PENGGUNAAN KNN *(K-NEARST NEIGHBOR)* UNTUK KLASIFIKASI TEKS BERITA YANG TAK-TERKELOMPOKKAN PADA SAAT PENGKLASTERAN OLEH STC (*SUFFIX TREE CLUSTERING*)

Jumadi, Edi Winarko

Abstrak

Dokumen teks yang dipublikasi di internet dari hari ke hari semakin banyak jumlahnya. Salah satu teknologi internet yang paling sering terjadi proses pemuktahiran konten dokumen teks ini, adalah microblogging yang dijadikan sebagai sarana untuk membangun komunitas di dunia maya dan penyebar informasi yang praktis dan cepat. Salah satunya adalah Twitter yang merupakan salah satu social media dengan jumlah tweet yang dipublikasi dalam hitungan jam oleh para pemilik akun tersebut, khususnya para jurnalis.

Berita-berita yang dipublikasi oleh para jurnalis melaui Twitter terkadang kurang nyaman untuk dibaca oleh para pembaca berita. Karena berita-berita tersebut ditampilkan secara tersusun beruntun ke bawah pada halaman web tersebut. Tetapi setelah tweet-tweet yang ada dikelompokkan secara tematik jadi semakin menarik karena pembaca dapat memilih berita-berita tertentu yang telah dikelompokkan oleh Algoritma Suffix Tree Clustering (STC). Tetapi pada algoritma ini, masih tetap menghasilkan dokumen-dokumen yang tidak memiliki kelompok. Pada Penelitian ini, dokumen-dokumen tersebut mencoba untuk di klasifikasikan ke dalam kelompok yang ada dengan menggunakan Algoritma K-Nearset Neighbor (KNN).

Kata-kata kunci: *Clasification, Clustering, TF-IDF, KNN, STC*

Pendahuluan

Berdasarkan penelitian yang dilakukan oleh Zamir dan Etzioni (1998), algoritma yang digunakan untuk melakukan pengklastran dokumen web kali pertama adalah *Suffix Tree Clustering (STC),* algoritma klasterisasi ini memiliki waktu linear dalam mengelompokkan dokumen hasil pencarian ke dalam bentuk group-group atau klaster berdasarkan kata atau frase yang terdapat di dalam dokumen yang ada. Kemudian Osiński dan Weiss (2004), mengembangkan *Open Source Framework* dengan nama *Carrot*^{2}.

Kesuksesan dan popularitas aplikasi *Carrot2* adalah mengorganisir hasil dari pencaraian di internet agar lebih mudah dalam menjelajah dalam bentuk pengelompokan secara tematik hasil pencarian pada saat menggunakan browser internet, yang dikenal dengan proses klasterisasi. Algoritma yang digunakan dalam proses pengelompokan ini, diantaranya adalah menggunakan algoritma *Suffix Tree Clustering*. Selanjutnya, penelitian yang telah dilakukan oleh Arifin dkk.(2008), dengan menggunakan Algoritma *Suffix Tree Clustering* dalam pengelompokkan berita dalam Bahasa Indonesia, memiliki tingkat *precision* yang sangat tinggi, yaitu 80%.Hal ini dikarenakan dalam Algoritma ini, menggunkaan *phrase* sebagai dasar pembentukan *cluster*.

Tetapi, kinerja algoritma STC yang dikembangkan oleh *Carrot*^{2} masih memiliki kekurangan. Hasil proses pengklasteran dengan algoritma ini, sering dijumpai banyak dokumen yang tidak terkelompokkan. Mengacu pada konsep yang dibahas oleh Liao (2002), untuk mengatasi permasalahan ini perlu adanya proses klasifikasi dokumen teks tersebut denga Algoritma K-NN ke dalam kelompok yang terbentuk oleh Algoritma STC sebelumnya.

Teori

Algoritma *Suffix Tree Clustering* (STC)

Inti dari suatu hasil pencarian yang menerapkan *clustering* adalah penggunaan algoritma *clustering.* Algoritma *Suffix Tree Clustering* (STC) memiliki dua kunci utama, yaitu :

1. Menggunakan *phrase* sebagai dasar pembentukan *cluster*nya.
2. Menggunakan suatu definisi *cluster* sederhana.

Suffix Tree Clustering memiliki dua langkah utama. Dalam langkah pertama, pencarian *shared phrase* untuk semua dokumen berita yang dikoleksi.

Kita menyebut *shared phrase* sebagai *phrase cluster* atau *base cluster,* yang ditemukan dengan menggunakan suatu struktur data yang dinamakan *suffix tree.* Dalam langkah kedua, kita mengkombinasikan *base cluster-base cluster* ke dalam suatu klaster. Penggabungan antar dua *base cluster* didasarkan pada jumlah dokumen yang melakukan *overlap* diantara kedua *base cluster* tersebut (Zamir & Etzioni, 1998). Suatu *phrase* yang dimaksud dalam konteks algoritma ini adalah urutan satu atau lebih kata-kata. STC memiliki tiga langkah utama, yaitu :

1. Pembersihan (*cleaning)* dokumen.
2. Identifikasi *base cluster* menggunakan *suffix tree.*
3. Mengkombinasikan *base cluster* ke dalam suatu *cluster.*

Beberapa karakteristik yang membuat *Suffix Tree Clustering* cocok digunakan untuk pengelompokkan dokumen. Pertama adalah membangkitkan klaster-klaster untuk pengelompokkan dokumen berdasarkan *phrase. Phrase* juga bermanfaat untuk membangun uraian dan keakuratan deskripsi dari klaster. Kedua, tidak tergantung pada model data. Hal itu mengasumsikan hanya dokumen dengan topik yang sama yang akan memiliki *shared phrase.* Ketiga, STC memperbolehkan adanya *overlaping cluster*. Hal itu sangat penting untuk menghindari pembatasan bahwa setiap dokumen hanya memiliki satu klaster saja, karena sering kita jumpai satu dokumen mempunyai lebih dari satu topik, dengan demikian terdapat kemiripan yang lebih dari satu kelompok dokumen. Keempat, STC menggunakan definisi klaster yang sederhana. Semua dokumen yang berisi salah satu *phrase cluster* akan menjadi anggota dari klaster tersebut. STC menggunakan *phrase* untuk mendeteksi kemiripan antar dokumen. STC menggunakan *suffix tree* untuk mengidentifikasi *phrase.* Fitur yang

membuat suksesnya STC sebagai algoritma *clustering* adalah adanya *overlaping cluster*. Kualitas klaster yang terbentuk dari algoritma STC ini akan menurun jika tanpa menggunakan *multiword phrase* dan tidak memperbolehkan adanya *overlaping cluster*.

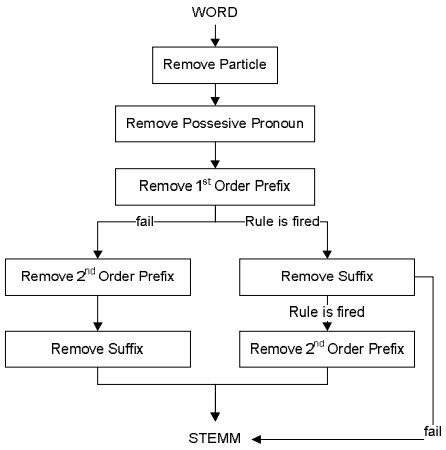
Pembersihan dokumen *(document cleaning)*

Document Cleaning adalah tahap awal dalam algoritma *Suffix tree Clustering*. Pada tahap ini, dokumen yang telah didapat dari proses download akan dibersihkan dan dipersiapkan untuk tahap selanjutnya. Proses untuk mempersiapkan dokumen meliputi proses pembersihan dokumen dari tag-tag HTML, proses analisa leksikal teks, proses penghapusan *stopword*, dan proses *stemming*.

***Stemming* dalam Bahasa Indonesiadengan Algoritma Porter**

Algoritma *stemming* untuk Bahasa Indonesia dengan nama *Porter Stemmer for Bahasa Indonesia* dikembangkan oleh Talla (2003) dan implementasi berdasarkan *English Porter Stemmer* yang dikembangkan oleh Frakes (1992). Beberapa modifikasi *Porter Stemmer* Bahasa Inggris telah dilakukan agar dapat digunakan untuk mengolah teks Bahasa Indonesia. Rancangan algoritma*Porter Stemmer for Bahasa Indonesia* dapat dilihat pada Gambar 1. Algoritma *Porter stemming for Bahasa Indonesia* bekerja berdasarkan struktur kata Bahasa Indonesia. Struktur kata pada Bahasa Indonesia terdiri dari *prefix*, *root*, *suffix*, *possessive_pronoun* dan *particle*. Struktur kata yang dimaksud ditulis dalam rangkaian opsional kata dalam format:

[*prefix1*]+[*prefix2*]+*root*+[*suffix*]+[*possessive_pronoun*]+[*particle*]



Gambar 1 Rancangan dasar *Porter Stemmerfor Bahasa Indonesia* (Talla, 2003)

Pada Gambar 1 terlihat beberapa langkah *removal* menurut aturan yang ada pada Tabel 1 sampai dengan Tabel 5. Dengan demikian algoritma *Porter Stemming for Bahasa Indonesia* terdapat 5 langkah pengecekan untuk melakukan proses *removal* sehingga mampu mendapatkan kata dasar (*root*) pada kata-kata teks Bahasa Indonesia.

Tabel 1 Peraturan pertama untuk *inflectional particles* (Talla, 2003)

Suffix	*Replacement*	*Measure Condition*	*Additional Condition*	*Examples*
kah	*NULL*	2	*NULL*	bukukah→buku
lah	*NULL*	2	*NULL*	dialah→dia
pun	*NULL*	2	*NULL*	bukupun→buku

Tabel 2 Peraturan kedua untuk *inflectional possesive pronouns* (Talla, 2003)

Suffix	*Replacement*	*Measure Condition*	*Additional Condition*	*Examples*
ku	*NULL*	2	*NULL*	bukuku→buku
mu	*NULL*	2	*NULL*	bukumu→buku
nya	*NULL*	2	*NULL*	bukunya→buku

Tabel 3 Peraturan untuk *first order of derivational prefixes* (Talla, 2003)

Suffix	*Replacement*	*Measure Condition*	*Additional Condition*	*Examples*
Meng	*NULL*	2	*NULL*	mengukur→ukur
Meny	s	2	v…	menyapu→sapu
Men	*NULL*	2	*NULL*	menduga→duga
Men	t	2	v…	menuduh→tuduh
Mem	p	2	v…	memilah→pilah
Mem	*NULL*	2	*NULL*	membaca→baca
Me	*NULL*	2	*NULL*	merusak→rusak
Peng	*NULL*	2	*NULL*	pengukur→ukur
Peny	S	2	v…	menyapu→sapu
Pen	*NULL*	2	*NULL*	penduga→duga

Pen	T	2	v…	penuduh→tuduh
Pem	P	2	v…	pemilah→pilah
Pem	*NULL*	2	*NULL*	pembaca→baca
Di	*NULL*	2	*NULL*	diukur→ukur
Ter	*NULL*	2	*NULL*	tersapu→sapu
Ke	*NULL*	2	*NULL*	kekasih→kasih

Tabel 4 Peraturan keempat untuk *second order of derivational prefixes* (Talla, 2003)

Suffix	*Replacement*	*Measure Condition*	*Additional Condition*	*Examples*
Ber	*NULL*	2	*NULL*	berlari→lari
Bel	*NULL*	2	ajar	belajar→ajar
Be	*NULL*	2	k*er..	bekerja→kerja
Per	*NULL*	2	*NULL*	perjelas-jelas
Pel	*NULL*	2	ajar	pelajar→ajar
Pe	*NULL*	2	*NULL*	pekerja→kerja

Tabel 5 Peraturan kelima untuk *derivational suffixes* (Talla, 2003)

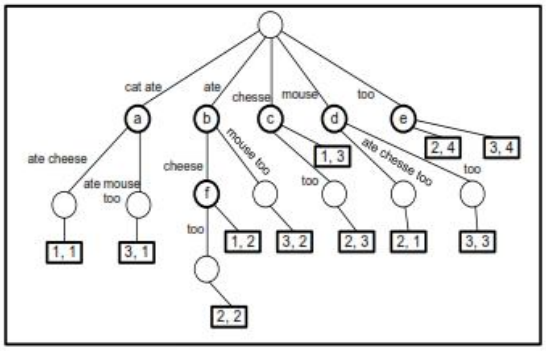
Suffix	*Replacement*	*Measure Condition*	*Additional Condition*	*Examples*
ka	*NUL*	2	prefix∉ {ke, peng}	Tarikkan →tarik (meng) ambilkan →ambil
n				
an	*NUL*	2	prefix∉ {di, meng, ter}	makanan →makan (per) janjian→janji
i	*NUL*	2	V|K…c_1c_2, $c_1 \neq s$, $c_2 \neq i$ and ∉ {ber, ke, peng}	Tandai→tanda (men) dapati→dapat warna→warnai

Identifikasi *BaseCluster*

Tahap identifikasi *base cluster* merupakan tahap terpenting dalam algoritma *suffix tree clustering*, karena pada tahap ini akan menghasilkan klaster-klaster dasar (Zamir & Etzioni, 1998). Pembentukkan *basecluster* dilakukan dengan cara menemukan share *phrase* antar dokumen. Untuk menemukan share *phrase* digunakan struktur data *suffix tree*. Dengan menggunakan struktur data ini, maka setiap dokumen akan direpresentasikan menjadi suatu kalimat. Untuk menemukan *base cluster* dapat

dilakukan dengan cara membuat suatu *invert index* dari *phrase* untuk semua dokumen.

Pembentukkan *suffix tree* untuk kalimat *cat ate cheese, mouse ate cheese too,* dan *cat ate mouse too* diperlihatkan pada Gambar 2 yang menunjukkan adanya internal node yang terbentuk. Setiap internal node merepresentasikan suatu kelompok dokumen dan share *phrase* untuk kelompok tersebut. Oleh karena itu, setiap internal node juga merepresentasikan *basecluster* yang terbentuk. Semua *base cluster* yang terbentuk dapat ditunjukkan pada Tabel 6.



Gambar 2 *Generate Suffix Tree* (Zamir & Etzioni, 1998)

Tabel 6. Base cluster yang terbentuk (Zamir & Etzioni, 1998)

Base Cluster	Phrase	Documents
a	cat ate	1, 3
b	ate	1, 2, 3
c	cheese	1, 2
d	mouse	2, 3
e	Too	2, 3
f	ate cheese	1, 2

Setiap *base cluster* yang terbentuk memiliki suatu *score*. Penghitungan score merupakan suatu fungsi dari jumlah dokumen yang masuk anggota *base cluster* dan jumlah kata yang menyusun *phrase* dari *base cluster*. Fungsi untuk menghitung skor *base cluster* ditunjukkan oleh Persamaan (1).

$$s(B) = |B|.f(|P|) \quad (1)$$

dimana pada Persamaan (1) :

$|B|$ adalah jumlah dokumen di dalam *base cluster B* dan

$|P|$ adalah jumlah kata yang menyusun frase *P*.

$f(|P| = 0$, jika $|P| = 1$ dan

$f(|P| = 6$, jika $|P| = 6$

Kombinasi *Base Cluster*

Tahap ini digunakan untuk menangani *overlaping cluster*. Dalam tahap ini, *phrase* tidak dipertimbangkan. Sebelum melakukan kombinasi antar *basecluster* (Zamir & Etzioni, 1998), kita harus menghitung dulu nilai *similiarity* antar *base cluster* yang didasarkan pada jumlah dokumen yang *overlap*. Adanya *overlaping* dokumen ini didasarkan karena dokumen memiliki lebih dari satu topik sehingga dokumen dapat memiliki lebih dari satu *shared phrase*.

Ukuran nilai *similiarity* menggunakan nilai biner. Rumus untuk menghitung nilai *similiarity* antar *base cluster* ditunjukkan pada Persamaan (2) dan (3)

$|Bm \cap Bn| / |Bm| > 0{,}5$ (2)

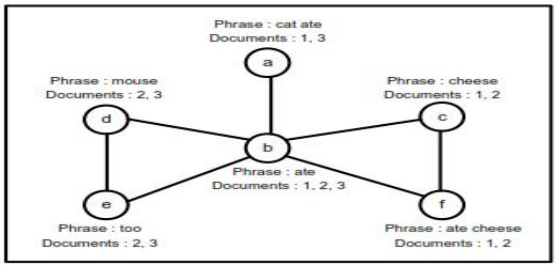
$|Bm \cap Bn| / |Bn| > 0{,}5$ (3)

dengan

$|Bm \cap Bn|$ adalah jumlah dokumen yang *overlap* terhadap *base cluster Bm* dan *Bn.*

$|Bm|$ dan $|Bn|$ masing-masing adalah jumlah dokumen dalam *base cluster Bm* dan *Bn.*

Dalam Persamaan 2 dan 3, menunjukkan penggunaan nilai *threshold* 0,5 karena nilai tersebut merupakan nilai tengah antara 0 sampai 1. Jika Persamaan (2) dan (3) bernilai benar maka *similiarity* akan bernilai 1 sehingga antara kedua *base cluster* tersebut akan terhubung. Jika salah satu dari Persamaan (2) dan (3) bernilai benar atau keduanya bernilai salah maka *similiarity* akan bernilai 0 sehingga antara kedua *base cluster* tersebut tidak terhubung. Keterhubungan antar *base cluster* dapat diceritakan dalam bentuk *base cluster graph* yang ditunjukkan pada Gambar 3



***Gambar 3** Base Cluster Graph* (Zamir & Etzioni, 1998)

Dari Gambar 3 menunjukkan bahwa antar *base cluster* terhubung sehingga dari 6 *base cluster* tersebut akan membentuk satu *cluster* tunggal. Untuk pembentukkan *cluster* digunakan algoritma *single link* dimana nilai *similiarity* minimal antar *base cluster* sebagai kriteria berhenti. Kita menggunakan algoritma ini, digunakan dalam domain *base cluster* dimana algoritma ini hanya menemukan keterhubungan antara *base cluster* yang terkecil.

Nearest Neighbor

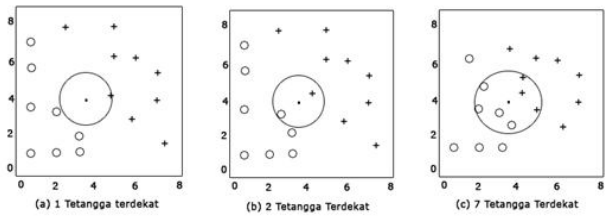
Nearest Neighbor merupakan suatu pengelompokkan suatu data baru berdasarkan jarak data baru ke beberapa data atau tetangga terdekat (Santosa, 2007:53). Menurut Kusrini dan Luthfi, (2009:94) *Nearest Neighbor* adalah suatu pendekatan untuk mencari kasus dengan menghitung kedekatan antara kasus baru dengan kasus lama, yaitu berdasarkan pada pencocokan bobot dari sejumlah fitur yang ada.

Algoritma *Nearest Neighbor*merupakan algoritma yang melakukan klasifikasi berdasarkan kedekatan lokasi (jarak) suatu data dengan data yang lain. Klasifikasi merupakan suatu pekerjaan menilai objek data untuk memasukkannya kedalam kelas tertentu dari sejumlah kelas yang tersedia.

Tujuan dari algoritma *Nearest Neighbor* adalah mengklasifikasikan obyek baru bedasarkan atribut dan *training sample*. Diberikan suatu titik *query*, selanjutnya akan ditemukan sejumlah *K* obyek atau (titik *training*) yang paling dekat dengan titik *query*. Nilai prediksi dari *query instance*akan ditentukan berdasarkan klasifikasi ketetanggaan.

Pada algoritma *Nearest Neighbor*, jarak antara satu data ke data yang lain dapat dihitung. Nilai jarak inilah yang digunakan sebagai nilai kedekatan atau kemiripan antara data uji dengan data latih. Nilai K pada *Nearest*

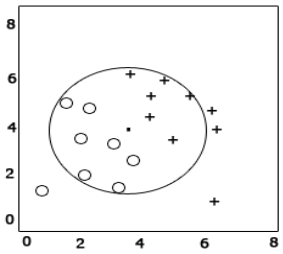
Neighbor berarti K-data terdekat dari data uji. Gambar 5 memberikan contoh algoritma *Nearest Neighbor*, tanda lingkaran untuk kelas 0, tanda plus untuk kelas 1. Seperti yang ditunjukan pada Gambar 3.4 (a), jika K bernilai 1, kelas dari 1 data latih sebagai tetangga terdekat (terdekat pertama) dari data uji tersebut akan diberikan sebagai kelas untuk data uji, yaitu kelas 1 . Jika K bernilai 2, akan diambil 2 tetangga terdekat dari data latih. Begitu juga jika nilai K adalah 3,4,5 dan sebagainya. Jika dalam K-tetangga ada dua kelas yang berbeda, akan diambil kelas dengan jumlah data terbanyak (voting mayoritas), seperti yang ditunjukan pada Gambar 3.4 (c). Pada Gambar Gambar 4 (c) terlihat bahwa kelas 0 mempunyai jumlah yang lebih banyak daripada kelas 1 sehingga kelas 1 akan dikategorikan kedalam kelas 0. Jika kelas dengan data terbanyak ada dua atau lebih, akan diambil kelas dari data dengan jumlah yang sama tersebut secara acak.



Gambar 4 *K- Nearest Neighbor* dengan nilai k-tetangga terdekat (Kusrini dan Luthfi, 2009)

Salah satu masalah yang dihadapi *Nearest Neighbor* adalah pemilihan nilai K yang tepat. Cara voting mayoritas dari K-tetangga untuk nilai K yang besar bisa mengakibatkan distorsi data yang besar, seperti yang ditunjukkan pada Gambar 3.5. misalnya diambil K bernilai 13, pada Gambar 3.5, kelas 0 dimiliki oleh 7 tetangga yang jauh, sedangkan kelas 1 dimiliki oleh 6 tetangga yang lebih dekat. Hal ini karena setiap tetangga mempunyai bobot yang sama terhadap data uji, sedangkan K yang terlalu kecil bisa

menyebabkan algoritma terlalu sensitif terhadap *noise*.



Gambar 5 *K- Nearest Neighbor* dengan nilai K yang besar (Kusrini dan Luthfi, 2009)

Metode Pengelompokkan Teks Menggunakan *Nearest Neighbor*

Pengkategorian teks Liao (2002) adalah proses pengelompokkan dokumen teks menjadi satu atau lebih kategori yang telah ditetapkan berdasarkan isi kontennya. Sejumlah teknik klasifikasi statistik dan mesin pembelajaran telah diterapkan untuk pengkategorisasian teks, diantaranya adalah*Regression Model, Bayesian Classifier, Decision Tree, Nearest Neighbor Classifier, Neural Networks,* dan*Support Vector Machines (SVM).*

Langkah pertama dalam pengkategorian teks adalah mengubah dokumen, yang biasanya merupakan string karakter, menjadi sebuah representasi yang cocok untuk algoritma pembelajaran dan tugas pengklasifikasian. Representasi dokumen yang paling umum digunakan adalah *Vector Space Model (VSM).* Dalam model ini, setiap dokumen direpresentasikan/diwakilkan oleh vektor kata-kata. Matriks A yang merupakan kata-berdasarkan-dokumen digunakan sebagai koleksi dokumen, contohnya $A = (a_{ij})$ dimana a_{ij} adalah bobot dari kata *i* pada dokumen *j*. Ada beberapa cara untuk menentukan bobot a_{ij}. Pada saat ini, berasumsikan f_{ij} merupakan frekuensi kata *i* pada dokumen *j*. *N* adalah jumlah dokumen dalam koleksi, *M* adalah jumlah kata yang berbeda dalam koleksi, dan n_i adalah total banyaknya kata *i* muncul di seluruh koleksi. Pendekatan yang paling sederhana adalah pembobotan *boolean*, yang menetapkan bobot dari a_{ij}

menjadi 1 jika kata tersebut muncul dalam dokukmen dan 0 untuk sebaliknya. Pendekatan lain yang sederhana adalah menggunakan frekuensi kata dalam dokumen, contohnya $a_{ij} = f_{ij}$. Pendekatan bobot yang lebih umum adalah pembobotan *tf - idf (term frequency - inverse document frequency)*:

$$a_{ij} = f_{ij} \times \log\left(\frac{N}{n_i}\right) \quad (4)$$

Sedikit variasi Kwok (1998) dari pembobotan *tf - idf* , yang memperhitungkan kemungkinan dokumen-dokumen memiliki panjang yang berbeda, adalah sebagai berikut :

$$a_{ij} = \frac{f_{ij}}{\sqrt{\sum_{i=1}^{M} f_{ij}^2}} \times \log\left(\frac{N}{n_i}\right) \quad (5)$$

Untuk matriks A, jumlah baris sesuai dengan jumlah kata M dalam koleksi dokumen. Mungkin ada ratusan ribu kata yang berbeda.Untuk mengurangi dimensi yang tinggi, penghapusan *stop-word* (kata-kata yang sering muncul yang tidak membawa informasi), *word stemming* (penghapusan *suffix*), dan teknik pengurangan dimensi tambahan, seleksi fitur atau re-parameterisasi, biasanya digunakan.

Untuk mengklasifikasi *class-unknown* pada dokumen X, algoritma pengklasifikasian *k-Nearest* mengurutkan/merangking dokumen tetangga diantara vektor-vektor dokumen pelatihan, dan menggunakan label kelas dari k-tetangga paling mirip untuk memprediksi kelas dokumen baru. Kelas-kelas tetangga ini dibobot dengan menggunakan persamaan dari masing-masing tetangga ke X, di mana persamaan diukur dengan jarak *Euclidean* atau nilai kosinus antara dua vektor dokumen. Persamaan kosinus didefinisikan sebagai berikut:

$$sim(X, D_j) = \frac{\sum_{t_i \in (X \cap D_j)} x_i \times d_{ij}}{\|x\|_2 \times \|D_j\|_2} \quad (6)$$

dimana *X* adalah dokumen tes, yang direpresentasikan sebagai suatu vektor; D_j adalah dokumen pelatihan ke *j*; t_i adalah kata yang berasal dari *X* dan D_j;

x_i adalah bobot dari kata t_i dalam *X*; d_{ij} adalah bobot dari kata t_i dalam dokumen D_j ; $\|X\|_2 = \sqrt{x_1^2 + x_2^2 + x_3^2 + \cdots}$ adalah normalisasi dari *X*, dan $\|D_j\|_2$ adalah normalisasi dari D_j.

Hasil dan diskusi

Proses perhitungan secara manual pada algoritma *Suffix Tree Clustering* (STC)

Pada proses ini, akan dijelaskan tahapan-tahapan pengelompokkan data dengan menggunakan sistem pengklasteran (*clustering*) dengan menggunakan algoritma *Suffix Tree Clustering* (STC).

Tahap pertama: Memfrase dokumen

Tahap pertama yang dilakukan adalah menyiapkan dokumen teks yang akan dikelompokkan menjadi klaster-klaster. Sebelum dokumen diproses lebih lanjut, dokumen tersebut harus melalui praproses yang meliputi; *tokenizing*, penghapusan *stop word* dan *stoplist* serta *stemming for Bahasa Indonesia*. Hasil praproses akan disimpan pada media penyimpanan yang ada karena akan digunakan juga pada proses klasifikasi. Pada kasus ini, dokumen-dokumen yang menjadi contoh adalah sebagai berikut:

D1: @mcsugm Sugeng bertemu dengan Edwinhttp://t.co/ri0IffQ7qe

D2: Rini menemui Edwin juga

D3: RT @mcsugm Sugeng menemui Rini juga

Tokenizing

Tujuan dari *tokenizing* adalah untuk memecah kalimat menjadi kata perkata secara terpisah yang dikenal dengan *term*. Hasil dari proses ini, adalah

D1: @mcsugm
Sugeng
bertemu
dengan
Edwin
http://t.co/ri0IffQ7qe

D2: Rini
menemui
Edwin
juga

D3: RT
@mcsugm
Sugeng
menemui
Rini
juga

Penghapusan *stopword*

Proses ini bertujuan untuk menghilangkan kata-kata yang tidak bermanfaat atau tidak mempengaruhi proses utama. Kata-kata yang masuk dalam kategori ini, diantaranya tanda baca, karakter dan kata khusus pada twitter yang telah didefinisikan sebagai *stopword*. Hasil dari proses ini, adalah

D1: Sugeng
bertemu
dengan
Edwin

D2: Rini
menemui
Edwin
juga

D3: Sugeng
menemui
Rini
juga

Proses penghapusan *stoplist*

Proses ini bertujuan untuk menghilakan kata-kata yang tidak bermanfaat atau tidak mempengaruhi proses utama. Kata-kata yang masuk dalam kategori ini, diantaranya tanda baca, karakter dan kata khusus pada twitter yang telah didefinisikan sebagai *stoplist*. Hasil dari proses ini, adalah

D1: Sugeng
bertemu
Edwin

D2: Rini
menemui
Edwin

D3: Sugeng
menemui
Rini

Stemming

Proses *stemming* bertujuan untuk mendapatkan kata dasar dari kata kerja yang telah mendapatkan imbuhan atau keterangan lainnya. *Porter Stemming for Bahasa Indonesia* merupakan algoritma *stemming* yang digunakan pada praproses. Pada implementasinya hasil *stemming* ini

dicek pada Daftar Kata Dasar yang ada. Hasil dari *stemming* ini, adalah

D1: Sugeng
temu
Edwin

D2: Rini
temu
Edwin

D3: Sugeng
temu
Rini

Secara keseluruha proses stemming dngan menggunakan algoritma *Porter Stemming for Bahasa Indonesia* dapat dilihat pada Gambar 1

Berdasarkan Gambar 1 yang menjelasakan 5 (lima) proses *stemming* yang terjadi pada algoritma *Porter stemming for Bahasa Indonesia*. Proses-proses yang ada berkaitan dengan penerapan 5 (lima) aturan yang ada pada algoritma ini. Algoritma *Porter stemming for Bahasa Indonesia* bekerja berdasarkan struktur kata Bahasa Indonesia. Struktur kata pada Bahasa Indonesia terdiri dari *prefix*, *root*, *suffix*, *possessive_pronoun* dan *particle*. Struktur kata yang dimaksud ditulis dalam rangkaian opsional kata dalam format:

[*prefix1*]+[*prefix2*]+*root*+[*suffix*] +[*possessive_pronoun*]+[*particle*]

Tahap kedua: Mengidentifikasi frase dokumen

Pada tahap ini, hal utama yang dilakukan adalah mentukan kata sebagai *base cluster* dan menentukan hubungannya dengan kata yang lainnya pada dokumen yang ada. Alat yang digunakan dalam proses ini, dalam mempresentasikan term-term yang ada pada setiap dokumen, adalah struktur data *Suffix Tree*.

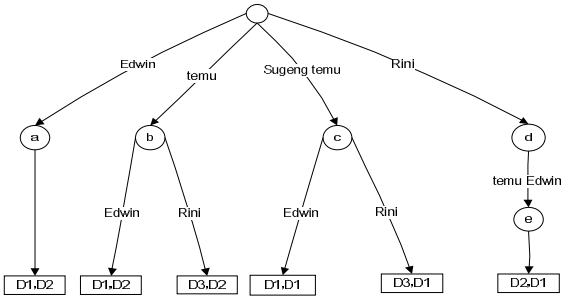
Dokumen-dokumen yang telah melalui praproses selanjutnya akan direpresntasikan dalam bentuk pohon *Suffix Tree*. Dokumen yang akan digunakan pada proses init, adalah

D1: Sugeng temu Edwin

D2: Rini temu Edwin

D3: Sugeng temu Rini

Hasil dari pemodelan dari ketiga dokumen tersebut, dapat dilihat pada Gambar 6



Gambar 6 Pembentukan *Suffix Tree*

Hasil analisis dari Gambar 4.2 dapat dilihat pada Tabel 7. Hasil ini, berupa rekapan keterkaitan antara simpul, frase yang ada dan dokumen yang terlibat.

Tabel 7 Hubungan antara Simpul dan Frase serta dokumen

No	Simpul	Frase	Dokumen	Skor
1	A	Edwin	D1, D2	2
2	B	Temu	D1, D2, D3	3
3	C	Sugeng temu	D1, D3	4
4	D	Rini	D2	1
5	E	temu Edwin	D1, D2	2

Nilai skor *s(B)* diperoleh dengan rumus, |*B*| dikali dengan *f(|P|)*. Nilai dari |B| merupakan jumlah dokumen dalam *base-cluster*, dan *f(|P|)* adalah jumlah kata yang menyusun frase P.

Tahap ketiga: Menggabungkan *Base Cluster*

Tahap ini, bertujuan menentukan klaster-klaster yang akan dibentuk dengan cara menggabungkan *base cluster* yang ada sesuai dengan aturan *similarity* yang ada. Penjelasan aturan ini, dapat dimulai dengan memperhatikan pada simpul a, terdapat 2 buah dokumen yang berhubungan, maka ditulis $|B_a|=2$, yaitu D1 dan D2. Sedangkan pada simpul b, terdapat 3 buah dokumen, sehingga ditulis $|B_b|=3$, yaitu D1, D2 dan D3. Kedua simpul, yaitu a dan b. Jika dioperasikan irisan himpunan maka dapat diperoleh dokumen yang berada di kedua simpul a dan b. Dengan demikian dapat diperoleh $|Ba \cap B_b|=2$, yaitu D1 dan D2 terdapat pada kedua simpul. Pada akhirnya nilai *similarity* kedua dokumen dapat

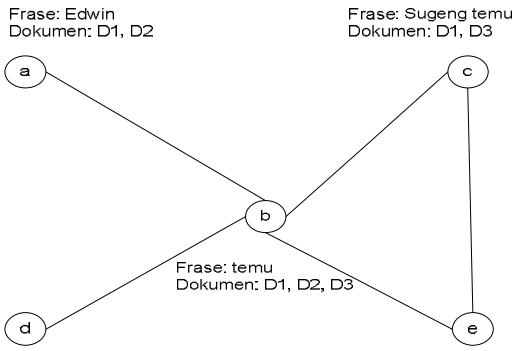
diketahui dengan menggunakan rumus untuk menentukan keterhubungan kedua simpul tersebut, dengan syarat sesuai Persamaan (2) dan (3).

Pada kasus ini, $|Ba \cap B_b|/|B_a|=2/2=1$, dan $|Ba \cap B_b|/|B_b|=2/3=0.67$ dengan demikiam kedua simpul memiliki *similarity*, jadi antara simpul a dan b berhubungan dan dapat berada dalam satu klaster atau kelompok. Rekap perhitungan dengan rumus *similarity* antar term-term pada dokumen yang ada, dapat dilihat hasil secara keseluruhan pada Tabel 8

Tabel 8 Nilai Similarity antar Simpul

Simpul	a	b	c	d	e
A	-	1.00	0.50	0.50	1.00
B	0.67	-	0.67	0.67	0.67
C	0.50	1.00	-	1.00	1.00
D	0.50	1.00	0.50	-	0.50
E	1.00	1.00	0.50	1.00	-

Tabel 8 menunjukan nilai *similarity* antar simpul-simpul yang ada. Sehingga nilai-nilai yang berkaitan ini, dapat direpresentasikan dalam bentuk graph seperti tertera pada Gambar 7



Gambar 7 Graph Klaster Frase

Rancangan Proses Perhitungan secara Manual pada Algoritma *Nearest Neighbor*

Perhitungan manual pada algoritma *Nearst Neighbor* dengan menggunakan dokumen teks anggota klaster *Other Topics* sebagai inputannya, akan dijelaskan secara detail yang meliputi deskripsi input, proses dan output yang dihasilkannya.

Kumpulan teks sebagai inputan untuk Algoritma Nearst Neighbor

Data yang digunakan pada proses klasifikasi dengan menggunakan algoritma *Nearest Neighbor,* merupakan hasil dari proses pengklasteran oleh pustaka *Carrot*[2] menggunakan algoritma *Suffix Tree Clustering* (STC).

Tabel 4.3 berisi kumpulan teks bebahasa Indonesia sebagai data inputan awal pada sistem pengelompokkan.

Tabel 9 Kumpulan data teks

No	Data
1	Sugeng bertemu dengan Edwin
2	Rini menemui Edwin juga
3	Sugeng menemui Rini juga
4	UGM singkatan dari Universitas Gadjah Mada

Tabel 9 Kumpulan data teks (lanjutan)

No	Data
5	UGM berlokasi di Yogyakarta
6	UGM mempunyai Jurusan Ilmu Komputer
7	Mahasiswa sering mendapatkan cemilan di Kantin
8	Mahasiswa sering menjelajah internet di perpustakaan
9	UGM memiliki akun twitter
10	Jurusan Ilmu Komputer memiliki akun twitter juga
11	Sugeng dan Rini pergi ke kantor setiap hari
12	Sugeng sedang duduk di atas kursi sekarang
13	Yogyakarta berlokasi di pulau Jawa
14	Muiz menyukai kopi
15	Muiz memiliki mobil baru
16	Muiz adalah seorang guru
17	Yoyok berbicara Bahasa Inggris dengan lancer
18	Yoyok bukan guru kursus Bahasa Inggris
19	Kuncoro telah pergi belanja kemarin
20	Yoyok berbicara Bahasa Inggris dan Sugeng berbicara Bahasa Jawa
21	Setelah Akas mendapatkan sarapan, dia pergi ke sekolah
22	Akas dan Ocha pergi ke Bandung
23	Cibaduyut berlokasi di Bandung
24	Akas dan Dika sedang bermain game
25	Ada pohon jeruk di kebunnya Akas
26	Akas telah dilahirkan di Bandung
27	Ocha telah dilahirkan di Bandung
28	Keukeu pernah ke kota Pemalang
29	Kota Pemalang berlokasi di Jawa Tengah
30	Ocha memiliki kucing dan ikan
31	Kucing memakan ikan
32	Tikus memakan ikan
33	Akas dan Ocha berbicara Bahasa Sunda
34	Akas pergi ke sekolah dengan mobil
35	Akas ingin menjadi dokter
36	Akas dan ibunya pergi ke Yogyakarta
37	Akas memiliki saudara
38	Yoyok dilahirkan di Yogyakarta
39	Yoyok memiliki akun twitter
40	Pemerintah Yogyakarta memiliki akun twitter dan facebook

Hasil proses pembentukan klaster oleh STC

Kumpulan data berjumlah 40 buah pada Tabel 10 diproses oleh algoritma *Suffix Tree Clustering* (STC) menghasilkan 4 klaster, termasuk klaster *other topics.* Hasil algoritma ini dapat dilihat pada Tabel 10,11, 12 dan Tabel 13

Tabel 10 Daftar anggota Klaster

C1 ”*Akas*”

No	Kode Dokumen	Teks
1	D19	Setelah Akas mendapatkan sarapan, diapergi ke sekolah
2	D20	Akas dan Ocha pergi ke Bandung
3	D22	Akas dan Dika bermain game
4	D24	Akas dilahirkan Bandung
5	D31	Akas dan Ocha berbicara Bahasa Sunda
6	D32	Akas pergi ke sekolah dengan mobil
7	D33	Akas ingin menjadi dokter
8	D34	Akas dan ibunya pergi ke Yogyakarta
9	D35	Akas memiliki saudara

Tabel 11 Daftar anggota Klaster C2

“*Akun twitter*”

No	Kode Dokumen	Teks
1	D6	UGM memiliki akun twitter
2	D7	Jurusan Ilmu Komputer memiliki akun twitter juga
3	D37	Yoyok memiliki akun twitter

Tabel 12 Daftar anggota Klaster C3

“*Yoyok*”

No	Kode Dokumen	Teks
1	D15	Yoyok berbicara Bahasa Ingris dengan lancar
2	D16	Yoyok bukan guru kursus Bahasa Inggris
3	D18	Yoyok berbicara Bahasa Inggris dan Sugeng berbicara Bahasa Jawa
4	D36	Yoyok dilahirkan di Yogyakarta
5	D37	Yoyok memiliki akun twitter

Tabel 13 Daftar anggota Klaster C4

“*Other Topics*”

No	Kode Dokumen	Teks
1	D0	Sugeng menemui Rini juga
2	D1	UGM berlokasi di Yogyakarta
3	D2	UGM memiliki Jurusan Ilmu Komputer
4	D3	UGM singkatan dari Universitas Gadjah Mada
5	D4	Mahasiswa sering mendapatkan cemilan di kantin
6	D5	Mahasiswa sering menjelajah internet di perpustakaan
7	D8	Sugeng bertemu dengan Edwin
8	D9	Sugeng and Rini pergi ke kantor setiap hari
9	D10	Sugeng sedang duduk di atas kursi sekarang
10	D11	Yogyakarta berlokasi di pulau Jawa

Tabel 13 Daftar anggota Klaster C4 "*Other Topics*" (lanjutan)

No	Kode Dokumen	Teks
11	D12	Muiz menyukai kopi
12	D13	Muiz memiliki mobil baru
13	D14	Muiz adalah seorang guru
14	D17	Kuncoro telah pergi belanja kemarin
15	D21	Cibaduyut berlokasi di Bandung
16	D23	Ada pohon jeruk di kebunnya Akas
17	D25	Ocha dilahirkan di Bandung
18	D26	Keukeu pernah ke Kota Pemalang
19	D27	Kota Pemalang berlokasi di Jawa Tengah
20	D28	Ocha memiliki kucing dan ikan
21	D29	Kucing memakan ikan
22	D30	Tikus memakan ikan
23	D38	Pemerintah Yogyakarta memiliki akun twitter dan facebook
24	D39	Rini menemui Edwin juga

b. Klaster tujuan klasifikasi berserta anggota

Kumpulan dokumen teks anggota klaster yang ada, selain klaster *other topics* dapat dilihat pada Tabel 14. Data pada Tabel 14 akan digunakan sebagai data *training* dalam proses perbandingan untuk setiap anggota klaster *other topics* terhadap dokumen teks yang ada pada setiap kelas sebagai kelas tujuan dari proses perpindahan dokumen anggota *other topics.*

Tabel 14 Daftar anggota klaster C1, C2 dan C3 hasil klasterisasi dengan STC

No	Kode Klaster	Kode Dokumen	Teks
1	C1	D19	Setelah Akas mempunyai sarapan, dia pergi ke sekolah
2	C1	D20	Akas dan Ocha pergi ke Bandung
3	C1	D22	Akas dan Dika bermain game
4	C1	D24	Akas dilahirkan di Bandung
5	C1	D31	Akas dan Ocha berbicara Bahasa Sunda
6	C1	D32	Akas pergi ke sekolah dengan mobil
7	C1	D33	Akas ingin menjadi dokter
8	C1	D34	Akas dan ibunya pergi ke

			Yogyakarta
9	C1	D35	Akas memiliki saudara
10	C2	D6	UGM memiliki akun twitter
11	C2	D7	Jurusan Ilmu Komputer memiliki akun twitter juga
12	C2	D37	Yoyok memiliki akun twitter
13	C3	D15	Yoyok berbicara Bahasa Inggris dengan lancar
14	C3	D16	Yoyok bukan guru kursus Bahasa Inggris
15	C3	D18	Yoyok berbicara Bahasa Inggris dan Sugeng berbicara Bahasa Jawa
16	C3	D36	Yoyok dilahirkan di Yogyakarta
17	C3	D37	Yoyok memiliki akun twitter

c. Proses klasifikasi dokumen teks anggota klaster *other topics*

Seandainya proses pengklaster dokumen teks dengan menggunakan STC menghasilkan 4 klaster termasuk klaster *other topics* dengan kode C4 dan seandainya juga jumlah anggota setiap masing-masing klaster yaitu C1 memiliki 4 anggota, C2 memiliki 3 anggota, klaster C3 memiliki 5, dan klaster C4 memiliki 24 anggota. Pada tahap klasifikasi akan dilakukan proses klasifikasi terhadap semua dokumen anggota klaster "*other topics"* yang berjumlah 24 untuk diklasifikasikan ke salah satu klaster C1, C2, C3 atau C4. Pada contoh kasus ini, akan melakukan percobaan proses pengklasifikasian terhadap salah satu dokumen anggota klaster *other topics* dengan kode D38 dan teks tersebut berisikan kalimat sebagai berikut "*Pemerintah Yogyakarta memiliki akun twitter dan facebook*". Pertanyaanya adalah "Apakah dokumen D38 ini akan diklasifikasikan ke klaster dengan label "Akas", "Akun Twitter" atau "Yoyok".

Langkah awal yang dilakukan untuk menjawab pertanyaan ini, adalah dengan terlebih dahulu menggabungkan dokumen teks D38 dengan anggota klaster C1, C2 dan C3 untuk diproses dalam perhitungan TF/IDF.

Tabel 15 merupakan kumpulan data yang telah melalui proses *pre-processing,* meliputi penghapusan *stopword* dan *stoplist,* dan proses *stemming* dengan menggunakan *Porter stemming for Bahasa Indonesia*. Hasil praproses yang digunakan dalam proses klasifikasi ini sebelum proses pengklasteran STC telah disimpan. Pada proses klasifikasi data praproses yang telah tersimpan akan dipergunkan kembali untuk dihitung dengan fungsi TF/IDF. Data hasil praproses dapat dilihat pada Tabel 15

Tabel 15 Hasil praproses

No	Kode Klaster	Kode Dokumen	Teks
1	**C4**	**D38**	**Pemerintah Yogyakarta miliki akun twitter facebook**
2	C1	D19	setelah Akas sarapan dia pergi sekolah

Tabel 15 Hasil praproses (lanjutan)

No	Kode Klaster	Kode Dokumen	Teks
3	C1	D20	Akas Ocha pergi Bandung
4	C1	D22	Akas Dika bermain game
5	C1	D24	Akas lahir Bandung
6	C1	D31	Akas Ocha bicara Sunda
7	C1	D32	Akas pergi sekolah mobil
8	C1	D33	Akas ingin dokter
9	C1	D34	Akas ibu pergi Yogyakarta
10	C1	D35	Akas milik saudara
11	C2	D6	UGM milik akun twitter
12	C2	D7	Jurusan Ilmu Komputer milik akun twitter
13	C2	D37	Yoyok milik akun twitter
14	C3	D15	Yoyok bicara Inggris lancar
15	C3	D16	Yoyok bukan guru kursus Bahasa Inggris
16	C3	D18	Yoyok

			bicara Inggris Sugeng bicara Jawa
17	C3	D36	Yoyok lahir Yogyakarta
18	C3	D37	Yoyok miliki akun twitter

d. Menghitung TF/IDF

Semua dokumen teks yang ada pada Tabel 15 dipisahkan (*tokenizing*) kata demi kata. Kata-kata yang ada diinventarisir agar tidak dicatat berulang kali, kemudian disebut dengan term tunggal. Term-term tunggal yang ada kemudian dihitung frekuensi kemunculan term pada setiap dokumen. Frekuensi kemunculan kemunculan term-term tunggal pada setiap dokumen, dikenal *term frequency (tf)* dan nilai frekuensi dokumen terhadap *tf*, yang dikenal dengan *document frequency (df)*. Sehingga nilai invers dari frekuensi setiap dokumen, yang dikenal dengan *invers document frequency* dapat dihitung dengan rumus log(n/df) dengan n adalah jumlah dokumen. Nilai-nilai tf, df, dan idf .

Langkah berikutnya, adalah menentukan bobot, dengan istilah *weight documentterm* (wdt). Nilai bobot berasal dari nilai idf untuk setiap term yang ada, kemudian dikalikan dengan nilai tf untuk setiap term di semua dokumen yang ada. Hasil operasi perkalian nilai idf dan nilai tf dari term-term yang ada dapat dilihat sebagai nilai bobot untuk semua term di setiap dokumen yang digunakan termasuk satu dokumen anggota teks berita klaster *other topics* sebagai *query* atau dokumen teks yang akan dibandingkan dengan dokumen teks yang ada pada klaster-klaster yang ada lainnya. Proses yang dilakukan setelah ditemukan nilai *idf* adalah penentuan nilai bobot dengan rumus $W_{dt} = tf_{dt} * idf_t$, *W* adalah *weight*, *d* adalah *document* dan *t* adalah term.

Nilai bobot dokumen pada salah satu anggota klaster *other topics*, dalam hal ini adalah D38 yang ada kemudian akan dikalikan dengan nilai bobot term-term untuk setiap dokumen yang ada

pada klaster lainnya. Perkalian nilai bobot term-term dokumen D38 dengan term-term angota klaster yang ada lainnya dapat ditentukan. Nilai bobot setiap term untuk setiap dokumen berdasarkan nilai tf dan idf, nilai bobot ini kemudian akan dihitung pada proses berikutnya untuk mendapatkan panjang vektornya.

e. Proses Perhitungan Panjang Vektor

Hasil perhitungan panjang vektor didapat dari hasil kuadrat bobot yang ada pada setiap dokumen, termasuk dokumen anggota klaster *other topics*. Pada proses pembobotan ini dilakukan dengan proses perkalian antara nilia tf dengan nilai idf untuk masing-masing term-term yang ada pada setiap dokumen yang dilibatkan pada proses penentuan nilai tf/idf. Hasil pembobotan terhadap term. Proses perhitungan selanjutnya berfungsi untuk mendapatkan nilai vektor semua dokumen anggota pada klaster yang ada, dengan mengkuadratkan nilai bobot yang ada pada setiap term dan dokumen yang ada. Proses perhitungan untuk mendapatkan nilai vektor dokumen D38 sebagai anggota *other topics* terhadap dokumen anggota klaster yang ada, dengan mengkuadratkan nili bobot yang ada pada setiap term dan dokumen yang ada.

Tabel 16 merupakan nilai akar rekap total nilai untuk setiap dokumen yang ada. Panjang vektor D_j didapat dengan melakukan penjumlahan panjang vektor untuk term-term yang ada disetiap dokumen dengan menggunakan fungsi SUM dan mencai nilai akar dari fungsi SUM untuk setiap dokumennya dengan rumus $|\vec{d_j}| = \sqrt{\sum_{i=1}^{t} W_{ij}^2}$, *d* adalah dokumen, *t* adalah term yang ada, *i* adalah indeks term dan *j* sebagai dokumen, sedangkan w adalah nilai bobot (*weight*).

Panjang vektor q dilakukan dengan menjumlahkan untuk term-term

yang ada dengan menggunakan fungsi SUM dan mencai nilai akar dari fungsi SUM untuk setiap dokumennya dengan rumus $|\vec{d}_j| = \sqrt{\sum_{i=1}^{t} w_{iq}^2}$, *d* adalah dokumen, *t* adalah term yang ada, *i* adalah indeks term dan *q* sebagai dokumen anggota klaster *other topics*, sedangkan w adalah nilai bobot (*weight*). Tabel 16 pada Kolom 1 (satu) menunjukan nilai vektor untuk setiap nilai term-term dokumen D38 sebagai query atau dokumen anggota *other topics.*

Tabel 16 Nilai Akar dari Summary

Panjang Vektor

No	Kode Dokumen	Nilai Similarity
1	**D38**	3.0745
2	**D19**	2.1811
3	**D20**	1.5864
4	**D22**	2.1947
5	**D24**	1.3825
6	**D31**	2.0376
7	**D32**	1.7837
8	**D33**	1.8004
9	**D34**	1.6961
10	**D35**	1.4055
11	**D6**	1.6189
12	**D7**	2.3119
13	**D37**	1.1282
14	**D15**	1.8217
15	**D16**	2.3578
16	**D18**	2.7615
17	**D36**	1.3203
18	**D37**	1.1282

f. Menghitung Kemiripan Dokumen Teks

Nilai-nilai yang ada digunakan untuk menghitung nilai kemiripan antar dokumen D38 salah satu anggota klaster *other topics* terhadap dokumen-dokumen anggota klaster yang ada, dengan menggunakan rumus *cosine similary*. Persamaan (6) merupakan rumus yang digunakan dalam perhitungan nilai kemiripan. Data nilai kemiripan dokumen-dokumen anggota klaster yang ada terhadap dokumen D38 anggota klaster *other topics*. Nilai kemiripan ini didapat berdasarkan rumus *similarity* Persamaan (6). Maka, diperoleh data sebagai berikut :

$$Sim\ (D_{19}, D_{38}) = \frac{0}{(3{,}0745 * 2{,}1811)} = 0$$

$$Sim\ (D_{20}, D_{38}) = \frac{0}{(3{,}0745 * 1{,}5864)} = 0$$

$$Sim\ (D_{22}, D_{38}) = \frac{0}{(3{,}0745 * 2{,}1947)} = 0$$

$$Sim\ (D_{24}, D_{38}) = \frac{0}{(3{,}0745 * 1{,}3825)} = 0$$

$$Sim\ (D_{31}, D_{38}) = \frac{0}{(3{,}0745 * 2{,}0376)} = 0$$

$$Sim\ (D_{32}, D_{38}) = \frac{0}{(3{,}0745 * 2{,}0745)} = 0$$

$$Sim\ (D_{33}, D_{38}) = \frac{0}{(3{,}0745 * 1{,}8004)} = 0$$

$$Sim\ (D_{34}, D_{38}) = \frac{0{,}9766}{(3{,}0745 * 1{,}6961)} = 0{,}187$$

$$Sim\ (D_{35}, D_{38}) = \frac{0{,}6982}{(3{,}0745 * 1{,}4055)} = 0{,}161$$

$$Sim\ (D_{6}, D_{38}) = \frac{2{,}2162}{(3{,}0745 * 1{,}689)} = 0{,}445$$

$$Sim\ (D_{7}, D_{38}) = \frac{1{,}3964}{(3{,}0745 * 2{,}3119)} = 0{,}196$$

$$Sim\ (D_{37}, D_{38}) = \frac{2{,}2162}{(3{,}0745 * 1{,}1282)} = 0{,}638$$

$$Sim\ (D_{15}, D_{38}) = \frac{0}{(3{,}0745 * 0)} = 0$$

$$Sim\ (D_{16}, D_{38}) = \frac{0}{(3{,}0745 * 2{,}3578)} = 0$$

$$Sim\ (D_{18}, D_{38}) = \frac{0}{(3{,}0745 * 2{,}7615)} = 0$$

$$Sim\ (D_{36}, D_{38}) = \frac{0{,}9766}{(3{,}0745 * 1{,}3203)} = 0{,}24$$

$$Sim\ (D_{37}, D_{38}) = \frac{2{,}2162}{(3{,}0745 * 1{,}1282)} = 0{,}638$$

Hasil perhitungan nilai kemiripan dokumen D38 sebagai dokumen *query* yang berada di klster *other topics* dibandingkan dengan dokumen yang ada pada klaster-klaster yang ada, dapat dilihat pada Tabel 17.

Tabel 17 merupakan daftar nilai *cosine similarity* dari dokumen-dokumen yang ada ada, terhadap dokumen D38 dan memiliki nilai lebih dari 0.

Tabel 17 Nilai *Cosine Similarity* dokumen lebih besar dari 0

C1	C2	C2	C2	C3	C3
D35	D6	D7	D37	D36	D37
0.161	0.445	0.196	0.638	0.24	0.638

g. Proses Klasifikasi dengan K-Nearest Neighbor

Tabel 17 merupakan daftar nilai *cosine similarity* sebanyak 7 dokumen (k=7) berdasarkan Tabel 18 dan telah diurutkan dari nilai terbesar sampai dengan nilai terkecil.

Tabel 18 Nilai Cosine Similarity secara terurut

C2	C3	C2	C3	C2	C1	C1
D37	D37	D6	D36	D7	D34	D35
0.638	0.638	0.445	0.24	0.196	0.187	0.161

Tabel 19 merupakan rekap dari hasil perhitungan dari banyaknya jumlah dokumen anggota klaster C1, C2 dan C3yang memiliki nilai kemiripan terhadap dokumen D38 salah satu

anggota klaster *other topics*. Pada klaster C1 ada 2 (dua) dokumen yang memiliki kemiripan dengan dokumen D38, klaster C2 terdapat 3 dokumen yang mirip dan klaster C3 hanya ada 1 dokumen yang anggota klasternya memiliki kemiripan dengan dokumen D38. Dengan demikian klaster C2 adalah klaster yang memiliki jumlah terbanyak dokumen anggotanya yang memiliki kemiripan dengan dokumen D38 dan kalster C2 ditetapkan sebagai klaster tujuan klasifikasi dari dokumen D38.

Tabel 18 Hasil Akhir

No	Kode Klaster	Jumlah Dokumen
1	C1	2
2	C2	3
3	C3	1

Hasil akhir dari proses pengklasifikasian ini adalah klaster C2 merupakan kode klaster yang direkomendasikan untuk dokumen D38 agar berpindah dari klaster *other topics* ke klaster dengan label *"Akun Twitter"*. Dengan demikian dokumen teks *other topics* dengan kode D38 yang berisi kalimat *"Pemerintah Yogyakarta memiliki akun twitter dan facebook"* menjadi anggota klaster dengan kode C2 yang bernama *"Akun Twitter"*.

Berdsarkan hasil proses klasifikasi anggota klaster *other topics* yang dapat diklasifikasikan ke klaster lainnya yang ada, secara teoritis dapat dibuktikan. Dengan dengan demikain proses klasifikasi dengan menggunakan algoritma *Nearest Neighbor* akan diimplementasikan untuk mengklasifikasi semua dokumen teks klaster *other topics* hasil dari proses klasterisasi dengan menggunakan algoritma *suffix tree clustering.*

Secara umum proses klasifikasi anggota klaster other topics dapat dilihat pada Gambar 8. Proses pengklasifikasian teks dokumen

anggota klaster *other topics* berdasarkan Gambar 8, dapat diketahui bahwa inputan berupa dokumen teks dari klaster-klaster yang ada dan dokumen teks klaster *other topics* yang disebut dengan *query* dengan simbol *q*. Semua dokumen teks yang ada harus dipenggal kata perkata agar dapat dikumpulkan semua kata-kata yang ada dan dicatat tanpa ada pengulangan yang kemudian disebut dengan *term*. Dokumen teks berasal dari hasil proses klasterisasi menggunakan algoritma *suffix tree clustering* dan telah mengalami praproses dokumen teks yang disimpan pada database.

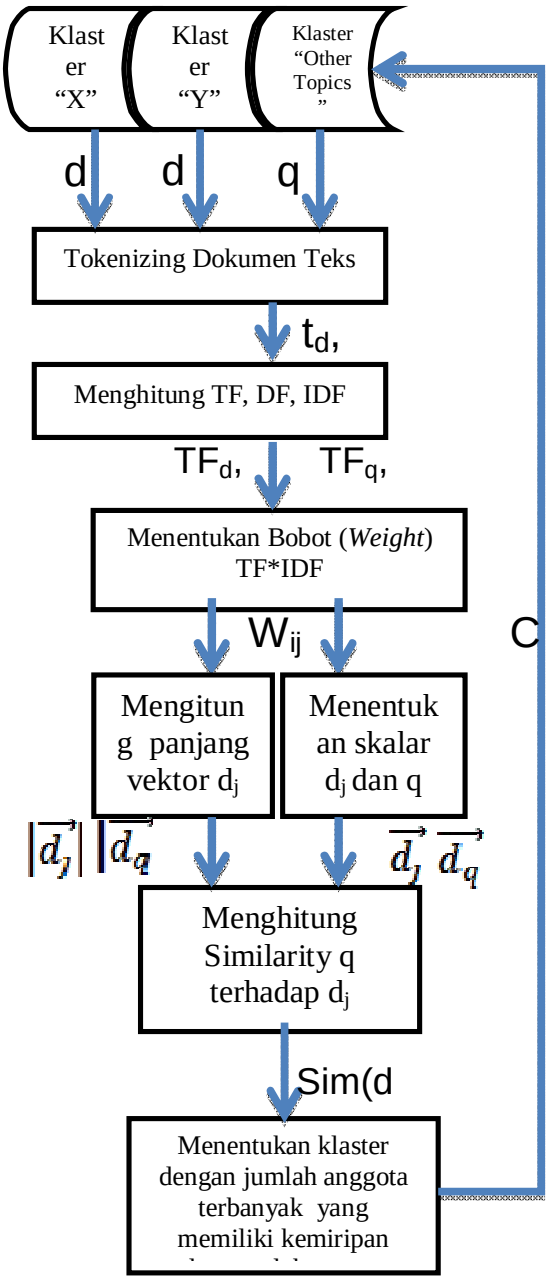
Term-term yang ada kemudian dihitung kemunculannya pada setiap dokumen yang ada. Jumlah kemunculan ini, disebut dengan *term frequency* (tf). Sedangkan jumlah dokumen yang mengandung term-term tertentu disebut dengan *document frequency* (df). Sehingga nilai *invers dokument frequency* (idf) dapat ditentukan dengan cara menghitung dengan fungsi *log* dari jumlah dokumen keseluruhan dibagi dengan nilai idf untuk setiap term-term yang ada.

Nilai-nilai idf yang dimiliki oleh term-term yang ada kemudian dikalikan dengan nilai tf, hasil dari perkalian ini disebut dengan bobot yang disimbolkan dengan *W*. Nilai bobot yang ada setiap term dan dokumen yang ada, dapat dicari nilai skalar dan panjang vektor dari dokumen klaster yang ada yang disimbolkan dengan d_j dan dokumen anggota klaster other topics yang disimbolkan dengan *q*.

Setelah didapat nilai skalar dan panjang vektor dokumen d*j* dan q, maka nilai-nilai ini digunakan untuk menghitung kemiripan dokumen d_j dengan dokumen q. Sehingga didapat nilai *similarity* antara dokumen d_j dan q dengan menggunakan Persamaan (6). Proses perbandingan untuk mengukur kemiripan ini, dilakukan secara berulang untuk semua dokumen klaster

other topics yang ada terhadap dokumen semua teks dokumen d*j* yang tersebar di klster-klaster yang ada.

Nilai similarity yang ada, diurutkan dari besar ke kecil, kemudian dihitung jumlah dokumen d_j pada setiap klaster. Klaster dengan jumlah banyaknya dokumen yang memiliki kemiripan ini, kemudian dijadikan sebagai klaster tujuan klasifikasi sehingga kode klaster dokumen *other topic* saat itu, yaitu q diubah kode klasternya ke klaster yang telah ditetapkan sebagai klaster tujuan perpindahan dokumen q. Hal ini dilakukan untuk semua anggota klaster other topics. Sehingga klaster *other topics* tidak memiliki anggota.



Gambar 8 Diagram Proses klasifikasi dokumen klster other topics

Kesimpulan

Proses pengklasteran teks berita dengan menggunakan Algoritma *Suffix*

Tree Clustering (STC) mampu mengkelompokkan berita secara tematik, tetapi masih banyak menghasilkan dokumen yang tak terkelompokkan. Sehingga dokumen tersebut, dikelompokkan ulang pada proses klasifikasi dengan menggunakan Algoritma *K-Nearest Neighbor* (K-NN) dan akhirnya K-NN mampu mengkelompokkan dokumen tersebut ke dalam kelompok yang ada.

Ucapan terima kasih

Penulis mengucapkan terima kasih kepada Bapak Edi Winarko atas kesediaanya dalam mensukseskan penelitian ini. Kemudian, dukungan yang telah penulis terima dari pihak Laboratorium Komputer JIKE FMIPA UGM.

Referensi

[1] Arifin, A. Z., Darwanto, R., Navastara, D. A., Ciptaningtyas, H. T., 2008, *Klasifikasi Online Dokumen Berita dengan Menggunakan Algoritma Suffix Tree Clustering*, Seminar Sistem Informasi Indonesia, ITS, Surabaya.

[2] Budhi, G. S., Gunawan I., Yuwono F., 2006, *Algoritma Porter Stemmer for Bahasa Indonesia untuk Pre-processing Text Mining Berbasis Metode Market Basket Analysis.*

[3] Cao G., Song D., Bruza P., 2003, *Suffix Tree Clustering on Post-retrieval Document*, Distributed System Technology Center, The University of Queensland

[4] Esko, U., 1995, *On-Line Construction of Suffix Trees.* In: Algorithmica, Vol. 14, No. 3., pp. 249-260.

[5] Farach. M., 1997, *Optimal Suffix Tree Construction with Large Alphabets,* In Proc. 38th Annual Symposium on Foundations of Computer Science , pages 137–143. IEEE

[6] Frakes, W.B., and Baeza R., 1992, *Information Retrieval, Data Structures and Algorithms.Prentice Hall.*

[7] Gusfield D., 1997, *Linear-Time Construction of Suffix Tree*, University of California, Cambridge University Press

[8] Kusrini dan Luthfi, E.T, 2009, *Algoritma Data Mining*, Andi Offset, Yogyakarta

[9] Kwok, J. T.-Y., 1998, *Automatic Text Categorization Using Support Vector Machine*, Proceedings of International Conference on Neural Information Processing, 347-351.

[10] Kwon O., Lee J., 2003, *Text Categorization Based on k-Nearest Neighbor Approach for Web Site Classification*, Elsevier Science Ltd.

[11] Liao Y., 2002, *Review of K-Nearest Neighbor Text Categorization Method*, https://www.usenix.org/legacy/events/sec02/full_papers/liao/liao_html/node4.htm, diakses 21 Agustus 2013

[12] Nagwani N. K. dan Verma S., 2011, *Software Bug Classification using Suffix Tree Clustering (STC) Algorithm*, IJCST vol. 2, Department of CS&E, National Institute of Technology Raipur

[13] Salton, G., 1983, *Introduction to Modern Information Retrieval*, McGraw-Hill BookCompany, New York.

[14] Sandi, F. R., 2012, *Klasifikasi Posting Twitter Kemacetan Lalu Lintas Kota Bandung Menggunakan Naïve Bayesian Classification*, Tesis, Program Studi S2 Ilmu Komputer, FMIPA, UGM.

[15] Santosa B., 2007, *Data Mining Teknik Pemanfaatan data untuk Keperluan Bisnis*, Graha Ilmu.

[16] Snowsill T., 2012, *Data Mining in Text Stream using Suffix Tree*, Disertasi, Jurusan

Matematika Teknik, Fakultas Teknik, Universitas Bristol, United Kingdom.

[17] Pribadi A.A., Martiana E., 2012, *Pencarian Judul TA menggunakan Text Mining dan Metode Suffix Tree,* Jurusan Tekknik Informatika, ITS

[18] Tala, F. Z., 2003, *A Study of Stemming Effects on Information Retrieval in Bahasa Indonesia*, Universiteit van Amsterdam, Netherland

[19] Weiss D., Osinki S., 2004, *Carrot^2 Clustering Framework,* Poznan University of Technology, Poznan Poland

[20] Weiner, P., 1973, *Linear pattern matching algorithms*, in Proceedings of the 14th Annual IEEE Symposium on Switching and Automata Theory, pp. 1–11,

[21] Wicaksono T., 2012, *Text Mining untuk Pencarian Dokumen Bahasa Inggris menggunakan Suffix Tree Clustering*, Jurusan Teknik Informatika, ITS

[22] Yang R., Xie H., dan Zhu Q., 2011, *Sentence-based Suffix Tree Clustering for Web Documents*, College of Computer Science, Chongqing University, Hongtao

[23] Zamir, O., Etzioni, O., 1998, *Web document clustering: a feasibility demonstration.* In: SIGIR 1998, pp. 46-54

Jumadi*
Informatics Department, Faculty of Science and Technology
UIN Sunan Gunung Djati Bandung
jumadi@uinsgd.ac.id

Edi Winarko
Computer Science and Electronics Department, Faculty of Mathematics and Natural Sciences
Universitas Gadjah Mada, Yogyakarta
edwin@ugm.ac.id

*Corresponding author