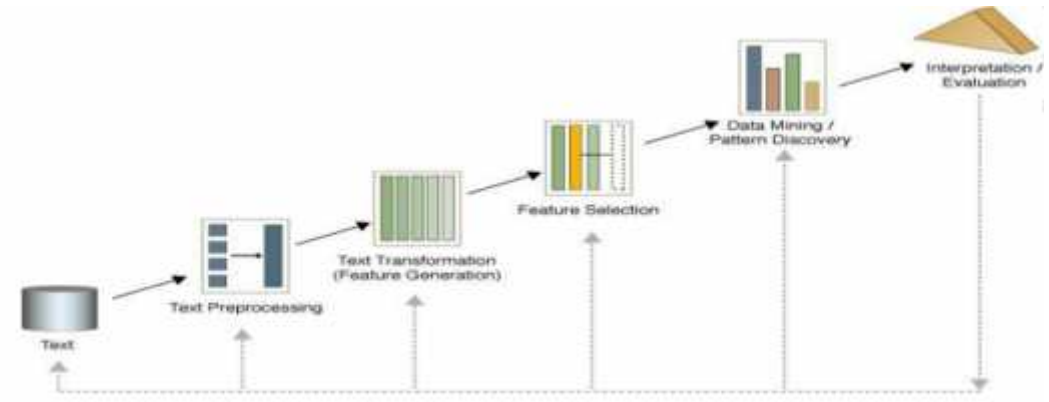
BAB II

LANDASAN TEORI

2.1 *Text Mining*

Text mining merupakan proses penambangan data (*mining*) yang berupa dokumen teks dengan tujuan mencari kata-kata yang dapat mewakili isi dari dokumen sehingga dapat dilakukan analisa keterhubungan/kesamaan antar dokumen. Permasalahan yang sering dihadapi dalam *Text mining* adalah jumlah data yang besar, *outlier*, dan banyaknya dimensi. Untuk itu *Text mining* memberikan solusi baru dalam hal pemrosesan, analisa, pengorganisasian, dan *clustering*/pengelompokkan dokumen teks dalam jumlah yang besar melalui beberapa tahap, yaitu tahap *text preprocessing*, *text transformation*, *feature selection*, *data mining* dan *evaluation* (Even dkk, 2002).



Gambar 2.1 Tahapan dalam *Text Mining* (Even dkk, 2002)

2.1.1 *Text Preprocessing*

Tahap ini adalah tahap dimana dilakukan proses pembersihan terhadap teks agar teks menjadi lebih terstruktur. Beberapa proses yang termasuk dalam tahap *text preprocessing* yaitu :

a. Pengubahan huruf besar/kapital menjadi huruf kecil (*case folding*).
b. Pembuangan/penghilangan tanda baca, simbol-simbol, dan karakter yang tidak memiliki arti penting terhadap dokumen.

2.1.2 *Text Transformation*

Setelah dilakukan tahap *preprocessing* terhadap dokumen selesai dilakukan, selanjutnya dilakukan tahap *text transformation* terhadap dokumen teks agar dapat diteruskan ke dalam tahap *feature selection*. Tahap ini adalah tahap dimana proses pemisahan kata dilakukan (*Tokenizing*) dan juga terdapat proses penghilangan imbuhan (*Stemming*).

Proses lain yang juga terdapat pada tahap ini yaitu transformasi hasil token menjadi bentuk tertentu, contoh menjadi kumpulan nilai *hash* atau *fingerprint*. Pada penelitian ini tahap *text preprocessing* seperti yang telah dijelaskan pada sub judul 2.1.1 dan tahap *text transformation* dilakukan dengan menggunakan tahapan yang ada pada algoritma *biword winnowing* yang merupakan pengembangan dari algoritma *winnowing*. Kedua algoritma ini adalah algoritma yang dapat menghasilkan informasi *fingerprint* dokumen teks yang penulis jadikan sebagai dasar dalam melakukan *clustering* terhadap dokumen teks. Pada penelitian ini, tidak diterapkan proses penghilangan imbuhan (*stemming*), karena tahap *text preprocessing* dan *transformation* telah dirangkul di dalam tahapan algoritma *biword winnowing* yang tidak menerapkan *stemming*.

2.1.2.1 Algoritma *Winnowing*

Algoritma *winnowing* merupakan urutan langkah-langkah yang dilakukan untuk menghasilkan *fingerprint* dari suatu dokumen (Scheilmer, 2003). Algoritma ini adalah salah satu algoritma yang dapat digunakan untuk mengekstrak *fingerprint* pada suatu dokumen yang berbasis *k-grams* atau *n-grams*. Input dari algoritma ini adalah dokumen teks yang diproses sehingga menghasilkan output berupa kumpulan nilai-nilai *hash* yang disebut dengan *fingerprint* (Muhammad Ridho, 2013). Beberapa langkah yang diterapkan dalam algoritma *winnowing* adalah sebagai berikut (Schleimer dkk, 2003) :

1. *Preprocessing* atau pembersihan teks yaitu tahap dimana pengubahan huruf besar/kapital menjadi huruf kecil, menghilangkan segala bentuk tanda baca atau simbol yang terdapat pada dokumen dan menghilangkan karakter yang tidak mempunyai makna/arti penting dalam dokumen.

Contohnya :

"The, claSsic. problem" menjadi "the classic problem"

2. Menerapkan metode *k-grams* yaitu tahap dimana dokumen dibentuk menjadi *substring* sepanjang *k* karakter dari sebuah string. Berikut hasil rangkaian *k-grams* dengan nilai k=5 :

   thecl hecla eclas class lassi assic ssicp sicpr icpro cprob probl roble oblem

3. Melakukan perhitungan nilai *hash* dari setiap *grams* yang telah terbentuk di atas. Adapun persamaan *hash* tersebut adalah :

$$H_{(c1\ldots ck)} = c_1*b^{(k-1)}+c_2*b^{(k-2)}+\ldots+c_{(k-1)}*b^{k}+c_k \qquad (2.1)$$

Keterangan :

c : nilai *ascii* karakter (desimal)

b : basis (bilangan prima)

k : banyak karakter (indeks karakter)

Untuk mendapatkan nilai *hash* dari metode *k-grams* selanjutnya digunakan persamaan *rolling hash* untuk mempercepat komputasi, contohnya sebagai berikut :

$$H_{(c2\ldots ck+1)} = H(c_1\ldots c_k)*b^{(k-1)}*b+c_{(k+1)} \qquad (2.2)$$

Dan Untuk mengetahui nilai ascii dari suatu karakter, dapat digunakan tabel ascii sebagai berikut :

Tabel 2.1 Tabel Ascii

Karakter	Nilai Ascii	Karakter	Nilai Ascii	Karakter	Nilai Ascii
A	65	a	97	0	48
B	66	b	98	1	49
C	67	c	99	2	50
D	68	d	100	3	51
E	69	e	101	4	52
F	70	f	102	5	53
G	71	g	103	6	54
H	72	h	104	7	55
I	73	i	105	8	56
J	74	j	106	9	57
K	75	k	107		
L	76	l	108		

M	77	m	109		
N	78	n	110		
O	79	o	111		
P	80	p	112		
Q	81	q	113		
R	82	r	114		
S	83	s	115		
T	84	t	116		
U	85	u	117		
V	86	v	118		
W	87	w	119		
X	88	x	120		
Y	89	y	121		
Z	90	z	122		

Berikut cara perhitungannya dengan b=3 dan k=5 :

thecl = ascii(t)*$3^{(5-1)}$+ascii(h)*$3^{(5-2)}$+ ascii(e)*$3^{(5-3)}$+ ascii(c)*$3^{(5-4)}$+ ascii(l)*$3^{(5-5)}$

= 116*$3^{(4)}$+104*$3^{(3)}$+101*$3^{(2)}$+99*$3^{(1)}$+105*$3^{(0)}$

= 13518

hecla = (13518-ascii(t)*$3^{(4)}$)*3+ascii(a)

= (13518-116*$3^{(4)}$)*3+97 = 12463

Nilai hash yang dihasilkan :

13518 12463 12232 12268 12852 12411 13774

13491 12639 12500 13551 13538 13021

4. Membentuk beberapa *window* dengan jumlah tertentu menggunakan konsep metode *k-grams*. Berikut *window* yang dihasilkan dari proses di atas dengan w=4 :

[13518, 12463, **12232**, 12268]
[12463, **12232**, 12268, 12852]
[**12232**, 12268, 12852, 12411]
[**12268**, 12852, 12411, 13774]
[12852, **12411**, 13774, 13491]
[**12411**, 13774, 13491, 12639]
[13774, 13491, 12639, **12500**]
[13491, 12639, **12500**, 13551]
[12639, **12500**, 13551, 13538]

[**12500**, 13551, 13538, 13021]

5. Memilih nilai *hash* minimum dengan posisi paling kanan dari setiap *window* yang telah terbentuk untuk dijadikan *fingerprint*. Adapun *fingerprint* yang terbentuk dari *window* di atas adalah :

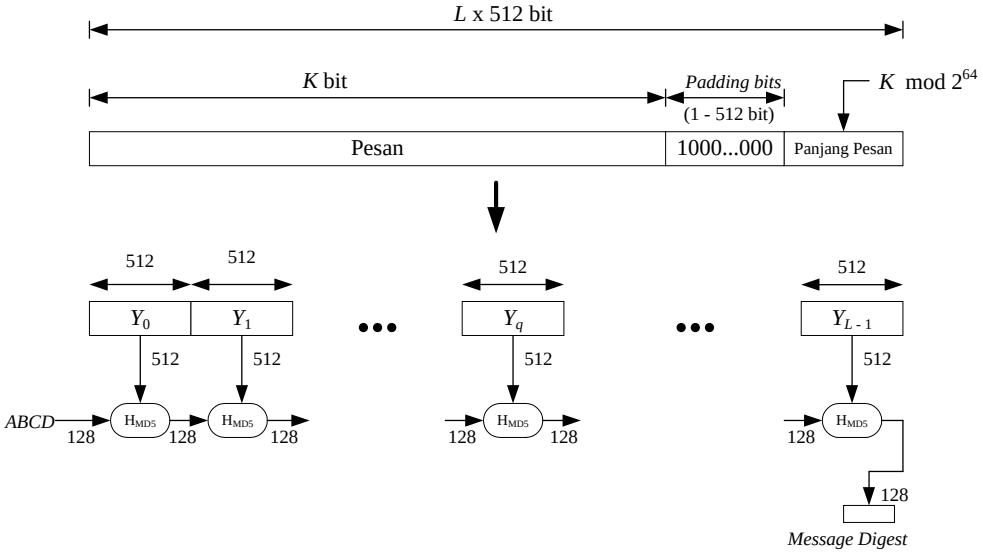
   [12232] [12268] [12411] [12500]

2.1.2.2 Algoritma *Biword Winnowing*

Algoritma *biword winnowing* adalah algoritma pengembangan dari algoritma *winnowing*. Tidak Seperti *winnowing* yang menerapkan konsep *k-grams*, algoritma ini menerapkan konsep *biword*, dimana *biword* sendiri merupakan suatu teknik *matching*/pencocokan pada proses pemisahan kata pada dokumen teks. Konsep *biword* menyerap konsep *phrase-based*, dimana pada tahap awal akan dilakukan konversi setiap dokumen untuk satu set bigram (dua kata).

a. MD5 (*Message Diggest* 5)

   MD5 adalah salah satu fungsi *hash* satu arah yang banyak digunakan dalam bidang kriptografi yang menerima input *string* yang panjangnya sembarang dan menghasilkan *message digest* dari input yang panjangnya tetap yakni sepanjang 128 bit atau 32 karakter.



Gambar 2.2 Pembuatan Atau Enkripsi MD5 Secara Garis Besar (Rinaldi Munir, 2013)

Tahap-tahap pembuatan *message digest* atau enkripsi MD5 secara garis besar adalah sebagai berikut (Rinaldi Munir, 2013) :

1. Penambahan Bit-Bit Pengganjal(*Padding Bits*)

   Di dalam tahapan ini, pesan di tambah dengan sejumlah bit pengganjal sedemikian rupa sehingga panjang pesan(dalam satuan bit) kongruen dengan 448 modulo 512. Jika panjang pesan adalah 448 bit, maka pesan tersebut ditambah 512 bit menjadi 960 bit. Jadi panjang bit-bit pengganjal adalah antara 1-512 bit, bit-bit pengganjal terdiri dari sebuah bit 1 diikuti dengan sisanya bit 0.

2. Penambahan Nilai Panjang Pesan

   Selanjutnya pesan yang telah diberi bit-bit pengganjal ditambah lagi dengan 64 bit yang menyatakan panjang pesan semula. Jika panjang pesan $> 2^{64}$ maka yang diambil adalah panjangnya dalam modulo 2^{64}. Dengan kata lain, jika panjang pesan semula adalah K bit, maka 64 bit yang ditambahkan menyatakan K modulo 2^{64}. Setelah ditambah dengan 64 bit, panjang pesan sekarang menjadi kelipatan 512 bit.

3. Inisialisasi Penyangga MD5

   Pada tahapan ketiga, MD5 membutuhkan 4 buah penyangga (*buffer*) yang masing-masing panjangnya 32 bit. Total panjang penyangga adalah $4 \times 32 = 128$ bit. Keempat penyangga ini menampung hasil antara dan hasil akhir. Keempat penyangga ini diberi nama A, B, C, dan D. Setiap penyangga diinisialisasi dengan nilai-nilai (dalam notasi HEX) sebagai berikut :

   A = 01234567

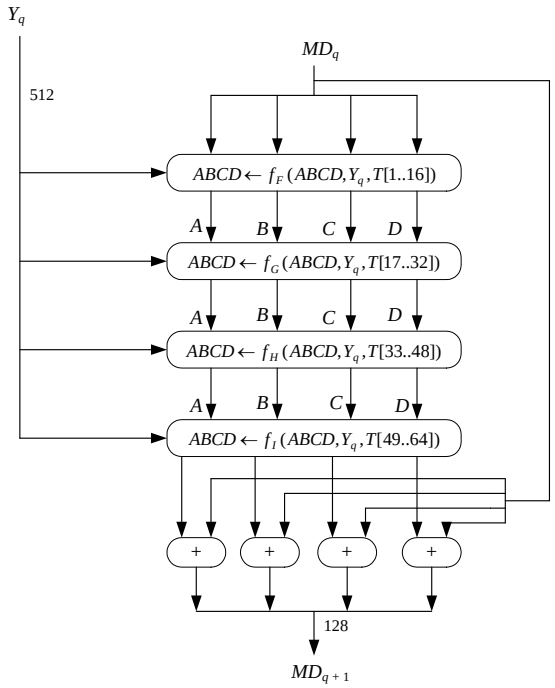
   B = 89ABCDEF

   C = FEDCBA98

   D = 76543210

4. Pengolahan Pesan Dalam Blok Berukuran 512 Bit

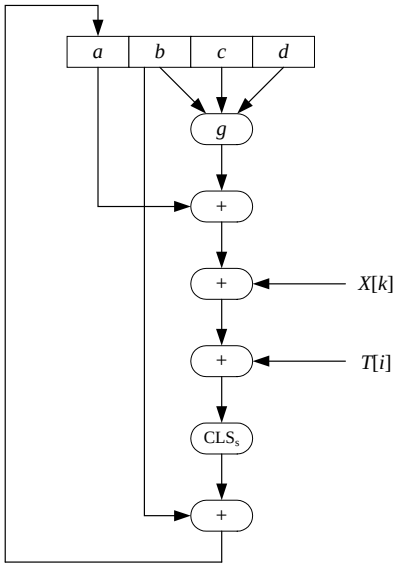
   Pada tahap ini, Pesan dibagi menjadi L buah blok yang masing-masing panjangnya 512 bit (Y_0 sampai Y_{L-1}), dimana setiap blok 512 bit diproses bersama dengan penyangga MD sehingga keluarannya adalah 128 bit, proses ini disebut proses H_{MD5}. Gambaran proses H_{MD5} adalah sebagai Berikut :



Gambar 2.3 Proses H_{MD5} (Rinaldi Munir, 2013)

Berdasarkan gambar 2.3, Y_q menyatakan blok 512 bit ke-q dari pesan (yang telah ditambah bit-bit pengganjal dan tambahan 64 bit nilai panjang pesan semula), dan MD_q adalah nilai *message digest* 128 bit dari proses H_{MD5} ke-q. Pada awal proses, MD_q berisi nilai inisialisasi penyangga MD.

Proses H_{MD5} terdiri dari 4 buah putaran, dimana masing-masing putaran melakukan operasi dasar MD5 sebanyak 16 kali dan setiap operasi dasar memakai sebuah elemen T. Jadi setiap putaran memakai 16 elemen Tabel T. berdasarkan gambar 2.3, fungsi-fungsi f_F, f_G, f_H, dan f_I masing-masing berisi 16 kali operasi dasar terhadap masukan, setiap operasi dasar menggunakan elemen Tabel T. Operasi dasar MD5 diperlihatkan pada gambar berikut :



Gambar 2.4 Operasi Dasar MD5 (Rinaldi Munir, 2013)

Keterangan variabel yang ada pada gambar 2.4 yaitu sebagai berikut :

a, b, c, d : Empat buah peubah penyangga 32 bit A, B, C, D

g : Salah satu fungsi F, G, H, I

CLS_s : *Circular left shift* sebanyak *s* bit

X[k] : Kelompok 32-bit ke-*k* dari blok 512 bit *message* ke-*q*, Nilai *k* = 0 sampai 15

T[i] : Elemen Tabel *T* ke-*i* (32 bit)

+ : Operasi penjumlahan modulo 2^{32}

Karena ada 16 kali operasi dasar, maka setiap kali selesai satu operasi dasar, penyangga-penyangga tersebut digeser ke kanan secara sirkuler dengan cara pertukaran sebagai berikut :

```
temp ← d
d ← c
c ← b
b ← a
a ← temp
```

Dan fungsi-fungsi dasar MD5 dijelaskan dalam tabel 2.2 sebagai berikut :

Tabel 2.2 Fungsi-Fungsi Dasar MD5

Nama	Notasi	g(b, c, d)
f_F	F(b, c, d)	$(b \wedge c) \vee (\sim b \wedge d)$
f_G	G(b, c, d)	$(b \wedge d) \vee (c \wedge \sim d)$
f_H	H(b, c, d)	$b \oplus c \oplus d$
f_I	I(b, c, d)	$c \oplus (b \wedge \sim d)$

Keterangan : Operator logika AND, OR, NOT, XOR masing-masing dilambangkan dengan $\wedge$, $\vee$, $\sim$, $\oplus$

Sedangkan nilai T[i], adalah sebagai berikut :

```
T[1] = D76AA478          T[17] = F61E2562
T[2] = E8C7B756          T[18] = C040B340
T[3] = 242070DB          T[19] = 265E5A51
T[4] = C1BDCEEE          T[20] = E9B6C7AA
T[5] = F57C0FAF          T[21] = D62F105D
T[6] = 4787C62A          T[22] = 02441453
T[7] = A8304613          T[23] = D8A1E681
T[8] = FD469501          T[24] = E7D3FBCB
T[9] = 698098D8          T[25] = 21E1CDE6
T[10] = 8B44F7AF         T[26] = C33707D6
T[11] = FFFF5BB1         T[27] = F4D50D87
T[12] = 895CD7BE         T[28] = 455A14ED
T[13] = 6B901122         T[29] = A9E3E905
T[14] = FD987193         T[30] = FCEFA3F8
T[15] = A679438E         T[31] = 676F02D9
T[16] = 49B40821         T[32] = 8D2A4C8A


T[33] = FFFA3942         T[49] = F4292244
T[34] = 8771F681         T[50] = 432AFF97
T[35] = 69D96122         T[51] = AB9423A7
T[36] = FDE5380C         T[52] = FC93A039
T[37] = A4BEEA44         T[53] = 655B59C3
T[38] = 4BDECFA9         T[54] = 8F0CCC92
T[39] = F6BB4B60         T[55] = FFEFF47D
T[40] = BEBFBC70         T[56] = 85845DD1
T[41] = 289B7EC6         T[57] = 6FA87E4F
T[42] = EAA127FA         T[58] = FE2CE6E0
T[43] = D4EF3085         T[59] = A3014314
T[44] = 04881D05         T[60] = 4E0811A1
T[45] = D9D4D039         T[61] = F7537E82
T[46] = E6DB99E5         T[62] = BD3AF235
T[47] = 1FA27CF8         T[63] = 2AD7D2BB
T[48] = C4AC5665         T[64] = EB86D391
```

Keluaran akhir dari algoritma MD5 adalah hasil penyambungan bit-bit di A, B, C, dan D. Misalkan G adalah isi sebuah teks uin.txt sebagai berikut :

```
Teknik informatika adalah jurusan terfavorit di universitas islam
negeri sultan syarif kasim riau. Hal ini terbukti dengan bertambahnya
jumlah mahasiswa teknik informatika dari tahun ke tahun.
```

Message digest dari teks uin.txt yang dihasilkan oleh algoritma *MD5* adalah 128-bit :

1010 1100 1001 0111 1100 0000 1010 0010 0100 0000 0110 1110 1110 1011 1110 0010 1010 0111 1111 0101 1111 0101 1111 0000 0010 1010 0000 0000 1000 1010 1010 0101

atau, dalam notasi HEX adalah:

ac97c0a2406eebe2a7f5f5f02a008aa5

b. Contoh Penerapan Algoritma *Biword Winnowing*

Langkah-langkah penerapan algoritma *biword winnowing* (Muhammad Ridho, 2013):

- *Preprocessing* atau pembersihan teks. Contoh :

  "The, claSsic. Problem IS just A Classic; problem" menjadi "the classic problem is just a classic problem"

- Memecah kalimat menjadi 2 kata. Contohnya :

  "the classic problem is just a classic problem "

  Menjadi :

```
the classic
classic problem
problem is
is just
just a
a classic
classic problem
```

- Melakukan proses enkripsi kata ke dalam MD5, agar kata dengan panjang karakter berbeda menjadi sama, yakni 32 karakter. Adapun hasil enkripsi *biword* di atas adalah :

```
[0] => fbae7b95f1da1e9e8e5667c7b1e50568
[1] => b041b9acdc7b08cb2ca0a002a249a289
[2] => 17fd1fc110aa4abf68768844ecf84ef0
[3] => cb6789f3b787f982365159bcf4bc05dc
[4] => d97d08ca02efb94c0785be08228b79cb
[5] => 0842898ad4699af11559cf026d24ac6e
[6] => b041b9acdc7b08cb2ca0a002a249a289
```

- Melakukan perhitungan nilai *hash* dengan menggunakan persamaan *hash* dan persamaan *rolling hash* seperti pada persamaan 2.1 dan 2.2, cara perhitungannya dapat dilihat pada contoh perhitungan nilai *hash* algoritma *winnowing* pada sub bab 2.1.2.1, Sehingga menghasilkan nilai *hash* sebagai berikut (nilai b=2 dan k=32) :

```
[0] => 421662169384
[1] => 326493898121
[2] => 264445942704
[3] => 378512521455
[4] => 348487189092
[5] => 220701029605
[6] => 326493898121
```

- Pembentukan *window-window* berdasarkan nilai *hash* yang dihasilkan. Adapun *window* yang terbentuk berdasarkan proses *hashing* di atas dengan w=4adalah :

  [421662169384, 326493898121, **264445942704**, 378512521455]
  [326493898121, 264445942704, 378512521455, 348487189092]
  [264445942704, 378512521455, 348487189092, 220701029605]
  [378512521455, 348487189092, **220701029605**, 326493898121]

- Mengambil nilai hash terkecil dari *window-window* yang telah terbentuk untuk dijadikan sebagai *fingerprint* dokumen. Adapun *fingerprint* yang terbentuk adalah [264445942704] dan [220701029605]

2.1.2.3 *Stemming*

Stemming adalah suatu proses yang di dalamnya terdapat langkah-langkah untuk mentransformasi kata-kata yang memiliki prefiks (awalan), sufiks (akhiran), dan konfiks (awalan dan akhiran) yang terdapat pada suatu dokumen teks menjadi bentuk kata dasarnya, contoh kata menghitung dan perhitungan menjadi hitung. Penggunaan *stemming* berbeda pada setiap bahasa karena perbedaan morfologi yang terdapat pada masing-masing bahasa. *Stemming* dapat dilakukan dengan menggunakan beberapa algoritma *stemming*, di antaranya algoritma Porter, algoritma Nazief dan Andriani, dan Porter *Stemmer For* Bahasa Indonesia. Berikut penjelasan mengenai algoritma-algoritma tersebut :

1. Algoritma Porter

   Algoritma Porter dikembangkan oleh Martin Porter pada tahun 1980 dan sering digunakan pada *stemming* bahasa inggris. Ide awal dari algoritma ini adalah bahasa inggris yang hanya mengenal penggunaan sufiks (akhiran), contoh sufiks *ing* pada kata *buying* atau *es* pada kata *catches*. Maka algoritma porter adalah algoritma yang berisi urutan langkah yang berfungsi menghilangkan sufiks pada kata berbahasa inggris tersebut, sehingga *buying* menjadi *buy* dan *catches* menjadi *catch*.

2. Porter *Stemmer For* Bahasa Indonesia

   Stemmer ini dikembangkan oleh W.B. Frakes pada tahun 1992. Porter *Stemmer For* Bahasa Indonesia dikembangkan karena struktur morpologi yang berbeda antara bahasa Indonesia dan bahasa inggris. Bahasa inggris hanya mengenal sufiks, sedangkan bahasa Indonesia mengenal prefiks, sufiks, dan konfiks. Artinya, *stemming* bahasa Indonesia lebih sulit dibanding *stemming* bahasa inggris.

3. Algoritma Nazief dan Andriani

   Selain dengan Porter Stemmer For Bahasa Indonesia, *stemming* bahasa Indonesia juga dapat dilakukan dengan menggunakan algoritma Nazief dan Andriani. Algoritma ini umum digunakan pada *stemming* bahasa Indonesia yang berisi urutan langkah yang berfungsi untuk menghilangkan prefiks, sufiks, dan konfiks yang terdapat pada kata berbahasa Indonesia. Algoritma ini menggunakan kamus kata dasar dalam implementasinya. Semakin lengkap kamus kata dasar tersebut, maka semakin berkualitas pula hasil stemming yang didapat.

2.1.3 *Feature Selection*

Feature selection adalah tahap pemilihan *feature*/ciri dari sebuah dokumen teks. *Feature selection* berbicara mengenai bagaimana cara merepresentasikan sebuah dokumen (*document representation*) dan cara menghitung jarak atau *similarity* antar dokumen dengan menggunakan beberapa fungsi. Pada tahap ini pembentukan pola yang dapat merepresentasikan sebuah dokumen teks dilakukan. Pola tersebut dapat ditemukan dengan banyak cara, salah satunya dengan menghitung frekuensi kemunculan suatu kata tertentu. Selanjutnya frekuensi kemunculan kata tersebut akan mengisi tempat sebagai dimensi dokumen dimana masing-masing dokumen dianggap sebagai vektor dalam ruang kata, lalu dilanjutkan dengan proses penghilangan atau pembuangan (reduksi) dimensi yang dianggap tidak penting pada dokumen.

2.1.3.1 *Document Representation*

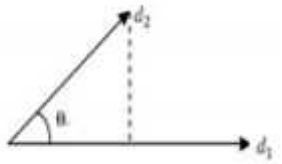
Ada banyak cara yang dapat digunakan untuk merepresentasikankan sebuah dokumen teks. Salah satu cara yang dapat digunakan adalah

merepresentasikan sebuah dokumen teks sebagai sebuah *bag of word*/sekumpulan kata, dimana cara ini banyak digunakan dalam *text mining*. Masing-masing kata yang terdapat pada dokumen merepresentasikan dimensi dalam ruang kata. Sedangkan masing-masing dokumen di dalam sebuah koleksi dokumen menjadi vektor yang memiliki nilai *non-negative* pada masing-masing dimensi (Anna Huang, 2008).

Penelitian ini menyerap konsep *bag of word* ini, dimana konsep kata pada *bag of word* digantikan dengan penggunaan *fingerprint* dokumen yang dihasilkan dari rangkaian proses dalam algoritma *biword winnowing*. Sehingga frekuensi kemunculan *fingerprint*-lah yang merepresentasikan dimensi pada masing-masing dokumen.

Sebagai contoh D = {d1, . . . , dn} adalah sekumpulan dokumen dan F = {f1, . . . ,fm}adalah sekumpulan *fingerprint* berbeda yang terdapat pada dokumen d pada koleksi dokumen D. Maka dapat diartikan bahwa sebuah dokumen direpresentasikan sebagai sebuah vektor berdimensi m, yang dapat dituliskan dengan D_d = (ff(d, f1), . . . , ff(d, fm)) dimana ff(d, f) adalah frekuensi kemunculan *fingerprint* f ∈ F pada dokumen d ∈ D.

Dengan merepresentasikan sebuah dokumen sebagai sebuah vektor, berarti yang perlu dilakukan adalah menghitung derajat kemiripan antara 2 dokumen sebagai korelasi antara vektor yang bersesuaian untuk menentukan jarak (ketidaksamaan) atau similaritas (kesamaan) 2 dokumen tersebut. Gambar 2.4 memperlihatkan sudut antara 2 dokumen pada ruang 2 dimensi.



Gambar 2.5 Sudut Antar Dokumen (Anna Huang, 2008)

Pada gambar 2.5 hanya memperlihatkan sudut antara 2 dokumen pada ruang 2 dimensi. Tapi pada prakteknya, ruang vektor atau dokumen mempunyai puluhan atau bahkan ribuan dimensi. Sehingga proses reduksi untuk mengurangi jumlah dimensi sangat diperlukan agar proses perhitungan similaritas, jarak dan *clustering* dokumen menjadi lebih cepat. Pada penelitian ini, proses dilakukan

dengan cara menghilangkan dimensi yang frekuensi kemunculannya lebih kecil dari 50% pada koleksi dokumen.

2.1.3.2 Fungsi Jarak Dan Similaritas Antar Dokumen

Sebelum masuk ke tahap *clustering*, perhitungan similaritas dan jarak harus dilakukan terlebih dahulu. *Output* dari perhitungan tersebut adalah nilai jarak antara 2 objek atau dokumen, karena itu sebuah perhitungan jarak mutlak dilakukan (Anna Huang, 2008). Untuk menghitung kesamaan atau *simililarity* antar dokumen dapat diukur dengan suatu fungsi similaritas atau jarak. Beberapa fungsi similaritas dan fungsi jarak antara lain :

1. *Jaccard Coefficient*

   Jaccard coefficient merupakan fungsi untuk menentukan tingkat kesamaan/simlaritas antar dua dokumen. *Jaccard coefficient* didefenisikan melalui persamaan berikut :

   $$\text{Sim}\ (D_i,D_j) = \frac{2\sum_{k=1}^{d} D_{ik} * D_{jk}}{\sum_{k=1}^{d} Dik + \sum_{k=1}^{d} Djk - \sum_{k=1}^{d} D_{ik} * D_{jk}} \quad (2.3)$$

2. *Euclidian Distance*

   Eucliadian distance adalah fungsi jarak yang banyak digunakan dalam permasalahan *clustering*, terkecuali dokumen teks. Ia dapat bekerja dengan baik pada ruang 2 atau 3 dimensi (Anna Huang, 2008). Adapun persamaan *Euclidian distance* didefenisikan sebagai berikut :

   $$\text{Dis}\ (D_i,D_j) = \sqrt{\sum_{k=1}^{d} (D_{ik} - D_{jk})^2} \quad (2.4)$$

3. *Cosine Similarity*

   Cosine Similarity merupakan salah satu fungsi similaritas yang paling banyak digunakan dalam *clustering*. *Output* dari perhitungannya adalah nilai similaritas antara 2 dokumen, contoh d_1 dan d_2. Semakin besar nilai similaritasnya, maka semakin dekat 2 dokumen tersebut. *Cosine similarity* didefenisikan melalui persamaan berikut :

   $$\text{Sim}\ (D_i,D_j) = \frac{\sum_{k=1}^{d} D_{ik} D_{jk}}{\sum_{k=1}^{d} (D)_{ik}{}^2 * \sum_{k=1}^{d} (D)_{jk}{}^2} \quad (2.5)$$

4. *Dice*

Dice adalah salah satu fungsi similaritas yang juga dapat digunakan dalam menentukan kesamaan antara 2 dokumen. Persamaannya didefenisikan sebagai berikut :

$$\text{Sim}\ (D_i,D_j) = \frac{2\ \sum_{k=1}^{d} D_{ik} D_{ik}}{\sum_{k=1}^{d} D_{ik} + \sum_{k=1}^{d} D_{jk}} \qquad (2.6)$$

Keterangan :

D_{ik} : *Fingerprint Frequency* k pada dokumen i

D_{jk} : *Fingerprint Frequency* k pada dokumen j

d : Dokumen

k : *Fingerprint Frequency*

Dari beberapa fungsi similaritas dan jarak di atas, pada penelitian ini penulis memilih persamaan *cosine similarity* untuk menentukan kesamaan antar dokumen. Fungsi *cosine similarity* dipilih karena menurut Strehl dkk (2000), fungsi ini merupakan fungsi *similarity* yang paling baik untuk tujuan *clustering* dokumen.

2.1.4 *Data Mining*

Setelah tahap pembentukan dan reduksi vector selesai, selanjutnya dilakukan *clustering* terhadap dokumen teks untuk menemukan pengetahuan baru pada dokumen teks. Ada banyak metode yang dapat dilakukan dalam melakukan *clustering* sebuah koleksi dokumen teks. Diantaranya *K-Means*, *Bisecting K-Means*, *centroid clustering*, dan lain sebagainya. Pada penelitian ini penulis memilih menggunakan metode *clustering K-Means* karena kemudahan dalam implementasinya dan waktu komputasinya yang cepat.

2.1.4.1 *Clustering* Dokumen Dan Metodenya

Clustering dokumen atau dikenal juga dengan pengelompokkan dokumen merupakan suatu cara untuk mengelompokkan dokumen-dokumen yang berada pada suatu koleksi dokumen ke dalam beberapa kluster/kelompok sesuai dengan kemiripan isi dari masing-masing dokumen yang ada pada koleksi dokumen tersebut. Dengan adanya teknik *clustering* dokumen dapat mendukung proses

pencarian yang lebih efisien, karena dokumen-dokumen yang telah dikelompokkan akan lebih mudah untuk ditemukan. Terlebih lagi jika dokumen-dokumen tersebut berada dalam koleksi dokumen yang besar.

Dalam *clustering* dokumen, dokumen yang berada pada satu kluster adalah dokumen-dokumen yang memiliki tingkat kesamaan yang tinggi satu sama lainnya dan dokumen yang berada pada kluster yang berbeda adalah dokumen yang tidak memiliki kesamaan (Christopher D. Manning dkk, 2009). Tingkat kesamaan ini dapat diukur atau diidentifikasi berdasarkan ciri pada masing-masing dokumen. Ciri tersebut dapat berupa kemunculan kata-kata khusus atau *fingerprint* dokumen.

Teknik *fingerprint* digunakan untuk menentukan keakuratan kesamaan antara 2 dokumen dalam mendeteksi adanya tindakan plagiat atau dokumen yang memiliki kesamaan isi. Pada kasus *clustering*, dokumen-dokumen yang ada dalam suatu koleksi dokumen dikelompokkan berdasarkan keidentikan *fingerprint* antara satu dokumen dengan dokumen lainnya dengan menggunakan suatu fungsi similaritas tertentu. Dokumen-dokumen yang memiliki *fingerprint* yang identik/sama akan berada pada 1 kluster, sedangkan dokumen-dokumen dengan *fingerprint* yang jauh berbeda berada pada kluster lain.

Tersedia banyak metode yang bisa digunakan untuk melakukan *clustering* terhadap dokumen. Secara umum metode-metode tersebut terbagi menjadi 2 karakteristik, yaitu *partitioning clustering* dan *hierarchical clustering*. Konsep dasar dari *partitioning clustering* adalah mempartisi koleksi dokumen menjadi k buah kluster pada langkah awal pengklusteran dimana jumlah kluster ditentukan oleh user, metode yang termasuk ke dalam karakteristik ini yaitu *K-Means*. Sedangkan konsep dasar *hierarchical clustering* adalah dengan membangun beberapa buah kluster dan mempartisinya secara rekursif yaitu secara *top-down* (*divisive*) atau *bottom-up* (*aglomerative*), jadi user tidak perlu menentukan jumlah kluster yang akan dibentuk.

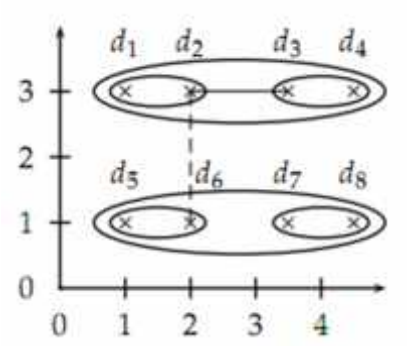
2.1.4.2 *Hierarchical Aglomerative Clustering*

Konsep dasar metode ini adalah dengan menganggap semua dokumen yang ada pada koleksi dokumen sebagai kluster. Kemudian dengan menggunakan fungsi similaritas antar kluster, proses penggabungan kluster dilakukan (Amir

Hamzah, 2009). Beberapa metode yang termasuk ke dalam kelompok ini adalah *Single Linkage Clustering*, *Complete Linkage Clustering*, *Group Average Agglomerative Clustering*, *Centroid Clustering* (Christopher D. Manning dkk, 2009). Penjelasan mengenai metode-metode tersebut, yaitu :

1. *Single Linkage Clustering*

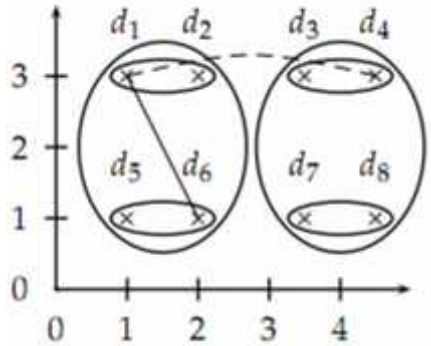
   Merupakan metode *clustering* dimana similaritas antara 2 kluster ditentukan oleh tingkat similaritas yang tertinggi (*most similar*). Artinya dokumen yang berada dalam 1 kluster adalah dokumen dengan jarak terdekat. Gambar 2.6 memperlihatkan cara metode *Single Linkage Clustering* dalam menentukan similaritas antar 2 dokumen. Dimana similaritas antara d_2 dan d_3 (garis tebal) lebih besar atau tinggi dibanding similaritas antara d_2 dan d_6(garis putus-putus), maka d_2 dan d_3 berada dalam 1 kluster.



Gambar 2.6 *Single Linkage Clustering* (Christopher D. Manning dkk, 2009)

2. *Complete Linkage Clustering*

   Metode ini adalah kebalikan dari metode *Single Linkage Clustering*. *Complete Linkage Clustering* menentukan similaritas antara 2 kluster berdasarkan tingkat similaritas yang terkecil (*most dissimilar*). Artinya dokumen yang berada dalam 1 kluster adalah dokumen dengan jarak terjauh. Kelemahan dari metode ini adalah sifatnya yang sangat sensitif terhadap outliers (dokumen atau vektor yang memiliki banyak kata-kata atau elemen yang tidak relevan). Gambar 2.7 memperlihatkan cara metode *Complete Linkage Clustering* dalam menentukan similaritas antar 2 dokumen. Dimana similaritas antara d_1 dan d_4 (garis tebal) lebih kecil atau rendah dibanding similaritas antara d_1 dan d_6(garis putus-putus), maka d_1 dan d_6 berada dalam 1 kluster.



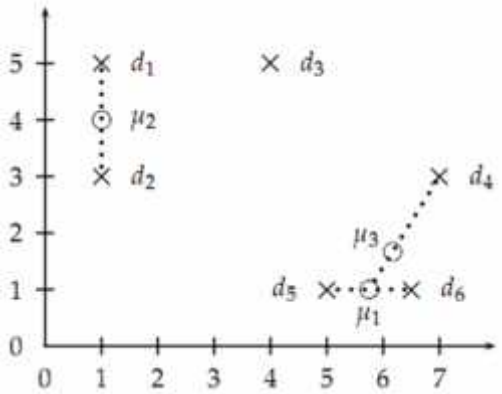
Gambar 2.7 *Complete Linkage Clustering* (Christopher D. Manning dkk, 2009)

3. *Group Average Agglomerative Clustering*

   Metode yang juga dikenal dengan *group average clustering* atau *average link clustering* ini menentukan similaritas antar dua kluster yang diukur dengan rata-rata hitung similaritas antar seluruh pasangan titik antara dua kluster.

4. *Centroid Clustering*

   Pada *centroid clustering* similaritas antara 2 kluster didefenisikan berdasarkan *centroid* 2 kluster tersebut. Artinya 2 kluster yang memiliki *centroid* dengan tingkat similaritas terdekat berada dalam 1 kluster. Metode *clustering* ini diawali dengan mengganggap semua dokumen yang berada pada suatu koleksi dokumen sebagai kluster, lalu dilakukan perhitungan jarak *centroid* antar kluster. Proses *clustering* berhenti jika jumlah kluster yang diiginkan telah terbentuk. Gambar 2.8 memperlihatkan 3 iterasi yang terdapat pada *centroid clustering* dalam melakukan *clustering* dokumen dengan simbol μ sebagai *centroid*. Dimana pada masing-masing iterasi 2 dokumen (d) dengan jarak *centroid* terdekat digabungkan. Metode ini dikenal juga dengan *centroid linkage hierarchical method* (CLHM).



Gambar 2.8 *Centroid Clustering* (Christopher D. Manning dkk, 2009)

2.1.4..3 *Divisive Clustering*

Sejauh ini yang banyak digunakan pada *hierarchical clustering* hanya *hierarchical aglomerative clustering*. Akan tetapi *hierarchical clustering* yang juga dapat digunakan adalah *divisive clustering* yang memiliki konsep *top-down clustering*, dimana proses *clustering* dimulai dengan mengganggap semua dokumen yang ada pada suatu koleksi dokumen sebagai 1 kluster. Kemudian kluster tersebut dipecah dengan menggunakan konsep *partitioning clustering*. Langkah tersebut terus dilakukan secara rekursif hingga masing-masing dokumen telah berada pada klusternya masing-masing (Christopher D. Manning dkk, 2009).

Secara konsep, *divisive clustering* lebih kompleks dibanding *hierarchical aglomerative clustering*, namun metode ini membutuhkan waktu komputasi yang lebih lama karena menerapkan konsep mempartisi kluster sebagai *subroutine*. Metode *clustering* yang termasuk ke dalam kelompok ini adalah *Bisecting K-Means* yang menyerap konsep *divisive klustering* secara keseluruhan. Langkah-langkah *clustering* dokumen dengan menggunakan *bisecting k-means* adalah sebagai berikut (Steinbach dkk, 2009) :

1. Pilih sebuah kluster yang ingin dipartisi.
2. Temukan 2 sub kluster dengan menggunakan konsep dasar *k-means* (tahap *bisecting*).
3. Ulangi langkah 2, dan ambil kluster dengan tingkat similaritas tertinggi.
4. Ulangi langkah 1, 2, dan 3 sampai jumlah kluster yang diinginkan tercapai.

2.1.4.4 *K-Means*

Metode *K-Means* adalah metode yang termasuk ke dalam kelompok *partitioning clustering* karena mengawali prosesnya dengan mempartisi suatu koleksi dokumen menjadi sebanyak k kluster, lalu dengan menggunakan fungsi similaritas, proses penggabungan antara *centroid* dan dokumen terdekat dilakukan. Metode yang diperkenalkan pertama kali oleh Mc Queen (1967) ini adalah metode yang paling umum digunakan dalam *clustering*/pengelompokkan karena mudah diimplementasikan dan kompleksitas waktunya linear. Adapun langkah *clustering*-nya adalah :

1. Tentukan berapa banyak jumlah kluster (K), misal K=3.

2. Tentukan titik pusat kluster (*Centroid*) secara acak.
3. Hitung jarak masing-masing dokumen yang ada dalam koleksi dengan *Centroid* menggunakan rumus korelasi antar 2 objek. Dalam penelitian ini penulis menggunakan rumus *cosine similarity*.
4. Hitung ulang *Centroid* berdasarkan dokumen yang ada pada kluster yang telah terbentuk dengan cara menghitung rata-rata dimensi dokumen yang ada pada kluster yang telah terbentuk. Berikut persamaan yang digunakan untuk membentuk *centroid* baru :

   Centroid Baru = $\frac{\sum_{i=1}^{t} d_{ij}+d_{ik}}{N}$ (2.7)

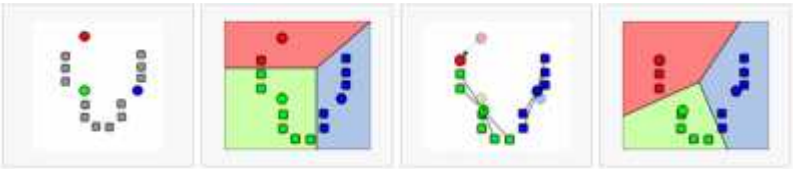
   Keterangan :

   d_{ij} : Dimensi ke-i dokumen j

   d_{ij} : Dimensi ke-i dokumen k

   N : Jumlah dokumen pada kluster t
5. Jika *Centroid* berubah maka ulangi proses 3 sampai 4, jika tidak maka proses pengelompokkan dokumen selesai.

Gambar 2.9 menunjukkan proses *clustering* dokumen dengan menggunakan metode *k-means* :



Gambar 2.9 *Clustering* Dokumen Dengan *K-Means*
(http://en.wikipedia.org/wiki/K-means_clustering)

2.1.5 *Evaluation*

Tahap ini adalah tahap dimana dilakukan proses evaluasi terhadap dokumen yang dihasilkan pada masing-masing kluster dari proses *clustering*. Evaluasi dilakukan untuk mengecek kembali apakah dokumen yang dihasilkan pada masing-masing kluster benar-benar merupakan dokumen yang memiliki kesamaan yang tinggi. Untuk mengukur nilai keberhasilan suatu hasil *clustering*/pengelompokkan, digunakan persamaan sebagai berikut :

$$Precision = \frac{\sum d}{N} \times 100\ \% \qquad (2.8)$$

Keterangan :

d : Dokumen yang berada pada kluster/kelompok yang tepat

N : Jumlah dokumen yang dikluster/kelompokan

Semakin besar nilai *precision*, maka semakin bagus kluster/kelompok yang dihasilkan.