TERAKREDITASI RISTEKDIKTI No. 36b/E/KPT/2016

Jurnal Rekayasa Elektrika

VOLUME 14 NOMOR 1 APRIL 2018

JRE	Vol. 14	No. 1	Hal 1–82	Banda Aceh, April 2018	ISSN. 1412-4785 e-ISSN. 2252-620X

Jurnal Rekayasa Elektrika Vol. 14, No. 1, April 2018, hal. 35-42
ISSN. 1412-4785; e-ISSN. 2252-620X, Terakreditasi RISTEKDIKTI No. 36b/E/KPT/2016
DOI: 10.17529/jre.v14i1.9027

Perbandingan Metode Klaster dan Preprocessing untuk Dokumen Berbahasa Indonesia

Amalia, Maya Silvi Lydia, Siti Dara Fadillah, dan Miftahul Huda
Program Studi Ilmu Komputer, Universitas Sumatera Utara
Jl. Dr. T. Mansur No. 9, Kampus Padang Bulan, Medan 20155
e-mail: amalia@usu.ac.id

***Abstrak*—Pengelompokan dokumen tanpa bimbingan (*unsupervised*) atau disebut juga clustering merupakan metode pengelompokan secara otomatis beberapa objek menjadi satu grup berdasarkan kesamaan ciri tertentu. Kualitas akurasi *clustering* ditandai banyaknya objek yang mirip dalam satu grup klaster. Peningkatan akurasi clustering dapat dilakukan dengan meningkatkan ketangguhan pada proses *preprocessing* dan juga pemilihan algoritma *clustering*. Penelitian ini merupakan penelitian eksperimental untuk mengobservasi proses klaster yang cocok untuk diimplementasikan pada dokumen berbahasa Indonesia. Eksperimen dilakukan untuk beberapa algoritma klaster seperti K-Means, K-Means++ dan *Agglomerative* dengan beberapa tahapan *preprocessing* yang bervariasi. Ujicoba klasterisasi dilakukan pada sebuah korpus yang berisi 100 dokumen Bahasa Indonesia untuk variasi *preprocessing* umum dengan penambahan fungsi LSA, penambahan fungsi TF-IDF dan juga penambahan fungsi LSA/TF-IDF. Untuk fungsi LSA kami menguji reduksi dimensi matrik dari 10% sampai dengan reduksi 90%. Hasil penelitian ini menunjukkan kisaran nilai persentasi reduksi untuk LSA yang baik adalah antara 50%–80%. Algoritma K-Means++ menghasilkan nilai purity yang lebih baik dari algoritma lainnya yang berarti memiliki akurasi klaster yang paling baik di antara algoritma lainnya.**

Kata kunci: ***klasterisasi dokumen, komparasi algoritma klaster, purity, preprocessing, dokumen Bahasa Indonesia***

***Abstract*— Clustering is an unsupervised method to group multiple objects based on the similarity automatically. The quality of clustering accuracy is determined by the number of similar objects in a correct cluster group. The robust preprocessing process and the choice of cluster algorithm can increase the efficiency of clustering. The objective of this study is to observe the most suitable method to cluster document in Bahasa Indonesia. We performed tests on several cluster algorithms such as K-Means, K-Means++ and Agglomerative with various preprocessing stages and collected the accuracy of each algorithm. Clustering experiments were conducted on a corpus containing 100 documents in Bahasa Indonesia with a commonly used preprocessing scenario. Additionally, we also attach our preprocessing stages such as LSA function, TF-IDF function, and LSA / TF-IDF function. We tested various LSA dimension reductions values from 10% to 90%, and the result shows that the best percentage of reduction rates between 50%-80%. The result also indicates that K-Means++ algorithm produces better purity values than other algorithms.**

Keywords: ***document clustering; cluster algorithm comparison; purity; preprocessing; Indonesian Documents***

I. PENDAHULUAN

Optimasi proses pencarian informasi berupa dokumen teks merupakan salah satu bidang penelitian yang berkembang pesat saat ini. Salah satu cara yang dapat diterapkan adalah dengan melakukan klasterisasi atau pengelompokan terlebih dahulu sebelum dilakukan proses pencarian informasi. Klasterisasi (*clustering*) adalah proses pengelompokan objek secara otomatis. Hal utama yang harus diperhatikan pada proses klasterisasi adalah akurasi klaster. Akurasi klaster dipengaruhi oleh banyak hal, salah satunya adalah pemilihan tahapan *preprocessing* dan juga pemilihan algoritma klaster. Tahapan *preprocessing* adalah tahapan awal untuk mempersiapkan dokumen agar lebih mudah untuk diproses.

Secara umum, tahapan *preprocessing* sebelum proses klasterisasi meliputi *case folding*, *tokenizing*, *filtering*, dan *stemming*. Tahapan *preprocessing* selanjutnya adalah membentuk data representasi dokumen. Contoh representasi dokumen adalah Vector Space Model (VSM) dan juga Latent Semantic Analysis (LSA). VSM merupakan representasi dokumen dalam bentuk vektor atau matrik yang merepresentasikan frekuensi kemunculan term untuk setiap dokumen pada korpus [1]or other pattern matching environment where stored entities (documents. Selain frekuensi kemunculan kata, sel pada matrik VSM dapat juga merupakan representasi bobot kata yang dikalkulasi dengan metode *Term Frequency and Inverse Document Frequency* (TF-IDF). VSM merupakan suatu representasi data yang banyak digunakan karena

Received 04 December 2017; Revised 02 April 2018; Accepted 07 April 2018

kesederhanaan prosesnya dan juga merupakan *baseline* awal untuk beberapa modifikasi representasi data tekstual [2] [3]. Kendala utama dari VSM yaitu menghasilkan jumlah dimensi yang besar sehingga diyakini tidak relevan untuk diterapkan di era internet yang menghasilkan dokumen dengan jumlah besar. Untuk itu diperlukan suatu proses reduksi dimensi dengan mengeliminasi *noise* atau data yang tidak dibutuhkan [4]. Pada penelitian ini, untuk mengantisipasi dimensi data yang besar, maka sebelum data direpresentasikan dalam bentuk VSM, dilakukan proses *filtering*. Fungsi *filtering* pada penelitian ini adalah proses pengurangan dimensi matriks VSM dengan hanya mengambil kata yang penting dan membuang kata-kata yang dianggap tidak penting, dimana pada penelitian ini kami mengeliminasi 758 *stopwords* dari sumber *stopwords* Tala [5]. Pengeliminasian kata-kata tidak penting ini secara signifikan dapat mengurangi dimensi data representasi [6] karena walaupun tidak memiliki makna penting namun kata-kata *stopwords* ini sering muncul di dokumen-dokumen Bahasa Indonesia [5]. LSA merupakan metode statistik aljabar yang mengekstrak struktur semantik yang tersembunyi dari kata dan kalimat [7]. LSA menggunakan metode *Singular Value Decomposition* (SVD) untuk mencari interelasi di antara kalimat dan kata [8], [9]1997. SVD ini mempunyai kapasitas reduksi *noise* yang membantu untuk meningkatkan akurasi klasterisasi. Pemilihan algoritma klaster juga merupakan faktor yang berpengaruh terhadap akurasi hasil klaster. Secara umum, terdapat 2 algoritma klaster yaitu *flat clustering* dan *hierarchy clustering*. *Flat clustering* seperti K-Means dan K-Means++ merupakan pengklasteran dokumen yang klaster hasilnya tidak memiliki struktur eksplisit yang menghubungkan klaster satu dengan klaster lain sedangkan *hierarchy clustering* seperti *agglomerative* merupakan pengklasteran dokumen yang menghasilkan klaster yang memiliki gugusan hirarki [10].

Penelitian terdahulu yang membahas tentang klasterisasi data tekstual pernah dilakukan oleh [2], [3], [11]–[22]. Penelitian [2] menerapkan LSA untuk mengantisipasi masalah polisemi dan sinonim. Penelitian [11] dan [18] mengimplementasikan metode SVD untuk mereduksi dimensi dari matrik dokumen dan hasil penelitian ini menunjukkan bahwa kualitas hasil klaster sebanding dengan kualitas untuk klasterisasi tanpa pengurangan dimensi. Penelitian [12] merupakan penelitian yang membandingkan hasil akurasi klaster jika beberapa pilihan *preprocessing* diterapkan, hasil penelitian ini menunjukkan representasi LSA dengan penambahan fungsi pembobotan kata TF-IDF dapat meningkatkan akurasi klasterisasi. Penelitian [17] dan [18] mengobservasi tentang penggunaan algorima klasterisasi K-Means, hasil penelitian ini mengungkapkan bahwa K-Means merupakan algoritma yang baik namun jumlah dokumen dalam klaster selalu berubah setiap tahapan pengujian. Hal ini disebabkan karena metode K-Means sangat sensitif terhadap *outlier* dan sangat dipengaruhi oleh nilai *centroid* awal [17][18]. Penelitian oleh [19] menghasilkan algoritma K-Means++ yang merupakan suatu penyempurnaan algoritma K-Means. Pada algoritma K-Means++ proses penentuan *centroid* awal dilakukan dengan hati-hati yaitu dengan menghitung nilai probabilitas yang sebanding dengan jarak kuadrat terdekat dari *centroid* yang ada. Penelitian oleh [20] melakukan modifikasi terhadap algoritma K-Means++ dengan cara implementasi Markov Chain Monte Carlo hasil penelitian menunjukkan dengan modifikasi ini, walaupun *run time* klasterisasi berkurang drastis namun kualitas klaster yang dihasilkan sama dengan K-Means++. Penelitian oleh [21] merupakan komparasi antara algoritma klaster K-Means dan *agglomerative* pada dataset Bahasa Inggris, hasil yang tak terduga menunjukkan kualitas akurasi K-Means lebih baik dibandingkan algoritma *agglomerative*. Padahal klaster hirarki sering dianggap memiliki kualitas klaster yang lebih baik. Penelitian lainnya yaitu [22] melakukan implementasi algoritma K-Means dan K-Means++ dengan dua buah pencarian jarak yang berbeda yaitu *cosine similarity* dan *jaccard coefficient*. Pada penelitian ini terlihat bahwa penggunaan *cosine similarity* lebih baik dari pada dengan menggunakan *jaccard coefficient*.

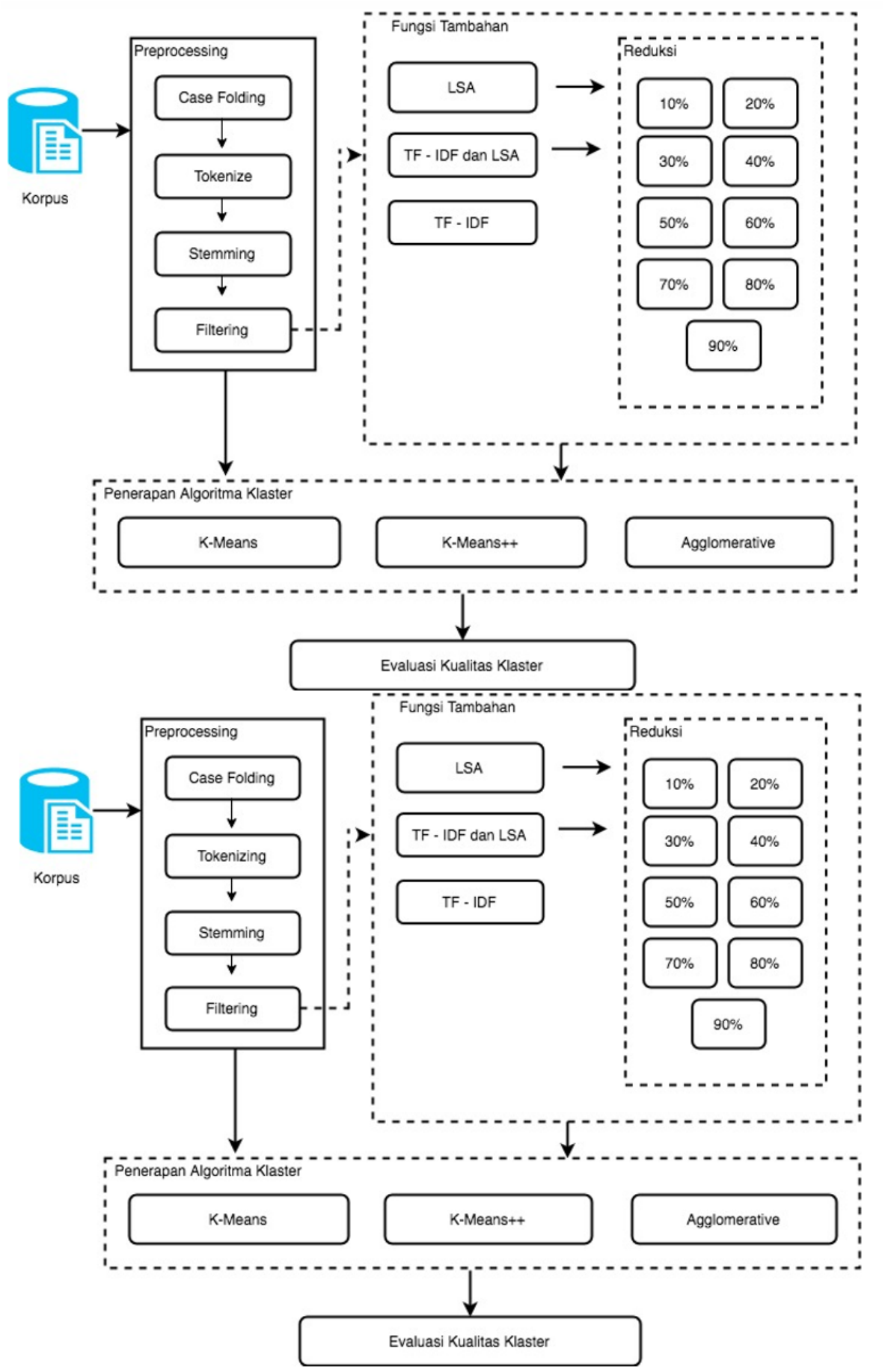
Berdasarkan latar belakang ini dapat diketahui bahwa beberapa perlakuan dapat meningkatkan akurasi klaster. Penelitian ini merupakan penelitian ekperimental yang bertujuan untuk mengobservasi beberapa metode *preprocessing* dan algoritma klaster. Perbedaan penelitian ini dengan penelitian terdahulu yaitu dataset yang digunakan adalah dataset Bahasa Indonesia. Berbeda dengan peneliti lain yang sebagian besar menerapkan hanya satu metode, pada penelitian ini selain *preprocessing* umum kami juga menambahkan beberapa kombinasi metode *preprocessing* sekaligus seperti LSA, TF-IDF dan juga LSA digabung dengan TF-IDF. Kami juga melakukan percobaan untuk mencari nilai akurasi klaster dengan berbagai nilai reduksi SVD dari 10% sampai dengan reduksi 90%. Pada penelitian ini kami juga membahas tentang tambahan metode *preprocessing* yang bertujuan untuk reduksi dimensi matrik seperti LSA. Eksperimen untuk setiap metode dan persentasi reduksi diujikan pada tiga algoritma berbeda yaitu K-Means, K-Means++ dan algoritma *agglomerative*. Pada penelitian ini, kami menerapkan metode *cosine similarity* untuk pencarian *similarity* antar dokumen.

II. METODE

Penelitian ini merupakan penelitian eksperimental untuk mengobservasi metode klaster yang memiliki tingkat akurasi yang paling tinggi. Uji coba klasterisasi dilakukan pada sebuah korpus yang berisi 100 dokumen Bahasa Indonesia. Parameter yang digunakan dalam pengujian ini adalah *purity* klaster. Tahapan penelitian seperti pada Gambar 1.

A. Korpus

Salah satu kendala yang dihadapi pada penelitian klasterisasi untuk Bahasa Indonesia adalah keterbatasan



Gambar 1. Metode Penelitian

sumber daya seperti korpus untuk *dataset*. Pada penelitian ini korpus yang digunakan adalah sekumpulan dokumen Bahasa Indonesia yang diambil dari repositori skripsi mahasiswa Universitas Sumatera Utara yang terdiri dari beberapa topik klaster. Pemilihan skripsi sebagai *dataset* dengan alasan skripsi merupakan karya ilmiah yang harus memenuhi tata Bahasa Indonesia sehingga diharapkan dapat menjadi *benchmark* untuk dokumen berbahasa Indonesia. Total dokumen *dataset* sebanyak 100 dokumen dengan pembagian topik klaster seperti olahraga, komputer, hukum, kesehatan, gigi, dan mata. Masing-masing dokumen mencapai sekitar 15.000 kata. Masing-masing topik memiliki jumlah dokumen sekitar 16 dokumen.

B. Preprocessing

Preprocessing di sini adalah tahapan awal umum sebelum kumpulan dokumen diklasterisasi. Tujuan dari *preprocessing* antara lain untuk menyeragamkan *dataset* agar lebih mudah diolah. Terdapat beberapa proses umum *preprocessing* yaitu *case folding*, *tokenizing*, *stemming*, dan *filtering*.

Case folding adalah proses untuk menyeragamkan jenis huruf pada dokumen. Pada penelitian ini kami mengubah semua huruf menjadi huruf kecil. Proses *tokenizing* adalah proses untuk memecah dokumen menjadi kata per kata. *Filtering* adalah tahapan untuk membuang kata-kata yang dianggap tidak penting, kami menerapkan daftar *stopword* oleh Tala yang merupakan *stopword* untuk Bahasa Indonesia. *Filtering* juga berfungsi untuk mereduksi dimensi data sehingga tidak terlalu besar. *Stemming* adalah proses mengambil kata dasar dengan membuang imbuhan kata. Pada penelitian ini kami menerapkan algoritma Nazief Adriani [23] yang memang merupakan *stemming* untuk Bahasa Indonesia.

C. Fungsi Tambahan

Pada penelitian ini, setelah tahapan *preprocessing* umum, dokumen diproses dalam 2 pilihan yaitu tanpa proses fungsi tambahan dan melalui proses fungsi tambahan. Untuk fungsi tambahan, dokumen diproses dengan fungsi LSA, TF-IDF, dan LSA/TF-IDF. Masing-masing tahapan menghasilkan representasi data yang berbeda.

1. LSA

LSA menggunakan metode SVD untuk mencari interelasi di antara kalimat dan kata [8] [9]1997. SVD ini mempunyai kapasitas reduksi noise yang membantu untuk meningkatkan akurasi klasterisasi. Proses diawali dengan input matriks yang merepresentasikan dokumen dengan term dan nilai reduksi X. Lalu dilakukan proses dekomposisi menjadi tiga buah matriks U, S, V^T yang didapat dari proses SVD. Tiga buah matriks tersebut direduksi sebanyak X elemen. Matriks hasil reduksi dikalikan dan hasilnya dikembalikan. Pada penelitian ini akan dicari persentasi reduksi LSA yang optimum. Pada penelitian ini proses reduksi akan dibandingkan dengan tingkat reduksi yang berbeda-beda yaitu 10%, 20%, 30%, 40%, 50%, 60%, 70%, 80%, dan 90%.

2. TF-IDF

TF-IDF merupakan suatu metode untuk mencari pembobotan kata (*term weighting*). Fungsi *term weighting* ini untuk menggantikan nilai sel VSM yang awalnya adalah jumlah frekuensi kemunculan terms untuk masing-masing dokumen dengan nilai perhitungan term weighting dengan TF-IDF. Proses diawali dengan input matriks yang merepresentasikan dokumen dengan term. Lalu untuk mendapatkan nilai TF-IDF setiap term terhadap dokumen maka dilakukan proses nilai frekuensi kemunculan kata pada suatu dokumen yang disebut dengan *Term Frequency (TF)* dikali dengan *Inverse Document Frequency (IDF)* yang merepresentasikan sedikit tidaknya jumlah dokumen yang memiliki kata tersebut dengan penambahan fungsi log dalam prosesnya. TF-IDF tidak hanya melihat dari jumlah kemunculan setiap kata saja tetapi juga pengaruh pentingnya kata tersebut dalam suatu korpus. TF-IDF mengkombinasikan kemunculan kata dan juga *Inverse Document Frequency* untuk menghasilkan nilai pembobotan suatu kata pada suatu dokumen.

D. Penerapan Algoritma Klaster

Setelah melewati proses *preprocessing* umum dan fungsi tambahan, selanjutnya diklasterisasi dengan algoritma klaster.

1. K-Means

Algoritma K-Means *clustering* adalah algoritma *partional clustering* berdasarkan analisis probabilistik [17]. K-Means merupakan *exclusive clustering* yang berarti objek hanya dapat masuk ke dalam satu kategori klaster. Dengan kata lain K-Means *cluster* tidak mengizinkan *overlapping clustering* pada satu objek. Proses diawali dengan input *K* yang merepresentasikan jumlah klaster yang akan dibangun. Lalu mencari nilai *centroid*/titik tengah tiap klaster secara random. Setelah itu dokumen dikelompokkan ke dalam klaster yang memiliki jarak terdekat dengan titik tengah klaster. Jika ada perubahan data pada klaster maka nilai titik tengah dicari kembali dan proses pengecekan jarak dokumen dan titik tengah klaster dilakukan kembali. Jika tidak ada perubahan data lagi pada klaster maka hasil klaster dikeluarkan.

2. K-Means++

Algoritma K-Means++ merupakan pengembangan dari algoritma K-Means yang dipublikasikan oleh David Arthur and Sergei Vassilvitskii. Pada algoritma K-Means, *centroid* awal dari tiap klaster ditentukan secara random sehingga hasil klaster dan lamanya proses klaster tergantung pada *centroid* awal yang kemungkinan bagus atau malah tidak bagus bergantung hasil randomnya [19]. Untuk itu pada algoritma K-Means++ proses penentuan *centroid* awal dilakukan dengan seksama yaitu dengan menghitung nilai probabilitas yang sebanding dengan jarak kuadrat terdekat dari *centroid* yang ada. K-Means++ menerima inputan dokumen dalam bentuk vektor dan K jumlah klaster dan

mengeluarkan output klaster yang dihasilkan.

3. Agglomerative

Algoritma *agglomerative* merupakan algoritma *hirearchy clustering*. *Agglomerative* merupakan proses *bottom-up* yang menganggap setiap dokumen merupakan klaster tunggal pada awal proses dan kemudian menggabungkan sepasang-sepasang klaster secara berturut-turut sampai semua kelompok telah digabungkan menjadi satu klaster yang berisi seluruh dokumen. *Bottom-up* pada algoritma *hirearchy clustering* disebut *agglomerative hirearchy clustering* [10]. Pengelompokkan *agglomerative* biasanya divisualisasikan sebagai *dendogram*. Setiap penggabungan klaster diwakili oleh garis horizontal. Koordinat y dari garis horizontal adalah kesamaan dari dua buah klaster yang digabungkan, dimana dokumen dipandang sebagai klaster tunggal. Asumsi mendasar dalam *agglomerative* adalah bahwa operasi penggabungan bersifat monoton. Monotonik berarti jika S_1, S_2, …, S_{K-1} adalah kombinasi persamaan dari penggabungan berturut-turut dari *agglomerative*, maka berlaku $S_1 \geq S_2 \geq ... \geq S_{K-1}$ [10]. Proses diawali dengan memasukkan nilai *K* yang merepresentasikan jumlah klaster yang akan dibangun. Pada algoritma ini setiap dokumen di awal proses merupakan klaster-klaster yang berbeda. Lalu dibentuk suatu tabel yang berisi jarak antar tiap klaster. Proses akan berjalan dengan mencari dua vektor yang saling berdekatan lalu menggabungkannya hingga didapat jumlah klaster yang diinginkan. Terdapat beberapa metode klaster *agglomerative* yaitu *Agglomerative Single Link*, *Agglomerative Complete Link*, dan *Agglomerative Average Link*. Penelitian yang dilakukan oleh [24] menyatakan bahwa metode *agglomerative average* yang paling cocok untuk tujuan klasterisasi dokumen. Untuk itu pada penelitian ini kami menggunakan metode *Agglomerative Average Link*. Algoritma *agglomerative* membuat tabel jarak antar dokumen vektor dan menggabungkan dokumen yang berdekatan untuk menghasilkan output klaster sesuai nilai K.

E. Evaluasi Akurasi Klaster

Pengujian dilakukan terhadap metode TF-IDF, LSA dan algoritma K-Means, K-Means++ dan *agglomerative* dengan melihat keberhasilan sistem melakukan proses pengklasteran terhadap dokumen berbahasa Indonesia. Parameter yang digunakan dalam pengujian ini adalah *purity* klaster. Implementasi ini akan memenuhi syarat kesesuaian klaster apabila hasil klaster menghasilkan nilai klaster yang sama dengan klaster aslinya. Purity adalah perhitungan jumlah objek terbanyak dari setiap klaster dibagi dengan jumlah dokumen. Semakin besar nilai *purity* maka akurasi semakin baik. Rumus untuk menghitung *purity* seperti pada Persamaan (1).

$$Purity = \frac{Jumlah\ N\ terbanyak\ tiap\ klaster}{Jumlah\ Dokumen}. \quad (1)$$

Pada penelitian ini, perhitungan *purity* dilakukan untuk semua tahapan proses yang diujicobakan. *Purity* untuk masing-masing algoritma klaster dengan berbagai fungsi tambahan *preprocessing*. Perhitungan *purity* juga dilakukan untuk nilai K yang berbeda yaitu yaitu 2 klaster, 3 klaster, dan 4 klaster. Jumlah topik klaster sebagai *dataset* sesuai dengan jumlah K yang diujicobakan. Misalnya untuk nilai K = 2 maka kami mengambil 2 kumpulan topik saja dari korpus contohnya olahraga dan komputer atau olahraga dan mata. Untuk pengujian K = 3 maka *dataset* yang diambil dari 3 topik dan untuk K = 4 maka *dataset* yang diuji terdiri dari 4 kumpulan topik. Setiap eksperimen dilakukan sebanyak 3 kali percobaan dengan kombinasi topik yang berbeda, yang selanjutnya diambil nilai rata-rata untuk perhitungan *purity*-nya.

III. HASIL DAN PEMBAHASAN

Hasil kalkulasi *purity* yang didapat berdasarkan metode klaster adalah sebagai berikut.

A. K-Means

Terlihat pada saat pengklasteran 2 dokumen reduksi 10%–80% menghasilkan nilai *purity* yang tinggi yaitu 0.70. Lalu pada pengklasteran 3 dokumen reduksi 50%–70% menghasilkan nilai *purity* yang baik yaitu 0,6. Selanjutnya pada pengklasteran 4 dokumen reduksi 50%–70% menghasilkan nilai *purity* yang baik yaitu 0,65.

Pada Tabel 2 terlihat nilai *purity* hasil pengklasteran dengan menggunakan pembobotan TF-IDF dan LSA, dan pengklasteran dengan K-Means. Untuk pengujian Algoritma LSA sendiri kita melakukan proses untuk nilai persentasi reduksi yang berbeda-beda dari 10%–90%. Terlihat pada saat pengklasteran 2 dokumen reduksi 10%–90% menghasilkan nilai *purity* yang paling tinggi yaitu 1,00. Lalu pada pengklasteran 3 dokumen reduksi 10%–90% menghasilkan nilai *purity* yang baik yaitu 0,73. Selanjutnya pada pengklasteran 4 dokumen reduksi 10%–80% menghasilkan nilai *purity* yang baik yaitu 0,70.

Dari Tabel 3 dan juga grafik pada Gambar 2 terlihat hasil *purity* untuk pengklasteran 2–4 klaster dokumen

Tabel 1. Hasil *purity* dengan Algoritma LSA dan Algoritma K-Means

Jumlah Klaster	Jumlah Reduksi								
	10%	20%	30%	40%	50%	60%	70%	80%	90%
2	0,7	0,7	0,7	0,7	0,7	0,7	0,7	0,7	0,5
3	0,53	0,53	0,53	0,53	0,6	0,6	0,6	0,53	0,53
4	0,55	0,55	0,55	0,55	0,65	0,65	0,65	0,55	0,55

Tabel 2. Hasil *purity* dengan TF-IDF, LSA dan K-Means

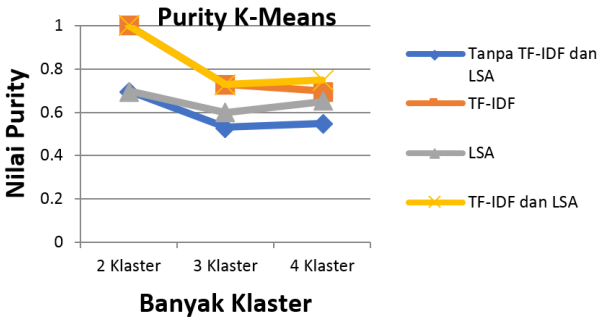
Jumlah Klaster	Jumlah Reduksi								
	10%	20%	30%	40%	50%	60%	70%	80%	90%
2	1	1	1	1	1	1	1	1	1
3	0,73	0,73	0,73	0,73	0,73	0,73	0,73	0,73	0,73
4	0,7	0,7	0,7	0,7	0,75	0,7	0,7	0,7	0,65

Tabel 3. Hasil *purity* dengan Algoritma K-Means untuk empat proses pembobotan yang berbeda

Jumlah Klaster	Tanpa Pembobotan	TF-IDF	LSA	TF-IDF dan LSA
2	0,7	1	0,7	1
3	0,53	0,73	0,6	0,73
4	0,55	0,7	0,65	0,75

Tabel 4. Hasil *purity* dengan Algoritma LSA dan Algoritma K-Means++

Jumlah Klaster	Jumlah Reduksi								
	10%	20%	30%	40%	50%	60%	70%	80%	90%
2	0,7	0,7	0,7	0,7	0,7	0,7	0,7	0,7	0,5
3	0,53	0,53	0,53	0,53	0,6	0,6	0,53	0,53	0,53
4	0,55	0,55	0,65	0,65	0,65	0,65	0,65	0,55	0,55



Gambar 2. Grafik nilai *purity* dengan Algoritma K-Means untuk empat proses *preprocessing* yang berbeda

menggunakan algoritma K-Means dan dengan 4 metode pembobotan yang berbeda. Terlihat bahwa semakin banyak klaster yang akan dibentuk maka nilai *purity* klaster akan semakin menurun yang berarti nilai kesesuaian klaster juga dipengaruhi oleh banyaknya klaster yang akan dibentuk. Semakin banyak klaster yang akan dibentuk maka kemungkinan kesesuaian suatu dokumen terhadap klasternya semakin rendah. Lalu terlihat juga pengaruh proses pembobotan sebelum dokumen vektor diklaster. Dari grafik pada Gambar 2 terlihat bahwa dokumen vektor yang melalui proses TF-IDF dan LSA sebelum diklaster akan membuat nilai *purity* klaster yang lebih baik daripada dokumen vektor yang tanpa melalui proses pembobotan sebelumnya.

B. K-Means++

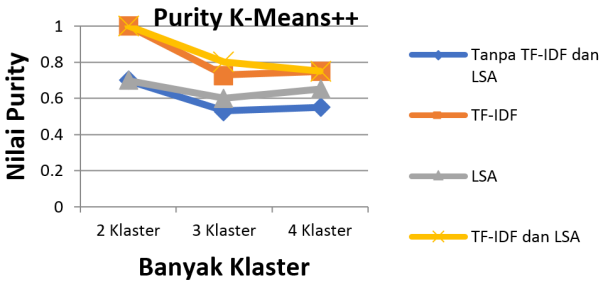
Proses yang sama diterapkan untuk algoritma K-Means++. Hasil dari *purity* ditunjukkan pada Tabel 4. Pada Tabel ini terlihat nilai *purity* hasil pengklasteran dengan menggunakan pembobotan LSA dan pengklasteran dengan K-Means++. Terlihat pada saat pengklasteran 2 dokumen reduksi 10%–80% menghasilkan nilai *purity* yang paling tinggi yaitu 0,70. Lalu pada pengklasteran 3 dokumen reduksi 50%–60% menghasilkan nilai *purity* yang baik yaitu 0,60. Selanjutnya pada pengklasteran 4 dokumen reduksi 30%–70% menghasilkan nilai *purity* yang baik yaitu 0,65.

Tabel 5. Hasil *purity* dengan TF-IDF, LSA, dan K-Means++

Jumlah Klaster	Jumlah Reduksi								
	10%	20%	30%	40%	50%	60%	70%	80%	90%
2	1	1	1	1	1	1	1	1	1
3	0,73	0,73	0,73	0,73	0,8	0,73	0,73	0,73	0,73
4	0,75	0,75	0,75	0,75	0,75	0,75	0,75	0,75	0,45

Tabel 6. Hasil *purity* dengan algoritmaTF-IDF, LSA dan K-Means++ untuk empat proses pembobotan yang berbeda

Jumlah Klaster	Tanpa Pembobotan	TF-IDF	LSA	TF-IDF dan LSA
2	0,7	1	0,7	1
3	0,53	0,73	0,6	0,8
4	0,55	0,75	0,65	0,75



Gambar 3. Grafik nilai *purity* dengan Algoritma K-Means untuk empat proses pembobotan yang berbeda

Pada Tabel 5 terlihat nilai *purity* hasil pengklasteran dengan menggunakan pembobotan TF-IDF dan LSA, dan pengklasteran dengan K-Means++. Terlihat pada saat pengklasteran 2 dokumen reduksi 10%–90% menghasilkan nilai *purity* yang paling tinggi yaitu 1. Lalu pada pengklasteran 3 dokumen reduksi 50% menghasilkan nilai *purity* yang baik yaitu 0,80. Selanjutnya pada pengklasteran 4 dokumen reduksi 10%–80% menghasilkan nilai *purity* yang baik yaitu 0,75.

Dari Tabel 6 dan juga grafik pada Gambar 3 terlihat hasil *purity* untuk pengklasteran 2–4 klaster dokumen dengan algoritma K-Means++ dan dengan 4 metode pembobotan yang berbeda. Terlihat bahwa semakin banyak klaster yang akan dibentuk maka nilai *purity* klaster akan semakin menurun. Lalu terlihat juga pengaruh proses pembobotan sebelum dokumen vektor diklaster. Dari grafik pada Gambar 3 terlihat bahwa dokumen vektor yang melalui proses TF-IDF dan LSA sebelum diklaster akan membuat nilai purity klaster yang lebih baik daripada dokumen vektor yang tanpa melalui proses pembobotan sebelumnya. Jika dibandingkan dengan hasil Purity untuk K-Means dengan proses pembobotan dengan TF-IDF dan LSA terlihat nilai *purity* dengan algoritma K-Means++ lebih tinggi.

Tabel 7. Hasil *purity* dengan Algoritma LSA dan Algoritma *Agglomerative Average Link*

Jumlah Klaster	Jumlah Reduksi								
	10%	20%	30%	40%	50%	60%	70%	80%	90%
2	0,7	0,7	0,7	0,7	0,7	0,7	0,7	0,7	0,7
3	0,53	0,53	0,53	0,53	0,53	0,53	0,53	0,53	0,53
4	0,55	0,55	0,5	0,5	0,5	0,5	0,5	0,55	0,5

Tabel 8. Hasil *purity* dengan metode TF-IDF, Algoritma LSA dan *Algoritma Agglomerative Average Link*

Jumlah Klaster	Jumlah Reduksi								
	10%	20%	30%	40%	50%	60%	70%	80%	90%
2	1	1	1	1	1	1	1	1	1
3	0,6	0,6	0,6	0,6	0,53	0,53	0,53	0,47	0,47
4	0,5	0,5	0,5	0,5	0,5	0,4	0,4	0,65	0,65

Tabel 9. Hasil *purity* dengan Algoritma *Agglomerative Average Link* untuk empat proses pembobotan yang berbeda

Jumlah Klaster	Tanpa Pembobotan	TF-IDF	LSA	TF-IDF dan LSA
2	0,7	1	0,7	1
3	0,53	0,6	0,53	0,6
4	0,5	0,5	0,55	0,65

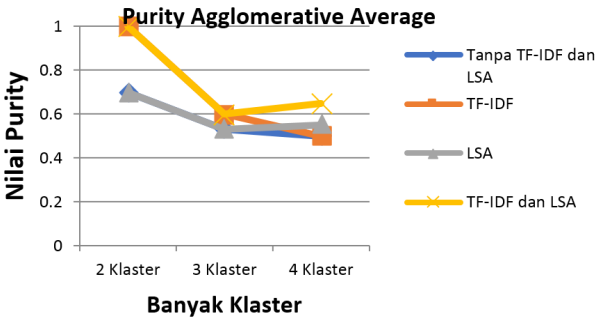
C. *Agglomerative*

Pada bagian ini akan dijabarkan nilai *purity* dari hasil klastering dokumen yang terdiri dari 2 klaster sampai 4 klaster menggunakan algoritma *Agglomerative Average Link*. Dimana sebelum diklaster vektor dokumen di lakukan dahulu proses pembobotan dengan 4 cara yang berbeda, yaitu tanpa proses pembobotan, pembobotan dengan TF-IDF, pembobotan dengan LSA, dan pembobotan dengan TF-IDF dilanjutkan dengan LSA.

Pada Tabel 7 terlihat nilai *purity* hasil pengklasteran dengan menggunakan pembobotan LSA dan pengklasteran dengan *Agglomerative Average Link*. Terlihat pada saat pengklasteran 2 dokumen dari reduksi 10%–90% menghasilkan nilai *purity* yang sama yaitu 0,70. Lalu pada pengklasteran 3 dokumen reduksi 10%–90% juga menghasilkan nilai Purity yang sama yaitu 0,53. Selanjutnya pada pengklasteran 4 dokumen reduksi 80% menghasilkan nilai *purity* yang baik yaitu 0,55.

Pada Tabel 8 terlihat nilai Purity hasil pengklasteran dengan menggunakan pembobotan TF-IDF dan LSA, dan pengklasteran dengan *Agglomerative Average Link*. Terlihat pada saat pengklasteran 2 dokumen reduksi 10%–90% menghasilkan nilai *purity* yang baik yaitu 1.00. Lalu pada pengklasteran 3 dokumen reduksi 10%–40% menghasilkan nilai *purity* yang paling baik yaitu 0,6. Selanjutnya pada pengklasteran 4 dokumen reduksi 80%–90% menghasilkan nilai *purity* yang baik yaitu 0,65.

Dari Tabel 9 dan juga grafik pada Gambar 4 terlihat hasil *purity* untuk pengklasteran 2–4 klaster dokumen



Gambar 4. Grafik nilai *purity* dengan Algoritma *Agglomerative Average Link* untuk empat proses pembobotan yang berbeda

dengan algoritma *Agglomerative Average Link* dan dengan 4 metode pembobotan yang berbeda. Terlihat bahwa semakin banyak klaster yang akan dibentuk maka nilai *purity* klaster akan semakin menurun walaupun dari grafik pada Gambar 4 pada hasil 4 klaster nilai *purity* kembali naik. Lalu terlihat juga pengaruh proses pembobotan sebelum dokumen vektor diklaster. Dari grafik pada Gambar 4 terlihat bahwa dokumen vektor yang melalui proses TF-IDF dan LSA sebelum diklaster akan membuat nilai *purity* klaster yang lebih baik daripada dokumen vektor yang tanpa melalui proses pembobotan sebelumnya.

IV. KESIMPULAN

Berdasarkan analisis, perancangan, dan hasil pengujian implementasi metode TF-IDF dan Algoritma LSA untuk pembobotan serta *clustering* dengan Algoritma K-Means, Algoritma K-Means++ dan Algoritma Agglomerative dapat disimpulkan bahwa kisaran nilai persentasi reduksi untuk LSA yang baik adalah antara 50%-80%. Pembobotan dengan metode TF-IDF dan algoritma LSA membuat nilai purity hasil klaster lebih tinggi dibandingkan dengan tidak dilakukan proses pembobotan pada dokumen vektor. Algoritma K-Means dan K-Means++ menghasilkan nilai purity yang lebih baik dibanding algoritma Agglomerative. Algoritma K-Means++ sedikit lebih baik dibanding algoritma K-Means karena algoritma K-Means++ merupakan pengembangan dari algoritma K-Means. Kesimpulan lain yang dapat diambil yaitu semakin banyak klaster yang akan dibentuk maka nilai purity klaster akan semakin menurun yang berarti nilai kesesuaian klaster juga dipengaruhi oleh banyaknya klaster yang akan dibentuk. Semakin banyak klaster yang akan dibentuk maka kemungkinan kesesuaian suatu dokumen terhadap klasternya semakin rendah.

UCAPAN TERIMA KASIH

Penelitian ini didukung oleh Lembaga Penelitian Universitas Sumatera Utara untuk pendanaan TALENTA USU Tahun 2017 No Kontrak 5338/UN5.1.R/PPM/2017 Tanggal Mei 22 May 2017

REFERENSI

[1] G. Salton, A. Wong, and C. S. Yang, “A vector space model for

automatic indexing," *Commun. ACM*, vol. 18, no. 11, pp. 613–620, 1975.

[2] R. Ju, P. Zhou, C. H. Li, and L. Liu, "An Efficient Method for Document Categorization Based on Word2vec and Latent Semantic Analysis," in *2015 IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing*, 2015, pp. 2276–2283.

[3] R. Jensi and D. G. W. Jiji, "A Survey on optimization approaches to text document clustering," *Int. J. Comput. Sci. Appl.*, vol. 3, no. 6, pp. 31–44, 2013.

[4] D. C. Anastasiu, A. Tagarelli, and G. Karipys, "Data Clustering : The Next Frontier," in *Data Clustering : Algorithms and Applications*, C. C. Aggarwal and C. K. Reddy, Eds. Minnesota: CRC Press, 2013, pp. 305–338.

[5] F. Z. Tala, "A Study of Stemming Effects on Information Retrieval in Bahasa Indonesia."

[6] A. Salem, J. Tang, and H. Liu, "Feature Selection For Clustering : A Review," in *Data Clustering : Algorithms and Applications*, CRC Press, 2013, p. 29.

[7] P. Wiemer-Hastings, "Latent Semantic Analysis," *Encycl. Lang. Linguist.*, pp. 1–8, 2004.

[8] N. E. Evangelopoulos, "Latent semantic analysis," *Wiley Interdiscip. Rev. Cogn. Sci.*, vol. 4, no. 6, pp. 683–692, 2013.

[9] T. Landauer, P. Foltz, and D. Laham, "An introduction to latent semantic analysis," *Discourse Process.*, vol. 25, no. 2–3, pp. 259–284, 1998.

[10] C. D. Manning, P. Raghavan, and H. Schutze, *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[11] T. F. Abidin, B. Yusuf, and M. Umran, "Singular Value Decomposition for dimensionality reduction in unsupervised text learning problems," in *2010 2nd International Conference on Education Technology and Computer*, 2010, pp. V4-422-V4-426.

[12] A. Amalia, M. S. Lydia, S. D. Fadilla, M. Huda, and D. Gunawan, "Document Clustering Optimization with Synonym Dictionary Check Function Case Study : Documents in Bahasa Indonesia."

[13] Y. Li, S. M. Chung, and J. D. Holt, "Text document clustering based on frequent word meaning sequences," *Data Knowl. Eng.*, vol. 64, no. 1, pp. 381–404, 2008.

[14] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data clustering: a review," *ACM Comput. Surv.*, vol. 31, no. 3, pp. 264–323, 1999.

[15] J.~Bellegarda, J.~Butzberger, Y.~Chow, N.~Coccaro, and D.~Naik, "A novel word clustering algorithm based on latent semantic analysis," *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, pp. 172–175, 1996.

[16] A. Huang, "Similarity measures for text document clustering," *Proc. Sixth New Zeal.*, no. April, pp. 49–56, 2008.

[17] A. K. Jain, "Data clustering: 50 years beyond K-means," *Pattern Recognit. Lett.*, vol. 31, no. 8, pp. 651–666, 2010.

[18] M. Umran *et al.*, "Pengelompokkan Dokumen Menggunakan K-Means Dan Singular Value Decomposition : Studi Kasus," pp. 1–5, 2009.

[19] D. Arthur and S. Vassilvitskii, "k-means++: the advantages of careful seeding," in *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, 2007, pp. 1027–1025.

[20] O. Bachem, M. Lucic, S. Hamed Hassani, and A. Krause, "Approximate K-Means ++ in Sublinear Time," *Proc. 30th Conf. Artif. Intell. (AAAI 2016)*, no. 2007, pp. 1459–1467, 2016.

[21] M. Steinbach, G. Karypis, and V. Kumar, "A Comparison of Document Clustering Techniques," *KDD Work. text Min.*, vol. 400, no. X, pp. 1–2, 2000.

[22] [22]B. Aubaidan, M. Mohd, M. Albared, and F. Author, "Comparative study of k-means and k-means++ clustering algorithms on crime domain," *J. Comput. Sci.*, vol. 10, no. 7, pp. 1197–1206, 2014.

[23] M. Adriani, J. Asian, B. Nazief, and H. E. Williams, "Stemming Indonesian : A Confi x-Stripping Approach," *ACM Trans. Asian Lang. Inf. Process.*, vol. 6, no. 4, pp. 1–33, 2007.

[24] A. El-Hamdouchi and P. Willett, "Comparison of Hierachic Agglomerative Clustering Methods for Document Retrieval," *Comput. J.*, vol. 32, no. 3, pp. 220–227, 1989.

Penerbit:
Jurusan Teknik Elektro, Fakultas Teknik, Universitas Syiah Kuala
Jl. Tgk. Syech Abdurrauf No. 7, Banda Aceh 23111
website: http://jurnal.unsyiah.ac.id/JRE
email: rekayasa.elektrika@unsyiah.net
Telp/Fax: (0651) 7554336