Implementasi *KD-Tree K-Means Clustering* untuk Klasterisasi Dokumen

Eric Budiman Gosno, Isye Arieshanti, dan Rully Soelaiman
Jurusan Teknik Informatika, Fakultas Teknologi Informasi, Institut Teknologi Sepuluh Nopember (ITS)
Jl. Arief Rahman Hakim, Surabaya 60111 Indonesia
e-mail: i.arieshanti@if.its.ac.id

***Abstrak*—Klasterisasi dokumen adalah suatu proses pengelompokan dokumen secara otomatis dan unsupervised. Klasterisasi dokumen merupakan permasalahan yang sering ditemui dalam berbagai bidang seperti *text mining* dan sistem temu kembali informasi. Metode klasterisasi dokumen yang memiliki akurasi dan efisiensi waktu yang tinggi sangat diperlukan untuk meningkatkan hasil pada mesin pencari web, dan untuk proses *filtering*. Salah satu metode klasterisasi yang telah dikenal dan diaplikasikan dalam klasterisasi dokumen adalah *K-Means Clustering*. Tetapi *K-Means Clustering* sensitif terhadap pemilihan posisi awal dari titik tengah klaster sehingga pemilihan posisi awal dari titik tengah klaster yang buruk akan mengakibatkan *K-Means Clustering* terjebak dalam *local optimum*. *KD-Tree K-Means Clustering* merupakan perbaikan dari *K-Means Clustering*. *KD-Tree K-Means Clustering* menggunakan struktur data *K-Dimensional Tree* dan nilai kerapatan pada proses inisialisasi titik tengah klaster. Pada makalah ini diimplementasikan algoritma *KD-Tree K-Means Clustering* untuk permasalahan klasterisasi dokumen. Performa klasterisasi dokumen yang dihasilkan oleh metode *KD-Tree K-Means Clustering* pada data set *20 newsgroup* memiliki nilai distorsi 3×10^5 lebih rendah dibandingkan dengan nilai rerata distorsi *K-Means Clustering* dan nilai NIG 0,09 lebih baik dibandingkan dengan nilai NIG *K-Means Clustering*.**

***Kata Kunci*—*K-Dimensional Tree*, *K-Means Clustering*, *KD-Tree K-Means Clustering*, Klasterisasi Dokumen**

I. PENDAHULUAN

KLASTERISASI dokumen adalah suatu proses pengelompokan dokumen secara otomatis dan *unsupervised*. Klasterisasi dokumen merupakan permasalahan yang sering ditemui dalam berbagai bidang seperti *text mining* dan sistem temu kembali informasi. Metode klasterisasi dokumen yang memiliki akurasi dan efisiensi waktu yang tinggi sangat diperlukan untuk meningkatkan hasil pada mesin pencari web, dan untuk proses *filtering*.

Klasterisasi dokumen merupakan proses untuk membagi sekumpulan dokumen ke dalam beberapa kelompok spesifik yang disebut sebagai klaster. Setiap klaster memiliki kumpulan dokumen yang memilki nilai kemiripan antar dokumen yang tinggi dan antar satu klaster dengan klaster yang lain memiliki dokumen dengan nilai kemiripan yang minimal.

Secara umum, permasalahan klasterisasi dapat dibagi menjadi 2 jenis yaitu *Hierarchial Clustering* dan *Partitional Clustering*. *Partitional Clustering* melakukan partisi sebuah data set menjadi sejumlah klaster dengan memaksimalkan nilai kemiripan antar data-data pada klaster yang sama. Mencari hasil partisi yang menghasilkan nilai optimal pada *Partitional Clustering* merupakan permasalahan *NP-Complete*. Tetapi terdapat beberapa strategi suboptimal untuk menyelesaikan permasalahan *partitional clustering*. Salah satu strategi suboptimal *partitional clustering* yang telah dikenal dan diaplikasikan dalam klasterisasi dokumen adalah *K-Means Clustering*.

K-Means Clustering merupakan metode pengoptimalan lokal yang sensitif terhadap pemilihan posisi awal dari titik tengah klaster. Sehingga pemilihan posisi awal dari titik tengah klaster yang buruk akan mengakibatkan algoritma *K-Means Clustering* terjebak dalam solusi lokal optimal [1]. *KD-Tree K-Means Clustering* [2] merupakan perbaikan dari algoritma *K-Means Clustering*. *KD-Tree K-Means Clustering* menggunakan struktur data *K-Dimensional Tree* dan nilai kerapatan pada proses inisialisasi titik tengah klaster. Tetapi, hasil evaluasi oleh *Redmond et al* [2] tidak melingkupi performa *KD-Tree K-Means Clustering* pada data set dokumen.

Makalah ini bertujuan untuk melakukan analisa terhadap proses klasterisasi dokumen menggunakan algoritma *KD-Tree K-Means Clustering*. Proses klasterisasi dokumen tersebut akan diuji menggunakan data set *20 newsgroup* dari *KDD UCI Archive* [3]. Diharapkan penggunaan *KD-Tree K-Means Clustering* dapat meningkatkan hasil dan performa dari klasterisasi dokumen dibandingkan dengan metode *K-Means Clustering* dasar yang menggunakan metode *Forgy's* [4].

Makalah ini terorganisasi menjadi 5 bab. Bab I menguraikan latar belakang serta tujuan penulisan makalah. Bab II menguraikan dasar teori yang digunakan untuk implementasi sistem. Bab III menguraiakan perancangan sistem klasterisasi dokumen. Hasil uji coba diuraikan pada Bab IV. Bab V merupakan kesimpulan yang dapat diambil dari makalah ini.

II. DASAR TEORI

A. Algoritma K-Means Clustering

Algoritma *K-Means Clustering* merupakan salah satu metode dari klasterisasi yang bertujuan untuk melakukan partisi dari sebuah data set ke dalam *K* klaster. Dalam prosesnya, *K-Means Clustering* akan mencari titik tengah dari semua klaster yang menghasilkan total nilai jarak dari setiap data observasi ke titik tengah klaster minimal [2]. Alur metode dari algoritma *K-Means Clustering* dapat dilihat pada Gambar 1.

Algoritma *K-Means Clustering* dengan nilai *K* tertentu secara umum memiliki kompleksitas *NP-Hard* [5]. Tetapi jika

Algoritma *K-Means Clustering*

1. Tentukan jumlah klaster K, dan jumlah iterasi maksimum.
2. Lakukan proses inisialisasi K titik tengah klaster.
3. Hubungkan setiap data observasi ke klaster terdekat
4. Kalkulasi ulang posisi titik tengah klaster.
5. Hitung nilai distorsi $D = \sum_{i=1}^{n} \left[\min_{j = (1 \ldots K)} d(x_i, c_j) \right]^2$
6. Apabila ada perubahan posisi titik tengah klaster atau jumlah iterasi < jumlah iterasi maksimum, kembali ke langkah 3. Jika tidak, maka kembalikan hasil klasterisasi.

Gambar 1. Alur algoritma *K-Means Clustering*

Fungsi kdtree (list point, *depth*)

1. Pilih Atribut/*axis* yang digunakan sebagai pembatas ($axis = depth \bmod k$) atau ($axis =$ longest dimension).
2. Urutkan data berdasarkan *axis* yang dipilih dan tentukan nilai *median* atau *mean* sebagai *pivot value.*
3. Buat node dan konstruksi *subtress* dengan melakukan fungsi rekursif:
   node.leftChild := kdtree(poin sebelum pivot value, depth+1);
   node.rightChild := kdtree(poin setelah pivot value, depth+1);

Gambar 2. Alur fungsi *Build KD-Tree*

diberikan nilai Distorsi yang diharapkan, maka Algoritma K-Means Clustering dapat diselesaikan dengan kompleksitas O(ndk+1 log n) di mana n menyatakan banyaknya data observasi [6].

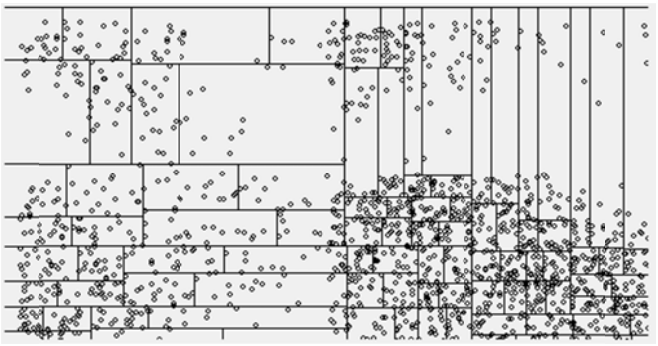
B. K-Dimensional Tree

K-Dimensional Tree (*KD-Tree*) [7] adalah data struktur yang bersifat *space-partitioning*, dan merupakan kasus spesial dari *binary space partitioning tree*. *KD-Tree* bertujuan untuk mengatur poin-poin dalam *k–dimensional space*. *KD-Tree* umumnya diaplikasikan dalam pencarian yang memiliki banyak kunci pencarian (*multidimensional key*) seperti *range search*, dan *nearest neighbor search* [8].

Dalam implementasi, *KD-Tree* adalah *binary tree* dimana setiap *node* pada *binary tree* tersebut adalah sebuah poin berdimensi *k*. Setiap *node* yang bukan merupakan *leaf* pada *KD-Tree* akan menghasilkan sebuah *hyperplane* yang memisakan sebuah ruang menjadi 2 bagian. Setiap poin yang berada di daerah sebelah kiri *hyperplane* merepresentasikan *node* yang berada di *subtree* sebelah kiri. Demikian pula dengan setiap poin yang berada di daerah sebelah kanan *hyperplane* akan merepresentasikan *node* yang berada di *subtree* sebelah kanan [8]. Alur metode dari fungsi pembentukan *KD-Tree* dapat dilihat pada Gambar 2. Contoh hasil dari *2-dimensional tree* dapat dilihat pada Gambar 3.

Untuk menentukan dimensi yang menjadi pemisah pada *K-Dimensional Tree* terdapat 2 metode. Metode yang pertama adalah menggunakan modulo dari nilai kedalaman *node* [8]. Pada metode ini semua *node* pada kedalaman yang sama akan memiliki dimensi pemisah yang sama. Cara yang kedua adalah menggunakan rentang dimensi. Pada setiap *node* akan dicari dimensi yang memiliki rentang terpanjang. Dimensi dengan rentang terpanjang tersebut akan digunakan sebagai dimensi pemisah pada *node* tersebut [2].

Secara umum, kompleksitas waktu untuk melakukan konstruksi *KD-Tree* memenuhi (1) yang memilki kompleksitas *amortized* O($n \log n$) hingga O($n \log^2 n$) [8].



Gambar 3. Contoh hasil *space partitioning* 2-*dimensional tree*

$$Tn = \begin{cases} O(1) & jika\ n = 1, \\ O(n) + 2T\left(\left\lceil \frac{n}{2} \right\rceil\right) & jika\ n > 1, \end{cases} \quad (1)$$

Karena *KD-Tree* merupakan *binary tree*, dan setiap *leaf* dan *internal node* menggunakan kapasitas O(1), maka dapat disimpulkan bahwa total kapasitas yang diperlukan *KD-Tree* sebanyak O(n) [8].

C. KD-Tree K-Means Clustering

KD-Tree K-Means Clustering [2] merupakan algoritma pengoptimalan dari algoritma *KKZ/Katsoudivinis* [9] yang menggunakan nilai kerapatan dan jarak antar poin/data dalam menentukan posisi awal titik tengah klaster *K-Means Clustering*. *KD-Tree K-Means Clustering* menggunakan struktur data *K-Dimensional Tree* dalam proses inisialisasi titik tengah klaster. *KD-Tree K-Means Clustering* juga merupakan metode yang mampu menangani data set yang memiliki *noise/outlier* dengan cara menghapus 20% calon titik tengah klaster dengan nilai kerapatan terendah [2].

Proses inisialisasi titik tengah klaster pada *KD-Tree K-Means Clustering* dimulai dengan pembentukan *KD-Tree* dengan menetapkan bahwa *leaf bucket* dari *KD-Tree* memiliki maksimal 20 data. Selanjutnya akan dikalkulasi nilai kerapatan dari setiap *leaf bucket* dengan formula $p_j = \frac{N_j}{V_j}$ di mana N_j menyatakan banyak data pada *leaf bucket j,*dan V_j menyatakan volume dari *leaf bucket j*. Pada implementasi dari *Redmond et al* [2] nilai volume V_j dapat dinyatakan dengan hasil perkalian dari semua rentang dimensi dari poin-poin pada *leaf bucket.* Ketika terdapat dimensi dengan nilai rentang nol. Maka nilai rentang dimensi tersebut akan digantikan dengan nilai *geometric mean* dari nilai rentang dimensi yang tidak bernilai nol [2].

Posisi dari setiap *leaf bucket* akan diindentifikasi sebagai nilai rerata dari posisi data-data yang berada di dalam *leaf bucket* tersebut. Sehingga untuk *leaf bucket* sebanyak q akan didapatkan sekumpulan point *leaf bucket* ($m_1,m_2,..,m_q$) pada *space* beserta dengan nilai kerapatan dari setiap poin ($p_1,p_2,...,p_q$) [2].

Untuk memilih titik tengah klaster awal pada proses *K-Means Clustering*, akan digunakan informasi *leaf bucket* yang telah didapatkan dari proses sebelumnya. Mula-mula, semua *leaf bucket* akan diurutkan berdasarkan nilai kerapatan dari yang terbesar hingga yang terkecil. *Leaf bucket* yang memiliki

Algoritma *KD-Tree K-Means Clustering*

1. Pembentukan *K-Dimensional Tree* dari data set. *K-Dimensional Tree* yang dibuat memiliki *leaf bucket* dengan jumlah data maksimal 20 per *leaf bucket*.
2. Untuk setiap *leaf bucket*($L_1,L_2,...,L_j$) , kalkulasi nilai kerapatan(P_j) dari setiap *leaf bucket* L_j, dan kalkulasi nilai titik tengah *leaf bucket*(M_j) dengan mencari nilai rerata dari semua point yang ada pada *leaf bucket* L_j.
3. Pilih titik tengah klaster pertama $C_1 = M_z$, dimana $z = \arg\max(P_j)$.
4. Untuk $t = 2,\ldots,K$:
   - Untuk $j = 1,...,q$ kalkulasi nilai *ranking leaf bucket* (G_j) dengan fomula :
     $G_j = \left\{ \min_{k=1\ldots t} [d(C_k, M_j)] \right\} \times P_j$.
   - $C_t = M_z$, dimana $z = \arg\max(G_j)$.
5. Hapus 20% *leaf bucket* dengan nilai kerapatan terendah. Kembali ke langkah 3 dan kalkulasi posisi K titik tengah klaster baru ($\hat{c}_1, \hat{c}_2, \ldots, \hat{c}_k$).
6. Jalankan algoritma *K-Means Clustering* dengan nilai inisialisasi titik tengah ($C_1,...,C_k$) dan ($\hat{c}_1, \hat{c}_2, \ldots, \hat{c}_k$).

Gambar 4. Alur Algoritma *KD-Tree K-Means Clustering*

nilai kerapatan tertinggi akan dipilih sebagai titik tengah klaster pertama. Pemilihan titik tengah klaster yang berikutnya akan didasarkan pada nilai jarak antara calon titik tengah/*leaf bucket* dengan titik tengah yang telah ditetapkan, dan nilai kerapatan dari *leaf bucket* tersebut. Semakin jauh jarak *leaf bucket* dari posisi titik tengah yang telah ditetapkan dan semakin besar nilai kerapatan *leaf bucket* tersebut, maka semakin tinggi peluang *leaf bucket* tersebut menjadi titik tengah klaster yang baru [2]. Secara matematis, formula untuk menentukan nilai dari *leaf bucket* dapat dilihat pada (2).

$$\boldsymbol{G_j = \left\{ \min_{k=1\ldots t} [d(C_k, M_j)] \right\} \times P_j} \quad (2)$$

Dengan fungsi *d* menyatakan fungsi jarak seperti *euclidean distance*, C_k menyatakan posisi titik tengah klaster, M_j menyatakan posisi *leaf bucket*, dan P_j menyatakan nilai kerapatan dari *leaf bucket*. Untuk setiap iterasi, *leaf bucket* dengan nilai G_j terbesar akan dipilih sebagai titik tengah klaster baru. Iterasi akan terus dijalankan hingga didapatkan *Kc* titik tengah klaster yang akan digunakan sebagai inisialisasi pada *K-Means Clustering* [2]. Alur kerja dari algoritma *KD-Tree K-Means Clustering* secara keseluruhan dapat dilihat pada Gambar 4.

Untuk mengatasi permasalahan terkait dengan terpilihnya data *outlier* menjadi calon titik tengah, algoritma akan membuang 20% *leaf bucket*/calon titik tengah yang memiliki nilai kerapatan terendah. Dengan hanya menggunakan 80% *leaf bucket* yang memiliki kerapatan tertinggi, maka kemungkinan data *outlier* terpilih menjadi *centroid cluster* dapat diminimalisasi sebelum algoritma dijalankan [2].

Nilai kerapatan P_j dapat diganti dengan *ranking density* ($\hat{P}_j$). *Leaf Bucket* dengan nilai terendah memiliki nilai $\hat{P}_j = 1$ dan *leaf bucket dengan* nilai tertinggi memiliki nilai $\hat{P}_j = n$. Tujuan dari penggunaan *ranking density* adalah untuk mencegah nilai kerapatan yang terlalu dominan dibandingkan dengan jarak *leaf bucket* ke titik tengah [2].

III. PERANCANGAN SISTEM KLASTERISASI DOKUMEN

Perancangan sistem klasterisasi dokumen ini dibagi menjadi 3 tahap yaitu tahap pra proses data set dokumen, tahap pembobotan kata/*term weighting* dan tahap klasterisasi menggunakan *KD-Tree K-Means Clustering*.

A. Tahap Pra Proses Data Set Dokumen

Sebelum dapat diterapkan proses pembelajaran seperti klasterisasi data set dokumen perlu untuk mengalami tahap pra proses terlebih dahulu. Tujuan dari tahap pra proses ini adalah mengubah data set dokumen ke dalam bentuk *bag of words* [10]. *Bag of words* merupakan suatu metode untuk merepresentasikan data set yang berbasis teks seperti dokumen, dan artikel ke dalam suatu matriks. *Bag of words* akan menyimpan informasi frekuensi kemunculan suatu kata dalam suatu dokumen dalam bentuk matriks 2 dimensi. *Bag of words* merupakan representasi dokumen yang bersifat *orderless* dimana *bag of words* akan mengabaikan unsur keterkaitan antara 1 kata dengan kata yang lain serta urutan kemunculan kata pada dokumen. Kalimat seperti *"home made"* akan memiliki nilai yang sama dengan *"made home"* pada *bag of words* [10].

Tahap pra proses akan dimulai dengan proses *stemming* kata, dan penghapusan *stop word*. *Stop Word* adalah kata-kata yang umum dan memiliki nilai yang tidak signifikan dalam proses *machine learning* pada dokumen. Sehingga *stop word* perlu untuk dihilangkan dari kamus *vocabulary* secara keseluruhan. Proses penghapusan *stop word* dapat mereduksi jumlah kata yang perlu disimpan pada proses pembelajaran secara signifikan [12]. Dalam implementasi makalah ini, penulis menggunakan daftar *stop word* yang dapat diunduh pada *repository Journal of Machine Learning Research* [13].

Stemming adalah sebuah metode untuk menyederhanakan suatu kata ke bentuk dasar/bentuk *stem* dari kata tersebut. Hasil kata pada *stemming* tidak harus identik dengan akar dari suatu kata secara morfologika. Tetapi hasil *stemming* cukup untuk menyederhanakan kata-kata yang berkaitan ke dalam bentuk *stem*/dasar yang sama. Algoritma *stemming* seringkali disebut juga sebagai *stemmer*. Sebagai contoh kata *computer*, *computing*, *computation,* dan *computes* akan memiliki bentuk *stem* yang sama yaitu *comput* [13]. Metode *stemming* yang digunakan pada proses ini adalah *porter stemmer* [14] yang telah umum digunakan dalam proses stemming kata berbahasa inggris.

Setelah kata-kata dalam artikel telah diseleksi dan disederhanakan dalam proses *stemming*, maka semua kata pada artikel yang tersisa akan dikumpulkan dan disimpan dalam bentuk *bag of words*. Untuk menghilangkan kata-kata yang berbentuk *noise* dan *typo*, kata yang dimasukkan ke dalam *bag of words* hanyalah kata yang muncul pada minimal 2 dokumen [13].

B. Tahap Pembobotan Kata/Term Weighting

Sebelum melakukan proses klasterisasi, data set akan mengalami proses pembobotan kata/*term weigthing* terlebih

dahulu. Proses pembobotan kata/*term weigthing* adalah sebuah proses untuk memberikan nilai pada sebuah kata. Kata yang memiliki nilai signifikan pada proses *machine learning* akan memiliki nilai bobot yang tinggi. Sebaliknya kata yang tidak memberikan pengaruh signifikan pada proses *machine learning* akan memiliki nilai bobot yang rendah [11].

Untuk proses pembobotan kata akan digunakan perhitungan *term frequency – inverse document frequency* (TF-IDF) [15]. Sebuah kata akan memiliki nilai tf-idf tinggi jika memiliki frekuensi kemunculan yang tinggi namun jumlah dokumen yang memuat kata tersebut dalam *corpus* sedikit [11].

Tf-idf merupakan produk hasil perkalian dari 2 nilai statistic yaitu *term frequency* (TF), dan *inverse document frequency* (IDF) [11]. Secara matematis, (3) menunjukkan formula matematis dari tf-idf, di mana t menyatakan kata yang dicari nilainya, d menyatakan dokumen yang mengandung kata tersebut, dan D menyatakan koleksi dari dokumen secara keseluruhan.

$$tfidf(t,d,D) = tf(t,d)\ x\ idf(t,D) \quad (3)$$

Nilai tf-idf memiliki banyak variasi nilai tergantung pada formula perhitungan dari *term frequency* dan *inverse document frequency*. Pada makalah ini akan digunakan formula perhitungan tf-idf yang menggunakan fungsi logaritma dari *term frequency* dan *inverse document frequency* seperti yang ditunjukkan pada (4) dimana f(t,d) menyatakan frekuensi kata *t* pada dokumen *d,* N menyatakan banyak dokumen pada data set, dan *Dft* menyatakan frekuensi dokumen yang memuat kata *t*.

$$tfidf(t,d,D) = (1 + \log(f(t,d))) \times\ log\ \frac{N}{Dft} \quad (4)$$

C. Tahap Klasterisasi Dokumen

Pada tahap ini data set yang telah mengalami tahap pra proses dan proses pembobotan kata akan diklasterisasi dengan menggunakan *KD-Tree K-Means Clustering* dan *K-Means Clustering* metode *Forgy's* sebagai pembanding.

IV. UJI COBA

Pada tahap ini akan dilakukan uji coba untuk menguji performa klasterisasi dokumen yang dihasilkan oleh *KD-Tree K-Means Clustering*. Data set yang digunakan sebagai bahan uji coba adalah data set *20 newsgroup* dari *KDD UCI Archive*. Uji coba dilakukan dengan membandingkan performa *KD-Tree K-Means Clustering* dengan performa 15 kali proses *K-Means Clustering*.

A. Parameter Evaluasi

Pengujian performa klasterisasi pada makalah ini akan dilakukan secara *supervised* dan *unsupervised*. Performa dari klasterisasi secara *unsupervised* akan ditentukan berdasarkan nilai distorsi atau jarak dari semua data ke titik tengah klaster terdekat. Pada data set yang sama, nilai distorsi yang semakin rendah menandakan hasil klasterisasi yang baik. Formula dari perhitungan nilai distorsi dapat dilihat pada (5) dimana *d* menyatakan fungsi jarak (*euclidean distance*), x_i menyatakan data ke-i dan c_j menyatakan klaster ke *j*.

$$D = \sum_{i=1}^{n} \left[\min_{j\ =\ (1\ \ldots\ K)} d(x_i, c_j) \right]^2 \quad (5)$$

Sedangkan untuk pengujian secara *supervised* akan dilakukan dengan cara menghitung nilai *Normalized Information Gain* (NIG) dari hasil klasterisasi [2]. NIG menggunakan konsep *entropy* sebagai pengukuran informasi. NIG merupakan hasil normalisasi dari *Information Gain* dimana nilai NIG akan bernilai 1 jika klasterisasi berhasil mendapatkan semua informasi sesuai dengan kelas, dan NIG akan bernilai 0 jika tidak ada informasi yang diterima dari hasil klasterisasi. Formula untuk menghitung NIG secara umum dapat dilihat pada (6). dimana EN_{Total} merepresentasikan nilai *Total Entropy* atau rerata informasi yang ada di setiap data pada data set dan *wEN* merepresentasikan nilai *Weighted Entropy* atau rerata informasi setiap poin pada setiap klaster. *Weighted Entropy* akan memberikan nilai 0 pada saat semua klaster homogen.

$$NIG = \frac{EN_{Total} - wEN}{EN_{Total}} \quad (6)$$

Perhitungan dari *Total Entropy,* dan *Weighted Entropy* dapat dilihat pada (7) dan (8) Nilai n menyatakan jumlah data pada data set, dan c_l menyatakan banyaknya data pada kelas l untuk setiap l=1,..,L sehingga $\sum_{l=1}^{L} c_l = n$. Pada (8), K menyatakan banyaknya klaster. Untuk setiap k=1,..,K, n_k menyatakan banyaknya data pada klaster k, dan EN_k menyatakan nilai *entropy* dari klaster K. Nilai *entropy* dari sebuah klaster dapat dihitung dengan menggunakan (9).

$$EN_{Total} = -\sum_{l=1}^{L} \left(\frac{c_l}{n}\right) log_2 \left(\frac{c_l}{n}\right)\ bits \quad (7)$$

$$wEN = \sum_{k=1}^{K} \left(\frac{n_k}{n}\right) EN_k\ \ bits \quad (8)$$

$$EN_k = -\sum_{l=1}^{L} \left(\frac{{c_l}^k}{n_k}\right) log_2 \left(\frac{{c_l}^k}{n_k}\right)\ \ bits \quad (9)$$

B. Parameter Uji Coba

Pada bagian ini akan dijelaskan nilai-nilai yang menjadi parameter pada hasil uji coba. Semua parameter pada hasil uji coba ini diturunkan dari 3 parameter utama perhitungan performa pada tugas akhir ini yaitu nilai distorsi, nilai *Normalized Information Gain*, dan waktu eksekusi proses klasterisasi dokumen dalam satuan milidetik. Daftar lengkap seluruh nilai parameter ini dapat dilihat pada Tabel 1.

C. Hasil Uji Coba

Pada skenario ini dilakukan uji coba untuk menghitung performa dari algoritma *KD-Tree K-Means Clustering* pada klasterisasi dokumen. Hasil performa *KD-Tree K-Means Clustering* pada klasterisasi dokumen ini akan dibandingkan dengam hasil dari algoritma *K-Means Clustering* pada data set yang sama. Data set yang digunakan pada skenario uji coba ini adalah data set kumpulan artikel *20 newsgroup* yang telah mengalami pra proses dan proses *term weighting*. Hasil percobaan dari skenario ini dapat dilihat pada Tabel 2.

Tabel 1.
Parameter Uji Coba

Nama Parameter	Deksripsi
K	Jumlah klaster pada proses klasterisasi
m	Jumlah fitur pada data set
row	Jumlah data pada data set
$\mathbf{D_{kd}}$	Nilai distorsi dari proses klasterisasi dokumen menggunakan algoritma *KD-Tree K-Means Clustering*
$\mathbf{Dmin_{fa}}$	Nilai distorsi minimum dari 15 kali proses klasterisasi dokumen menggunakan algoritma *K-Means Clustering*
$\mathbf{\mu_{fa}}$	Nilai rerata distorsi dari 15 kali proses klasterisasi dokumen menggunakan algoritma *K-Means Clustering*
$\mathbf{\sigma_{fa}}$	Standar deviasi distorsi dari 15 kali proses klasterisasi dokumen menggunakan algoritma *K-Means Clustering*
$\mathbf{N_{fa>kd}}$	Jumlah proses klasterisasi dokumen dari 15 kali iterasi menggunakan algoritma *K-Means Clustering* yang memiliki nilai distorsi lebih baik daripada *KD-Tree K-Means Clustering*
$\mathbf{N_{fa<kd}}$	Jumlah proses klasterisasi dokumen dari 15 kali iterasi menggunakan algoritma *K-Means Clustering* yang memiliki nilai distorsi lebih baik daripada *KD-Tree K-Means Clustering*
$\mathbf{NIG_{kd}}$	Nilai NIG dari proses klasterisasi dokumen menggunakan algoritma *KD-Tree K-Means Clustering*
$\mathbf{NIG_{fa}}$	Nilai NIG maksimum dari 15 kali proses klasterisasi dokumen menggunakan algoritma *K-Means Clustering*
$\mathbf{T_{kd}}$	Waktu eksekusi dari proses klasterisasi dokumen menggunakan algoritma *KD-Tree K-Means Clustering*
$\mathbf{Tmin_{fa}}$	Waktu eksekusi minimum dari 15 kali proses klasterisasi dokumen menggunakan algoritma *K-Means Clustering*
$\mathbf{Tmax_{fa}}$	Waktu eksekusi maksimum dari 15 kali proses klasterisasi dokumen menggunakan algoritma *K-Means Clustering*
$\mathbf{Ttotal_{fa}}$	Waktu eksekusi total dari 15 kali proses klasterisasi dokumen menggunakan algoritma *K-Means Clustering*

Tabel 2.
Hasil Uji Coba Klasterisasi pada *20 newsgroup*

Parameter	Nilai
K	20
m	20.536
$\mathbf{D_{kd}}$	$4{,}14 \times 10^7$
$\mathbf{Dmin_{fa}}$	$4{,}12 \times 10^7$
$\mathbf{\mu_{fa}}$	$4{,}17 \times 10^7$
$\mathbf{\sigma_{fa}}$	$3{,}00 \times 10^5$
$\mathbf{N_{fa>kd}}$	4
$\mathbf{N_{fa<kd}}$	11
$\mathbf{NIG_{kd}}$	0,18
$\mathbf{NIG_{fa}}$	0,09
$\mathbf{T_{kd}}$	13.295.630
$\mathbf{Tmin_{fa}}$	2.535.463
$\mathbf{Tmax_{fa}}$	12.619.237
$\mathbf{Ttotal_{fa}}$	105.325.216

Dari hasil uji coba ini dapat dilihat bahwa proses klasterisasi dokumen pada algoritma *KD-Tree K-Means Clustering* memiliki nilai distorsi sebesar $4{,}14 \times 10^7$. Hasil ini lebih buruk 2×10^5 dibandingkan nilai distorsi minimum hasil *K-Means Clustering*. Namun Hasil ini lebih baik 3×10^5 dibandingkan dengan nilai rerata distorsi dari *K-Means Clustering* dengan standar deviasi $9{,}9 \times 10^4$. Sedangkan pada perhitungan nilai NIG, hasil dari *KD-Tree K-Means Clustering* memiliki nilai NIG 0,18. Hasil ini lebih baik 0,09 dibandingkan dengan nilai NIG maksimum yang didapatkan oleh *K-Means Clustering.*

Untuk waktu eksekusi dari algoritma *KD-Tree K-Means Clustering* pada klasterisasi dokumen *20 newsgroup* memiliki waktu eksekusi 13.295.630 milidetik (3 jam 41 menit). Sedangkan waktu eksekusi dari sekali proses *K-Means Clustering* memiliki rentang antara 2.535.463 milidetik hingga 12.619.237 milidetik (42 menit hingga 3 jam 30 menit).

V. KESIMPULAN

Dari hasil uji coba dapat diambil kesimpulan bahwa performa klasterisasi dokumen yang dihasilkan oleh metode *KD-Tree K-Means Clustering* pada data set *20 newsgroup* menghasilkan nilai distorsi 3×10^5 lebih rendah dibandingkan dengan nilai rerata distorsi dari *K-Means Clustering* dan nilai NIG 0,09 lebih baik dibandingkan nilai NIG maksimum *K-Means Clustering*.

Sedangkan dari segi waktu, algoritma *KD-Tree K-Means Clustering* memiliki waktu *training* yang cukup lama. Oleh karena itu, perbaikan yang bisa dilakukan pada penelitian selanjutnya adalah dengan memperbaiki efisiensi running time dari *KD-Tree K-Means Clustering* sehingga waktu eksekusi dari *KD-Tree K-Means Clustering* secara keseluruhan dapat ditingkatkan.

DAFTAR PUSTAKA

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

[2] Stephen J Redmond and Conor Heneghan, "A method for initialising the K-means clustering algorithm using kd-trees," *Pattern Recognition Letters*, vol. 28, no. 8, pp. 965–973, June 2007.

[3])D UCI Archive. (2013, March) 20 Newsgroups Dataset. [Online]. dd.ics.uci.edu/databases/20newsgroups/20newsgroups.html

[4] E. W. Forgy, "Cluster analysis of multivariate data: efficiency versus interpretability of classifications," *Biometrics*, vol. 21, pp. 768–769, 1965.

[5] M. Mahajan, P. Nimbhorkar, and K. Varadarajan, "The Planar k-Means Problem is NP-Hard," *Lecture Notes in Computer Science*, vol. 5431, pp. 274–285, 2009.

[6] M. Inaba, N. Katoh, and H. Imai, "Applications of weighted Voronoi diagrams and randomization to variance-based k-clustering," in *Proceedings of 10th ACM Symposium on Computational Geometry*, 1994, pp. 332–339.

[7] J.L Bentley, "Multidimensional binary search trees used for associative searching," *Communications of the ACM*, vol. 18, no. 9, pp. 509-517, 1975.

[8] M. da Berg, O. Cheong, M. van Kreveld, and M. Overmars, *Computational Geometry, Algorithm and Applications*, 3rd ed.: Springer, 2008.

[9] I. Katsavounidis, C.C.J. Kuo, and Z. Zhen, "A new initialization technique for generalized lloyd iteration," *IEEE Signal Processing Letter*, vol. 1, no. 10, pp. 144–146, 1994.

[10] G. Salton and MJ. McGill, *Introduction to modern information retrieval*.: McGraw-Hill, 1986.

[11] Christopher D. Manning, Raghavan Prabhakar, and Hinrich Schütze, *An Introduction to Information Retrieval*, 1st ed., Cambridge University Press, Ed. Cambridge, England: Cambridge University Press, 2009.

[12] David D. Lewis, Yiming Yang, Tony G. Rose, and Fan Li. (2004) Journal of Machine Learning Research. [Online]. http://jmlr.csail.mit.edu/papers/volume5/lewis04a/a11-smart-stop-list/english.stop

[13] Harun Uğuz, "A two-stage feature selection method for text categorization by using information gain, principal component

analysis, and genetic algorithm," *Knowledge-Based System*, vol. 24, no. 7, pp. 1024-1032, April 2011.

[14] M.F. Porter, "An algorithm for suffix stripping," *Program*, vol. 14, no. 3, p. 130−137., 1980.

[15] Gerard Salton and Christopher Buckley, "Term-weighting approaches in automatic text retrieval," *Information Processing and Management*, pp. 513-523, 1988.