JURUSAN TEKNIK INFORMATIKA

FAKULTAS TEKNOLOGI INFORMASI

INSTITUT TEKNOLOGI SEPULUH NOPEMBER

USULAN TUGAS AKHIR

1. IDENTITAS PENGUSUL

Nama : Eric Budiman Gosno

NRP : 5109100153

Dosen Wali : Waskitho Wibisono, S.Kom, M.Eng., Ph.D.

2. JUDUL TUGAS AKHIR

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

"Implementation of K-Means Clustering Algorithm with KD-Tree for Document Clustering"

3. URAIAN SINGKAT

Informasi data yang berbasis teks/data yang berbentuk dokumen teks merupakan bentuk data yang lazim dijumpai pada sistem informasi penting seperi jejaring sosial, sistem informasi perusahaan, dan arsip berita web. Ukuran data dokumen yang disimpan umumnya berkembang dengan sangat cepat sehingga proses manajemen, dan analisis data dokumen menjadi permasalahan yang sangat penting.

Klasterisasi dokumen merupakan suatu proses untuk membagi sebuah set dokumen ke dalam beberapa kelompok spesifik yang dalam proses klasterisasi disebut sebagai *cluster* di mana setiap *cluster* memiliki kumpulan dokumen yang memilki nilai kesamaan/*similiarity* antar dokumen yang tinggi dan antar satu *cluster* dengan *cluster* yang lain memiliki dokumen dengan nilai kesamaan yang minimal. Salah satu metode *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/*centroid cluster* sehingga pemilihan posisi awal dari *centroid cluster* yang buruk akan mengakibatkan algoritma *K-Means Clustering* terjebak

dalam *local optimization*. *KD-Tree K-Means Clustering* merupakan pengoptimalan dari algoritma *K-Means Clustering* dengan menggunakan struktur data *K-Dimensional Tree*, dan *density rank* pada proses inisialisasi *centroid cluster*.

Pada tugas akhir ini penulis mencoba untuk melakukan proses klasterisasi dokumen dengan menggunakan algoritma *KD-Tree K-Means Clustering*. Diharapkan penggunaan *KD-Tree K-Means Clustering* dapat meningkatkan hasil dan performa dari klasterisasi dokumen dibandingkan dengan metode-metode klasterisasi dokumen yang sebelumnya.

4. PENDAHULUAN

4.1 LATAR BELAKANG

Informasi data yang berbasis teks/data yang berbentuk dokumen teks merupakan bentuk data yang lazim dijumpai pada sistem informasi penting seperi jejaring sosial, sistem informasi perusahaan, dan arsip berita web. Volume data dokumen yang disimpan seringkali berkembang dengan sangat cepat sehingga proses manajemen, dan analisis dari data dokumen menjadi permasalahan yang sangat penting.

Klasterisasi dokumen merupakan suatu proses untuk membagi sebuah set dokumen ke dalam beberapa kelompok spesifik yang dalam proses klasterisasi disebut sebagai *cluster* di mana setiap *cluster* memiliki kumpulan dokumen yang memilki nilai kesamaan/*similiarity* antar dokumen yang tinggi dan antar satu *cluster* dengan *cluster* yang lain memiliki dokumen dengan nilai kesamaan yang minimal.

Secara umum, permasalahan *Clustering* dapat dibagi menjadi 2 jenis yaitu *Hierarchial Clustering* dan *Partitional Clustering*. *Partitional Clustering* merupakan suatu permasalahan untuk membagi/melakukan partisi dari sebuah dataset ke dalam sejumlah kelompok data yang disebut sebagai *cluster* dengan memaksimalkan total nilai kemiripan/*similiarity* antar data-data yang ada pada *cluster* yang sama. Mencari hasil partisi yang menghasilkan nilai optimal pada *Partitional Clustering* merupakan permasalahan *NP-Complete*, tetapi terdapat beberapa strategi suboptimal yang dapat memberikan solusi yang kompetitif dalam kompleksitas linear. 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/*centroid cluster* sehingga pemilihan posisi awal dari *centroid cluster* yang buruk akan mengakibatkan algoritma *K-Means Clustering* terjebak

dalam *local optimization* [1]. Saat ini, telah diusulkan beberapa metode pengoptimalan dari *K-Means Clustering* pada proses inisialisasi *centroid cluster*, salah satunya adalah algoritma *KD-Tree K-Means Clustering* [2]. *KD-Tree K-Means Clustering* merupakan pengoptimalan dari algoritma *K-Means Clustering* dengan menggunakan struktur data *K-Dimensional Tree*, dan *density rank* pada proses inisialisasi *centroid cluster*. Hasil uji penelitian [2] telah menunjukkan bahwa *KD-Tree K-Means Clustering* dapat meningkatkan hasil *clustering* 4-71% dibandingkan dengan metode-metode pengoptimalan *K-Means Clustering* sebelumnya, dan waktu eksekusi 27 kali lebih cepat dibandingkan dengan metode *K-Means Clustering* dasar (*Forgy's Method* [3]), tetapi hasil uji penelitian tersebut tidak melingkupi klasterisasi pada data dokumen.

Oleh karena itu, pada tugas akhir ini akan dilakukan proses klasterisasi dokumen dengan menggunakan algoritma *KD-Tree K-Means Clustering*. Diharapkan penggunaan *KD-Tree K-Means Clustering* dapat meningkatkan hasil dan performa dari klasterisasi dokumen dibandingkan dengan metode *K-Means Clustering*.

4.2 RUMUSAN MASALAH

Beberapa permasalahan yang dibahas dalam tugas akhir ini dapat diuraikan sebagai berikut:

1. Bagaimana mengimplementasikan algoritma *KD-Tree K-Means Clustering* pada kasus klasterisasi dokumen?
2. Bagaimana hasil dan performa dari algoritma *KD-Tree K-Means Clustering* dibandingkan dengan metode *K-Means Clustering*?

4.3 BATASAN MASALAH

Permasalahan yang dibahas dalam tugas akhir ini memiliki batasan-batasan sebagai berikut:

1. Algoritma akan diimplementasi menggunakan bahasa pemrograman C# dengan bantuan perangkat lunak Microsoft Visual Studio 2010.
2. Pustaka yang digunakan saat mengimplementasi algoritma menggunakan pustaka yang terdapat dalam *.NET Framework Class Library.*
3. Kriteria evaluasi dari hasil klasterisasi dokumen akan didasarkan terhadap pengukuran internal, dan eksternal. Kriteria evaluasi eksternal merupakan evaluasi

hasil *clustering* dibandingkan dengan kategori/label dokumen yang sebenarnya. Kriteria evaluasi eksternal hanya akan dilakukan jika pada informasi dataset terdapat atribut kategori/label dokumen. Untuk kriteria evaluasi internal akan didasarkan pada waktu eksekusi algoritma, dan hasil klasterisasi algoritma. Waktu eksekusi algoritma merupakan waktu yang diperlukan untuk menyelesaikan proses klasterisasi dokumen dari data yang berbentuk *vector space model*. Proses *preprocessing* dan *postprocessing* dari klasterisasi dokumen tidak termasuk dalam evaluasi waktu eksekusi. Hasil klasterisasi dokumen akan dievaluasi dengan menggunakan *F-Measure* [4] dan nilai total kemiripan antara dokumen dengan *centroid cluster* terdekat menggunakan *Euclidean Distance* dan *Cosine.*

4. Proses *stemming* pada fase *preprocessing* menggunakan metode *Porter Stemming* [5].
5. Proses konversi dokumen menjadi *word vector space* akan menyimpan maksimal 1000 kata sebagai atribut dari data *word vector space.*
6. Dokumen pada dataset menggunakan Bahasa Inggris/*English language*.

4.4 TUJUAN TUGAS AKHIR

Tujuan dari pengerjaan tugas akhir ini adalah mengimplementasikan, dan melakukan uji performa dari algoritma *KD-Tree K-Means Clustering* dalam permasalahan klasterisasi dokumen.

4.5 MANFAAT TUGAS AKHIR

Manfaat yang diharapkan dari hasil tugas akhir ini adalah memberikan referensi untuk metode klasterisasi dokumen yang dapat menghasilkan performa yang baik dalam segi waktu, kompleksitas, dan akurasi hasil *clustering*.

5 TINJAUAN PUSTAKA

5.1 Algoritma K-Means Clustering

Algoritma *K-Means Clustering* merupakan salah satu metode dari analisis cluster/*clustering* yang bertujuan untuk melakukan partisi dari sebuah dataset ke dalam *k* cluster. Dalam prosesnya, *K-Means Clustering* akan mencari titik tengah/*centroids* dari semua *cluster* yang menghasilkan total nilai jarak/*distortion* dari setiap data observasi ke titik tengah

cluster seminimal mungkin. Alur metode dari algoritma *K-Means Clustering* dapat dilihat pada Gambar 1.

Algoritma *K-Means Clustering*

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

Gambar 1. Psudocode K-Means Clustering

Algoritma *K-Means Clustering* dengan nilai *K* tertentu secara umum memiliki kompleksitas *NP-Hard* [6], tetapi jika diberikan nilai *D*/Distorsi yang diharapkan, maka Algoritma *K-Means Clustering* dapat diselesaikan dengan kompleksitas $O(n^{dk+1} \log n)$ dimana *n* menyatakan banyaknya data observasi [7]. Keunggulan dari K-Means Clustering adalah mudah untuk diimplementasi, dan diaplikasikan pada dataset yang berukuran besar seperti pada permasalahan segmentasi pasar, visi computer, *geostatic*, astronomi, dan pertanian.

5.2 K-Dimensional Tree

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

Dalam implementasi, *KD-Tree* adalah *binary tree* dimana setiap *node* pada *binary tree* tersebut adalah sebuah *point* 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. *Pseudocode* dari pembentukan *KD-Tree* dapat dilihat pada Gambar 2.

Fungsi kdtree (list point, *depth*)

1. Pilih Atribut/*axis* yang digunakan sebagai pembatas ($axis = depth \bmod k$).
2. Urutkan data/point berdasarkan *axis* yang dipilih dan pilih *median* sebagai *pivot element.*
3. Buat node dan konstruksi *subtress* dengan melakukan fungsi rekursif:
   node.leftChild := kdtree(points sebelum median, depth+1);
   node.rightChild := kdtree(points setelah median, depth+1);

Gambar 2. Pseudocode fungsi *Build KD-Tree*

5.3 KD-Tree K-Means Clustering

KD-Tree K-Means Clustering [2] merupakan algoritma pengoptimalan dari algoritma *KKZ/ Katsoudivinis* [9] yang menggunakan *density* dan *distance* antar poin/data dalam menentukan posisi awal *centroid cluster K-Means Clustering*. *KD-Tree K-Means Clustering* menggunakan struktur data *K-Dimensional Tree* dalam proses inisialisasi *centroid cluster*.

Hasil uji penelitian telah menunjukkan bahwa *KD-Tree K-Means Clustering* dapat meningkatkan hasil *clustering* 4-71% dibandingkan dengan metode-metode pengoptimalan *K-Means Clustering* sebelumnya, dan waktu eksekusi 2700% lebih cepat dibandingkan dengan metode *K-Means Clustering* dasar [3]. *KD-Tree K-Means Clustering* juga merupakan metode yang mampu menangani dataset yang memiliki *noise/outlier* dengan cara menghapus 20% calon *centroid cluster* dengan nilai *density* terendah.

Proses inisialisasi *centroid cluster* pada *KD-Tree K-Means Clustering* dimulai dengan pembentukan *kd-tree* dengan menetapkan bahwa *leaf bucket* dari *kd-tree* memiliki maksimal $\frac{n}{10Kc}$ data dimana *n* menyatakan banyak data dan *Kc* menyatakan banyak *cluster* yang akan dibuat pada proses *clustering*. Selanjutnya akan dikalkulasi nilai *density* dari setiap *leaf bucket* dengan formula $p_j = \frac{N_j}{V_j}$ dimana N_j menyatakan banyak data pada *leaf bucket j*,dan V_j menyatakan volume dari *leaf bucket j*.

Posisi dari setiap *leaf bucket* akan diindentifikasi sebagai nilai *mean* dari posisi data-data yang berada di dalam *leaf bucket* tersebut. Sehingga untuk *leaf bucket* sebanyak *q* akan

didapatkan sekumpulan point *leaf bucket* (*m1,m2,..,mq)* pada *space* beserta dengan nilai *density* dari setiap point (*p1,p2,…,pq)*.

Untuk memilih *initial centroid* 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 *density* dari yang terbesar hingga yang terkecil. *Leaf Bucket* yang memiliki nilai *density* tertinggi akan dipilih sebagai *centroid cluster* pertama. Pemilihan *centroid cluster* yang berikutnya akan didasarkan pada nilai jarak antara calon *centroid/leaf bucket* dengan *centroid* yang telah ditetapkan, dan nilai *density* dari *leaf bucket* tersebut. Semakin jauh jarak *leaf bucket* dari posisi *centroid* yang telah ditetapkan, dan semakin besar nilai *density leaf bucket* tersebut, maka semakin tinggi peluang *leaf bucket* tersebut menjadi *centroid* yang baru. Secara matematis, formula untuk menentukan nilai dari *leaf bucket* dapat dilihat pada Persamaan (1):

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

Dengan fungsi *d* menyatakan fungsi jarak(*euclidean distance), Ck* menyatakan posisi *centroid cluster, Mj* menyatakan posisi *leaf bucket*, dan *Pj* menyatakan nilai *density* dari *leaf bucket*. Untuk setiap iterasi, *Leaf bucket* dengan nilai *Gj* terbesar akan dipilih sebagai *centroid cluster* baru. Iterasi akan terus dijalankan hingga didapatkan *Kc centroid cluster* yang akan digunakan sebagai inisialisasi pada *K-Means Clustering*.

Untuk mengatasi permasalahan terkait dengan terpilihnya data *noise/outlier* menjadi calon *centroid,* algoritma akan membuang 20% *leaf bucket*/calon *centroid* yang memiliki nilai *density* terendah. Dengan hanya menggunakan 80% *leaf bucket* yang memiliki *density* tertinggi, maka kemungkinan data *outlier* terpilih menjadi *centroid cluster* dapat diminimalisasi sebelum algoritma dijalankan. Alur kerja dari algoritma *KD-Tree K-Means Clustering* secara keseluruhan dapat dilihat pada Gambar 3.

Algoritma *KD-Tree K-Means Clustering*

1. Pembentukan *K-Dimensional Tree* dari dataset. *K-Dimensional Tree* yang dibuat akan memiliki jumlah *leaf bucket* maksimal $q = 10*Kc$ (*Kc* = banyak *cluster* pada *K-Means Clustering*).
2. Untuk setiap *leaf bucket*($L_1,L_2,\ldots,L_j$) , kalkulasi nilai *density*(P_j) dari setiap *leaf bucket* L_j, dan kalkulasi nilai titik tengah *leaf bucket*(M_j) dengan mencari nilai

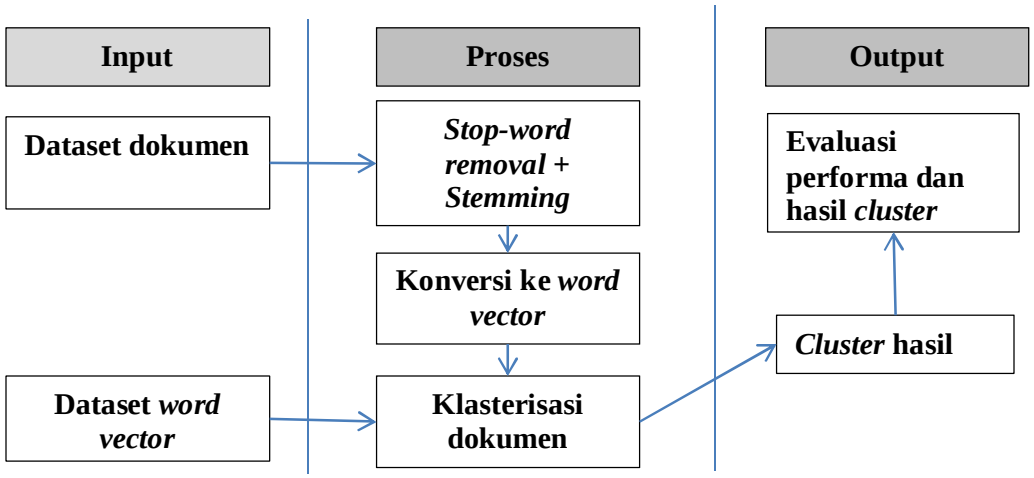
mean dari semua point yang ada pada *leaf bucket* L_j.

3. Pilih *centroid* pertama $C_1 = M_z$, dimana $z = \arg\max(P_j)$.
4. Untuk $t = 2,\ldots,K$:
   - Untuk $j = 1,\ldots,q$ 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 $z = \arg\max(G_j)$.
   - Hapus 20% *leaf bucket* dengan nilai *density* terendah. Kembali ke langkah 3 dan kalkulasi posisi *K centroid cluster* baru $(\hat{c}_1, \hat{c}_2, \ldots, \hat{c}_k)$.
5. Jalankan algoritma *K-Means Clustering* dengan nilai inisialisasi titik tengah ($C_1,\ldots,C_k$) dan $(\hat{c}_1, \hat{c}_2, \ldots, \hat{c}_k)$.

Gambar 3. Algoritma KD-Tree K-Means Clustering

6 METODOLOGI

Berikut merupakan alur, input dan output dari algoritma yang akan dibuat seperti pada Gambar 4.



Gambar 4. Gambaran alur kerja tugas akhir

Keterangan Gambar 4:

1. **Input**:

   Input dari algoritma meruapakan dataset dokumen, atau dataset dokumen yang sudah dikonversi ke dalam model *Word Vector Space.*

2. **Proses:**

- ***Stemming***: Proses *Stemming* bertujuan untuk menyerderhanakan kata-kata yang ada pada dokumen ke dalam bentuk sederhana (Contoh : mengubah kata *national* menjadi *nation*). Proses ini merupakan proses *preprocessing* dari dataset dokumen sebelum dikonversi menjadi *word vector*. Implementasi dari proses ini dapat dilihat pada Subbab 6.1.2.1.
- ***Stop-word removal***: Proses ini bertujuan untuk menghilangkan *term-term* pada dokumen yang termasuk dalam *stop word*. Proses ini merupakan proses *preprocessing* dari dataset dokumen sebelum dikonversi menjadi *word vector*. Implementasi dari proses ini dapat dilihat pada Subbab 6.1.2.2.
- **Konversi ke *word vector***: Proses ini bertujuan untuk mengubah dataset yang berbentuk dokumen menjadi *word vector*. Implementasi dari proses ini dapat dilihat pada Subbab 6.1.2.3.
- **Klasterisasi dokumen**: Proses ini bertujuan untuk melakukan proses klasterisasi pada dataset dokumen yang sudah dikonversi menjadi *word vector*. Pada tugas akhir ini, akan diimplementasikan 2 algoritma klasterisasi dokumen yaitu *KD-Tree K-Means Clustering*, dan *K-Means Clustering*. Implementasi dari proses ini dapat dilihat pada Subbab 6.1.2.4.

3. **Output:**

   Proses implementasi tugas akhir ini akan menghasilkan output berupa dokumen-dokumen yang telah dikelompokkan dalam *cluster-cluster* sebagai hasil dari proses klasterisasi dokumen, dan evaluasi performa, dan hasil dari proses klasterisasi.

6.1 Langkah Pengerjaan

Dalam pembuatan aplikasi ini, penulis telah menyusun beberapa langkah dalam pengerjaan aplikasi. Berikut langkah kerja yang telah disusun penulis:

6.1.1 Studi Literatur

Studi literatur mencakup pembelajaran pada algoritma-algoritma terkait, analisis *prove of correctness*, serta perhitungan kompleksitas. Pada tahap ini juga dilakukan studi terhadap beberapa implementasi terdahulu apabila tersedia untuk dijadikan acuan pada tahap selanjutnya.

6.1.2 Implementasi Algoritma dan Uji Coba

Hasil studi pada tahap sebelumnya menjadi dasar pada tahap implementasi. Bahasa yang digunakan adalah C# dengan bantuan IDE Microsoft Visual Studio 2010.

6.1.2.1 Implementasi *Stemmer*

Untuk proses *Stemmer*/penyederhanann kata pada dataset dokumen, penulis menggunakan *Porter Stemmer* [5]. *Pseudocode* dari *Porter Stemmer* dapat dilihat pada Gambar 5.

Porter Stemmer

1. Hapus bentuk *plurals* dan *suffix* seperti -ed, -ing.
2. Apabila pada *terms* terdapat karakter vokal selain *i*, ubah karakter *y* menjadi *i*
3. Lakukan proses *mapping* kata yang memiliki *double suffixes* menjadi *single suffix* (Contoh: -ization, -ational, dan lain lain).
4. Proses suffix-suffix khusus seperti –full, -ness, dan lain lain.
5. Hapus *suffix* –ant, -ence, dan lain lain.
6. Hapus *suffix* –*e.*

Gambar 5. Pseudocode *Porter Stemmer*

6.1.2.2 Implementasi *Stop-Word Removal*

Untuk mendeteksi apakah suatu kata merupakan *Stop Word* dalam bahasa inggris, digunakan database *Stop-Word* yang dibuat oleh *Lewis et al* [10]. *Pseudocode* dari metode *Stop-Word Removal* yang digunakan dapat dilihat pada Gambar 6.

Algoritma *Stop-Word Removal*

1. Bentuk daftar frekuensi *terms*/kata dari setiap dokumen pada dataset.
2. Lakukan pengecekan apakah kata tersebut termasuk dalam *database stop-word*. Jika kata tersebut termasuk dalam *database*, maka hapus kata dari daftar *terms.*
3. Lakukan langkah ke 2 untuk semua kata pada daftar *terms.*

Gambar 6. Pseudocode metode stop-word removal

6.1.2.3 Implementasi Konversi Dokumen ke *Word Vector*

Untuk proses konversi dokumen ke *Word Vector*, penulis menggunakan metode *term frequencies and term weighting* TF-IDF [11]. TF/*Term Frequency* dapat dikalkulasi dengan formula $TF(t,d) = \frac{f(t,d)}{\max\{f(w,d):w \in d\}}$. IDF/*Inverse Document Frequency* dapat di kalkulasi

dengan formula $IDF(t, D) = log \frac{|D|}{|\{d \in D : t \in d\}|}$. Sehingga nilai TF-IDF secara keseluruhan adalah $TFIDF(t, d, D) = TF(t, d) \times IDF(t, D)$.

6.1.2.4 Implementasi Algoritma *Clustering*

Terdapat 2 algoritma *Clustering* yang diimplementasi yaitu algoritma *KD-Tree K-Means Clustering* sebagai algoritima yang dibahas, dan algoritma *K-Means Clustering* sebagai pembanding hasil,dan performa. Pseudocode dari *KD-Tree K-Means Clustering* dapat dilihat pada Gambar 7. Sedangkan Pseudocode dari algoritma *K-Means Clustering* dapat dilihat pada Gambar 8.

Algoritma *KD-Tree K-Means Clustering*

1. Pembentukan *K-Dimensional Tree* dari dataset. *K-Dimensional Tree* yang dibuat akan memiliki jumlah *leaf bucket* maksimal q =10**Kc* (*Kc* = banyak *cluster* pada *K-Means Clustering*).
2. Untuk setiap *leaf bucket* (L_1,L_2 ,…,L_j) , kalkulasi nilai *density*, P_j dari setiap *leaf bucket* L_j, dan kalkulasi nilai titik tengah *leaf bucket*, M_j, dengan mencari nilai *mean* dari semua point yang ada pada *leaf bucket* L_j.
3. Pilih *centroid* pertama *C1* = *Mz*, dimana $z = \arg\max(Pj)$.
4. Untuk t = 2,…,*K*
   - Untuk j = 1,...,q kalkulasi nilai $Gj = \left\{ \min_{k = 1 \ldots t} \left[d(C_k, M_j) \right] \right\} \times P_j$.
   - $C_t = M_z$, dimana $z = \arg\max(G_j)$.
   - Hapus 20% *leaf bucket* dengan nilai *density* terendah. Kembali ke langkah 3 dan kalkulasi posisi *K centroid cluster* baru ($\hat{c}_1, \hat{c}_2, \ldots, \hat{c}_k$).
5. 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 7. Algoritma KD-Tree K-Means Clustering

Algoritma *K-Means Clustering*

1. Tentukan jumlah cluster K, nilai distorsi *threshold* Dt, dan jumlah iterasi maksimum.
2. Lakukan proses inisialisasi K titik tengah *cluster*.
3. Hubungkan setiap data observasi ke *cluster* terdekat.
4. Kalkulasi ulang posisi titik tengah cluster.

5. Hitung nilai distorsi $D = \sum_{i=1}^{n} \left[\min_{j = (1 \ldots K)} d(x_i, C_j) \right]^2$.

Apabila nilai D belum memenuhi Dt dan jumlah iterasi < jumlah iterasi maksimum, kembali ke langkah 3. Jika tidak, maka kembalikan hasil *cluster*.

Gambar 8. Algoritma K-Means Clustering

6.1.3 Eksperimen dan Evaluasi

Pada tahap ini hasil implementasi dari *KD-Tree K-Means Clustering* akan diujikan pada dataset dokumen yang telah ditentukan. Performa, dan hasil dari proses *Clustering* antara metode yang diusulkan dengan metode *K-Means Clustering* akan menjadi dasar acuan dari tahap evaluasi, dan pengambilan kesimpulan.

Input yang digunakan pada fase eksperimen dan evaluasi adalah dataset *reuters-21578* dari *UCI Machine Learning Repository* [11], dan *20 newsgroup dataset* [12].

6.1.4 Penyusunan buku tugas akhir

Tahap ini merupakan penyusunan laporan berupa buku tugas akhir sebagai dokumentasi pelaksanaan tugas akhir, yang mencakup seluruh teori, implementasi, serta hasil pengujian yang telah dikerjakan.

7 JADWAL PEMBUATAN TUGAS AKHIR

Tugas akhir ini diharapkan bisa dikerjakan sesuai jadwal, sebagai berikut.

Tahapan	2013																			
	Maret				April				Mei				Juni				Juli			
Penyusunan Proposal																				
Studi Literatur																				
Implementasi																				
Pengujian dan Evaluasi																				
Penyusunan Buku																				

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

[4] B. Larsen and C. Aone, "Fast and effective text mining using linear-time document clustering," in *Proceedings of the fifth ACM SIGKDD Int'l conference on knowledge discovery and data mining*, 1999, pp. 16–22.

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

[6] 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.

[7] 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.

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

[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] 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

[11] David D. Lewis. (2013, March) Reuters-21578 Text Categorization Collection Data Set. [Online]. http://archive.ics.uci.edu/ml/datasets/Reuters-21578+Text+Categorization+Collection

[12] University of California, Irvine. (2013, March) 20 Newsgroups Dataset. [Online]. http://kdd.ics.uci.edu/databases/20newsgroups/20newsgroups.html

[13] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze, *An Introduction to Information Retrieval*. Cambridge, England: Cambridge University Press, 2009, pp. 117-119.