0.89	0.97	0.97	0.91	0.97
1500	0.74	0.76	0.94	0.93	0.89	0.97	0.97	0.93	0.97



Gambar 5: Nilai F-measure pada penerapan *chi-square* dan SVD

5.2.2 Time Taken

Tabel 5 dan 6 adalah hasil dari *time taken* yang diperlukan dari *clustering* pada sistem operasi windows vista (2.0GHz AMD Turion, RAM 4G). Tabel 5 adalah hasil dari percobaan pertama, waktu yang dibutuhkan ketika SVD diterapkan untuk *k-means*. Tabel 6 adalah hasil dari percobaan kedua ketika *chi-square* diterapkan sebelum melakukan tahapan SVD. Seperti yang terlihat pada percobaan pertama (tabel 5), nilai *rank-k* yang kecil dapat mempercepat waktu proses *clustering*. Hal ini dapat dibandingkan dengan *original* k-means yang memerlukan waktu 3119 milidetik untuk proses *clustering* dokumen.

Tabel 5: Time taken pada penerapan SVD

Rank-k	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*
Time Taken	324	337	339	341	358	366	375	385	390

Tabel 6: Time taken pada penerapan chi-square dan SVD

Jumlah Term	*Rank-k*								
	2	3	4	5	6	7	8	9	10
10	180	187	189	192	191	194	190	199	210
50	186	220	221	210	207	205	210	200	215
100	212	227	225	221	225	222	219	219	220
500	216	233	239	234	238	235	223	212	225
1000	230	237	247	245	233	247	244	226	235
1500	253	257	251	260	260	250	251	265	244

Sedangkan ketika *chi-square* diterapkan sebagai seleksi fitur sebelum tahapan SVD (tabel 6), terlihat bahwa jumlah *term* yang sedikit dapat mempercepat waktu dari *clustering*. Ketika hanya SVD saja yang digunakan, waktu yang diperlukan lebih dari 300 milidetik. Dapat dibandingkan dengan saat diterapkan *chi-square* sebelum tahapan SVD, waktu yang diperlukan kurang dari 300 milidetik. Penggunaan *chi-square* pada *clustering* dokumen dapat menurunkan waktu komputasi dari SVD hingga 48 persen**.**

6. PENUTUP

Penelitian ini mengusulkan penggunaan *chi-square* sebagai seleksi fitur untuk diterapkan sebelum tahapan SVD dalam proses *clustering* dokumen. Metode ini mampu mengatasi masalah dari penggunaan SVD yang memiliki waktu komputasi yang cukup lama. Hasil percobaan menunjukkan bahwa penggunaan *chi-square* sebelum tahapan SVD memiliki kinerja yang lebih baik dalam hal F-measure dan *time taken*. Penggunaan *chi-square* dapat mengurangi waktu komputasi *clustering* dokumen tanpa mengurangi tingkat akurasi dari *clustering*. Untuk penelitian lebih lanjut, diusulkan penggunaan fitur seleksi pada *online clustering*, dimana koleksi dokumen pada *online clustering* bersifat dinamis.

DAFTAR PUSTAKA

[1] H. Al-mubaid and A.S. Umair, "A new text categorization technique using distributional clustering and learning logic," *IEEE Trans. Knowl. Data Eng*, vol. 18, 2006, pp. 1156-1165.

[2] M. Thangamani and P. Thangaraj, "Integrated Clustering and Feature Selection Scheme for Text Documents," *Journal of Computer Science*, vol. 6, 2010, pp. 536-541.

[3] C.C. Aggarwal and P.S. Yu, "Finding Generalized projected Cluster in High Dimensional Spaces," *Proc. of SIGMOD 2000*, 2000, pp. 70-81.

[4] T.G. Kolda and D.P. Leary, "A Semidiscrete Matrix Decomposition for Latent Semantic Indexing in Information Retrieval," *ACM Transactions on Information Systems*, vol. 16, 2008, pp. 322-346.

[5] L.S. Wang, "Relevance Weighting of Multi-Term Queries for Vector Space Model," *Proc. of Computational Intelligence and Data Mining*, Nashville, TN: 2009, pp. 0-6.

[6] R. Peter, S. G, D. G, and S. Kp, "Evaluation of SVD and NMF Methods for Latent Semantic Analysis," *International Journal of Recent Trends in Engineering*, vol. 1, 2009, pp. 308-310.

[7] Z. Chen and W.M. Com, "An Evaluation on Feature Selection for Text Clustering," *Proc. of ICML 2003*, Washington DC: 2003.

[8] C. Sun, X. Wang, and J. Xu, "Study on Feature Selection in Finance Text Categorization," *Science And Technology*, 2009, pp. 5077-5082.

[9] Q. Yang, "Support vector machine for customized email filtering based on improving latent semantic indexing," *Proceedings of the Fourth International Conference on Machine Learning and Cybernetics*, vol. 6, 2005, pp. 3787 - 3791.

[10] M.M. Hasan and Y. MATSUMOTO, "Document Clustering : Before and After the Singular Value Decomposition," *Joho Shori Gakkai Kenkyu Hokoku*, vol. 99, 1999, pp. 47-54.

[11] M. Shameem and R. Ferdous, "An efficient K-Means Algorithm integrated with Jaccard Distance Measure for Document Clustering," *Proc. of AH-ICI 2009*, Kathmandu: IEEE, 2009, pp. 1-6.

[12] M.H. Dunham, *Data Mining- Introductory and Advanced Concepts*, Pearson Education, 2006.

[13] I. Yoo and X. Hu, "A Comprehensive Comparison Study of Document Clustering for a Biomedical Digital Library MEDLINE," *JCDL'06*, 2006, pp. 220-229.

[14] S.K. P and E.V. Prasad, "A Novel Document Representation Model for Clustering," *International Journal of Computer Science & Communication*, vol. 1, 2010, pp. 243-245.

[15] L. Muflikhah and B. Baharudin, "Document Clustering using Concept Space and Cosine Similarity Measurement," *International Conference on Computer Technology and Development*, Kota Kinabalu: 2009, pp. 58 - 62.
[16] A. Güven, Ö.Ö. Bozkurt, and O. Kalıpsız, "Advanced Information Extraction with n-gram based LSI," *World Academy of Science, Engineering and Technology*, vol. 17, 2006, pp. 13-18.
[17] H. Kim, H. Park, and B.L. Drake, "Extracting unrecognized gene relationships from the biomedical literature via matrix factorizations," *Proc. of BMC Bioinformatics*, 2007, pp. 1-11.