BAB II

TINJAUAN PUSTAKA

A. Tinjauan Pustaka

Penelitian-penelitian yang pernah dilakukan di bidang *information retrieval* telah memunculkan berbagai metode pembobotan dan *clustering* untuk mengelompokkan dokumen teks ke dalam beberapa *cluster*. Salah satu metode *clustering* yang simpel, efisien, dan cepat adalah metode *k-Means* (Arthur, 2006). Sedangkan metode pembobotan yang paling populer dan hasilnya masih dianggap yang terbaik sampai saat ini adalah pembobotan TF-IDF (Elkan, 2005), meski ada pembobotan lain seperti *BM25 weighting*.

Metode *clustering k-Means* sendiri merupakan metode yang biasa digunakan untuk mengelompokkan data yang terstruktur seperti *database*. Dari data itu dibentuk beberapa dimensi pengukuran untuk penentuan *cluster* dari tiap data.

Setiap penelitian yang sudah dilakukan terhadap metode *clustering k-Means*, masalah yang belum dibahas adalah nilai *threshold* berapakah yang optimal dalam menggunakan metode *k-Means* ini. Nilai *threshold* adalah nilai batasan sampai kapan iterasi (perulangan) dari *k-Means* ini diteruskan.

Sampai sekarang belum ditemukan pula adanya implementasi pembobotan TF-IDF ke dalam metode *k-Means*. Penelitian kali ini dilakukan dengan menggabungkan salah satu metode *ad-hoc retrieval* paling populer yaitu VSM (*Vector Space Model*) ke dalam *k-Means*. Metode VSM, terutama bagian pembobotan TF-IDF-nya sering digunakan menjadi dasar pengembangan metode-

metode *ad-hoc retrieval* yang baru (Abbasi, 2009; Abual-Rub, 2007; Le Wang, 2007).

Hasil kajian pustaka terhadap metode *k-Means clustering* dan pembobotan TF-IDF dirangkum dalam tabel 1:

Tabel 1. Penelitian terdahulu mengenai *k-Means* dan TF-IDF

Peneliti	Penelitian yang dilakukan
Torsten Schielder dan Holger Meuss (2002)	Mengembangkan VSM untuk dapat melakukan proses query dan perangkingan pada dokumen XML.
D. T. Pham (2004)	Menggunakan two-phase *k-Means* untuk penerapan *k-Means* pada data yang besar atau banyak.
Sanjiv K. Bhatia (2004)	Menggunakan *Adaptive k-Means clustering* untuk menentukan nilai k dari *k-Means*.
D. T. Pham (2005)	Penentuan nilai k pada *k-Means* menggunakan beberapa pendekatan
David Arthur dan Sergei Vassilvitskii (2006)	Meneliti waktu yang diperlukan oleh *k-Means* untuk melakukan proses *clustering*.
Thaddeus Tarpey (2007)	Menghitung nilai k dari *k-Means* secara otomatis dengan metode *parametric k-Means*.
Yacine Rezgui (2007)	Menggunakan metode TF-IDF untuk mengembangkan konstruksi domain ontologi
Shai Ben-David (2009)	Menggunakan *risk-minimizing clustering* algorithm untuk menstabilkan proses *clustering* dari *k-Means*.

Tabel 1 lanjutan

Peneliti	Penelitian yang dilakukan
Kosuke Takano (2009)	Merekonstruksi Metode *Vector Space Model* untuk personalisasi hasil pencarian yang membutuhkan *feedback* dan analisa dari *user*.
Ashish Kathuria (2010)	Menggunakan *k-Means* untuk melalukan klasifikasi keinginan *user* search engine apakah dia mencari informasi tertentu atau mencari website tertentu.
Christophe Moulin (2010)	Penggunaan TF-IDF untuk klasifikasi *file image*
Juan Ramos (2010)	Menggunakan TF-IDF untuk mengembalikan dokumen yang relevan terhadap *query* dari *user*.
Mark Ming (2010)	Mengembangkan metode *k-Means* menjadi *ik-Means* (*intelligent k-Means*) untuk menentukan nilai k secara otomatis.
Yihen Chen (2010)	Membandingkan metode SOM dan *k-Means* dalam proses *clustering* dimana metode *k-Means* lebih simpel dan cepat walau SOM memiliki per*forma* lebih baik.

B. Landasan Teori

1. Information Retrieval

Information Retrieval (IR) adalah ilmu pencarian informasi dari sejumlah data yang sudah hilang karena terlalu banyaknya data yang ada. Ilmu ini dipopulerkan oleh Vannevar Bush (1945) dan implementasinya mulai dikenalkan pada tahun 1950-an. Pada tahun 1990-an, sudah banyak

teknik dan metode dari in*form*atioan retrieval yang dikembangkan dan dipakai.

Metode penggunaan IR dibagi menjadi 4 macam (Manning, 2008) yaitu: *ad-hoc retrieval*, *clustering/classification*, *topic detection & tracking*, dan *filtering*. Masing-masing memiliki model yang beranekaragam dan dipakai sesuai dengan kasus yang sesuai.

Dalam metode yang manapun, selalu dilakukan proses pembangunan indeks (*index construction*) dari dokumen teks supaya mempermudah perhitungan kemiripan dan relevansi. Proses pembangunan indeks tersebut meliputi proses *stemming*, *stopword removal*, dan tokenisasi. Setelah indeks berhasil dibangun, barulah model-model perhitungan IR dapat diimplementasikan.

2. Clustering

Sebagai salah satu metode IR, *clustering* melakukan pengelompokan data berdasarkan *cluster*/kelas dan merupakan teknik untuk mengorganisasikan data yang tidak terstruktur tersebut menjadi suatu struktur data yang mempunyai nilai informasi tertentu. Tujuan dari algoritma ini adalah untuk mengelompokkan dokumen yang memiliki kesamaan (*similarity*) ke dalam satu *cluster* tertentu. Dengan kata lain, dokumen dalam satu *cluster* akan memiliki relevansi satu sama lain, tetapi tidak memiliki relevansi dengan dokumen di dalam *cluster* lain. Dalam proses yang baik, data yang dihasilkan dalam satu *cluster* akan memiliki

tingkat kesamaan yang tinggi, dan akan memiliki nilai kesamaan yang rendah dengan *cluster* yang lainnya

clustering melakukan pengelompokan data tanpa berdasar pada kelas data tertentu yang sudah ditetapkan dari awal. Proses ini sangat berbeda dengan proses pada *classification* yang pada awal proses harus memberikan kelas-kelas data. Sehingga *clustering* sering disebut dengan pengelompokan data yang tidak terstruktur.

Dalam algoritma *clustering*, biasanya diperlukan fungsi jarak yang digunakan untuk mengukur jarak (kemiripan suatu data dengan data yang lain). Beberapa fungsi jarak yang dapat digunakan dalam proses *clustering* adalah :

- *Euclidean Distance*, merupakan perhitungan jarak dengan menggunakan 2 keadaan.
- *Manhattan Distance*, disebut juga dengan taxicap-norm, yang menghitung jarak dengan menggunakan 1 keadaan.
- *Mahalanobis Distance*, merupakan fungsi jarak yang digunakan untuk menghitung data dalam skala dan korelasi yang berbeda.

3. Pembobotan TF-IDF

Metode TF-IDF (Robertson, 2004) merupakan metode untuk menghitung bobot setiap kata yang paling umum digunakan pada *information retrieval*. Metode ini juga terkenal efisien, simpel dan memiliki hasil yang akurat (Ramos, 2010). Metode ini akan menghitung

nilai *Term Frequency* (TF) dan *Inverse Document Frequency* (IDF) pada setiap token (kata) di setiap dokumen dalam korpus. Metode ini akan menghitung bobot setiap token t di dokumen d dengan rumus:

$$w_{(t,d)} = TF_{(t,d)} * IDF_{(t)} \quad \text{.....} \quad (1)$$

Dimana nilai IDF(t) didapatkan dari :

$$IDF_{(t)} = \log(|D|/DF_{(t)}) \quad \text{.....} \quad (2)$$

Keterangan :

$TF_{(t,d)}$: Jumlah kemunculan token t pada dokumen d

$IDF_{(t)}$: Nilai IDF token t

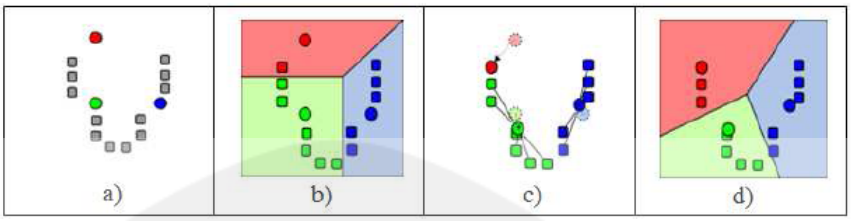
$DF_{(t)}$: jumlah dokumen yang memuat token t

$|D|$: jumlah dokumen dalam korpus

4. k-Means clustering

Metode *k-Means clustering* merupakan metode *clustering* yang dikenalkan oleh Lloyd (1982). Metode *k-Means* adalah metode yang terkenal cepat dan simpel (Arthur, 2006). *k-Means* adalah metode *clustering* yang mengelompokkan semua data yang dimiliki ke dalam k *cluster*, dimana nilai k sudah ditentukan sebelumnya.

k-Means mengelompokkan data berdasarkan jarak dari tiap dokumen ke pusat *cluster* (*centroid*) yang sudah ditentukan sebanyak k, dan mengelompokkan data-data ke pusat *cluster* yang terdekat.



Gambar 1. Ilustrasi proses *k-Means clustering*.

(Sumber: wikipedia.org)

Algoritma dari metode *k-Means* itu sendiri adalah sebagai berikut:

a. Pilih secara acak vektor dokumen yang akan digunakan sebagai *centroid* awal sebanyak k.
b. Cari *centroid* yang paling dekat dari setiap dokumen.
c. Hitung ulang untuk menentukan *centroid* baru dari setiap *cluster*.
d. Lakukan langkah 2 dan 3 hingga *centroid* tidak mengalami perubahan lagi.

Rumus perhitungan *centroid* baru dari setiap *cluster* dicari dengan menggunakan rumus :

$$M_k = \left(\frac{1}{n_k}\right) \sum_{i=1}^{n_k} x_{ik} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (3)$$

Dengan :

M_k : Nilai *centroid* dari suatu *cluster*

n_k : jumlah dokumen yang berada dalam satu *cluster*

X_{ik} : nilai X dari sampel dokumen ke-i yang termasuk *cluster* k(Ck)

Sedangkan untuk menemukan jarak dua dokumen digunakan rumus *euclidean distance* :

$$d_{(i,j)} = \sqrt{(|x_{i1} - x_{i1}|^2 + |x_{i2} - x_{i2}|^2 + .. + |x_{ip} - x_{ip}|^2)} \quad \ldots\ldots \quad (4)$$

Dengan :

$d_{(i,j)}$: jarak dokumen ke-i ke dokumen ke-j

$x_{i(n)}$: kata ke n di dokumen ke-i.

$x_{j(n)}$: kata ke n di dokumen ke-j.

Untuk mengaplikasikan *k-Means* dalam *clustering* dokumen teks, maka dibentuklah vektor dokumen dengan jumlah dimensi sebanyak token unik dalam korpus.

Berikut ini diberikan contoh penerapan *k-Means clustering* yang sudah dijelaskan di atas. Sebagai sampel, terdapat 4 buah dokumen sebagai berikut yang akan dikelompokkan ke dalam 2 *cluster*:

Dokumen 1 (D1) : “Shipment of gold damage in a fire”

Dokumen 2 (D2) : “Delivery of silver arrived in a silver truck”

Dokumen 3 (D3) : “Shipment of gold arrived in a truck”

Dokumen 4 (D4) : “Silver truck arrived in the silver city”

Nilai *k* adalah 2 dan *centroid* awal yang terpilih secara random adalah dokumen 1 dan 2. Nilai *threshold* adalah 20%.

Berdasarkan data sampel di atas, maka dilakukan proses pembangunan indeks untuk membentuk *lexicon* (kamus token) dan pembobotan dengan TF-IDF hingga didapat nilai w (bobot) sesuai dengan rumus yang tersedia.

Berikut ini diberikan *lexicon* yang terbentuk dan contoh perhitungan untuk token "shipment". Token "shipment" dimiliki oleh Dokumen 1 sebanyak 1, dan dokumen 3 sebanyak 1. Jadi df untuk token "shipment" adalah 2. Berikut ini perhitungan IDF untuk token "shipment"

$$\text{IDF}_{(shipment)} = \log\left(\frac{\text{jumlah dokumen}}{\text{jumlah dokumen yang memuat shipment}}\right)$$

$$\text{IDF}_{(shipment)} = \log\left(\frac{4}{2}\right)$$

$$\text{IDF}_{(shipment)} = 0.301$$

Berikut ini adalah perhitungan TF-IDF untuk w dari token "shipment" menurut rumus (1):

$$w_{(shipment)} = \text{TF}_{(\text{shipment})} * IDF(\text{shipment})$$

$$w_{(shipment)} = 1 * 0.301$$

$$w_{(shipment)} = 0.301$$

Langkah diatas dilakukan kepada setiap token yang dihasilkan. Maka akan dihasilkan data seperti tabel di bawah ini:

Tabel 2. Perhitungan bobot tiap token dalam *lexicon*

	D(1)		D(2)		D(3)		D(4)			
	TF	**w**	**TF**	**w**	**TF**	**w**	**TF**	**w**	**IDF**	**DF**
shipment	1	0.301	0	0	1	0.301	0	0	0.301	2
of	1	0.125	1	0.125	1	0.125	0	0	0.125	3
gold	1	0.301	0	0	1	0.301	0	0	0.301	2
damage	1	0.602	0	0	0	0	0	0	0.602	1

Tabel 2 Lanjutan

	D(1)		D(2)		D(3)		D(4)			
	TF	**w**	**TF**	**w**	**TF**	**w**	**TF**	**w**	**IDF**	**DF**
in	1	0	1	0	1	0	1	0	0	4
a	1	0.125	1	0.125	1	0.125	0	0	0.125	3
fire	1	0.602	0	0	0	0	0	0	0.602	1
delivery	0	0	1	0.602	0	0	0	0	0.602	1
silver	0	0	2	0.602	0	0	2	0.602	0.301	2
arrived	0	0	1	0.125	1	0.125	1	0.125	0.125	3
truck	0	0	1	0.125	1	0.125	1	0.125	0.125	3
the	0	0	0	0	0	0	1	0.602	0.602	1
city	0	0	0	0	0	0	1	0.602	0.602	1

Keempat dokumen tersebut akan dibagi menjadi 2 *cluster*, sehingga perlu ditentukan 2 *centroid*/pusat *cluster* awal. Karena *centroid*/pusat *cluster* diambil secara random, *centroid* pertama (C1) akan diinisialisasi berada pada koordinat yang sama dengan dokumen 1 dan *centroid* kedua (C2) berada pada koordinat yang sama dengan dokumen 2.

Setelah itu, kita lakukan iterasi dalam pembentukan *cluster* dari 4 dokumen yang tersedia dalam korpus.

Iterasi Pertama:

Dengan data pembobotan tersebut, maka dapat dihitung jarak antara masing-masing dokumen dengan masing-masing *centroid* menggunakan rumus *euclidean distance*. Berikut ini adalah proses perhitungan jarak dengan menggunakan *euclidean distance*:

$$d_{(D1,Cent1)} = \sqrt{(|x_{i1} - x_{i1}|^2 + |x_{i2} - x_{i2}|^2 + .. + |x_{ip} - x_{ip}|^2)}$$

$$d_{(D1,Cent1)} = \sqrt{(|0.301 - 0.301|^2 + |0.301 - 0.301|^2 + .. + |0 - 0|^2)}$$

$$d_{(D1,Cent1)} = 0$$

$$d_{(D1,Cent2)} = \sqrt{(|x_{i1} - x_{i1}|^2 + |x_{i2} - x_{i2}|^2 + .. + |x_{ip} - x_{ip}|^2)}$$

$$d_{(D1,Cent2)} = \sqrt{(|0.301 - 0|^2 + |0.125 - 0.125|^2 + .. + |0 - 0|^2)}$$

$$d_{(D1,Cent2)} = 1.2892$$

Proses perhitungan jarak dilakukan pada setiap dokumen terhadap setiap *centroid* yang ada. Berikut ini adalah hasil perhitungannya:

Tabel 3. perhitungan jarak dokumen dengan *centroid*

Dokumen	***Centroid***	**Jarak**
D(1)	***Centroid* 1**	**0**
D(1)	*Centroid* 2	1.2892
D(2)	*Centroid* 1	1.2892
D(2)	***Centroid* 2**	**0**
D(3)	***Centroid* 1**	**0.86958**
D(3)	*Centroid* 2	0.9519
D(4)	*Centroid* 1	1.4338
D(4)	***Centroid* 2**	**1.0576**

Berdasarkan perhitungan di atas, maka pada iterasi pertama masing-masing dokumen akan masuk ke salah satu *cluster* yang terdekat, berikut ini hasil *cluster* pada iterasi pertama:

Tabel 4. Pembagian dokumen ke dalam *cluster*

Dokumen	Jarak	Masuk ke *cluster*
D(1)	0	1
D(2)	0	2
D(3)	0.86958	1
D(4)	1. 0576	2

Kemudian dilakukan perhitungan untuk menentukan posisi *centroid* yang baru. Dimana dihitung melalui rata-rata dari masing-masing vektor yang merupakan anggota dari *cluster*. Didapatkan :

Tabel 5. perhitungan *centroid* C1 yang baru

		D1	D3	C1 (rata-rata)
1	a	0.125	0.125	0.125
2	arrived	0	0.125	0.0625
3	city	0	0	0
4	damage	0.602	0	0.301
5	delivery	0	0	0
6	fire	0.602	0	0.301
7	gold	0.301	0.301	0.301
8	in	0	0	0
9	of	0.125	0.125	0.125
10	shipment	0.301	0.301	0.301
11	silver	0	0	0
12	the	0	0	0
13	truck	0	0.125	0.0625

Tabel 6. Perhitungan *centroid* C2 yang baru

		D2	D4	C2 (rata-rata)
1	a	0.125	0	0.0625
2	arrived	0.12494	0.12494	0.12494

3	city	0	0.60206	0.30103
4	damage	0	0	0
5	delivery	0.60206	0	0.30103
6	fire	0	0	0
7	gold	0	0	0
8	in	0	0	0
9	of	0.125	0	0.0625
10	shipment	0	0	0
11	silver	0.60206	0.60206	0.60206
12	the	0	0.60206	0.30103
13	truck	0.12494	0.12494	0.12494

Setelah mendapatkan vector posisi untuk *centroid* 1 dan 2, dihitung jarak perpindahan dari *centroid* lama ke baru dengan menggunakan rumus *euclidian*. Hal ini yang akan menentukan apakah iterasi akan dilanjutkan atau tidak.

$$d(Cent1L, Cent1B) = \sqrt{(|x_{i1} - x_{i1}|^2 + |x_{i2} - x_{i2}|^2 + .. + |x_{ip} - x_{ip}|^2)}$$

$$d(Cent1L, Cent1B) = \sqrt{(|0.301 - 0.301|^2 + |0.301 - 0.301|^2 + .. + |0 - 0|^2)}$$

$$d(Cent1L, Cent1B) = 0$$

$$d(Cent1L, Cent1B) = \sqrt{(|x_{i1} - x_{i1}|^2 + |x_{i2} - x_{i2}|^2 + .. + |x_{ip} - x_{ip}|^2)}$$

$$d(Cent1L, Cent1B) = \sqrt{(|0.301 - 0|^2 + |0.301 - 0.301|^2 + .. + |0 - 0|^2)}$$

$$d(Cent1L, Cent1B) = 0.4348$$

Dengan melakukan proses diatas, maka dihasilkan:

Tabel 7. Jarak *centroid* lama dan *centroid* baru

		Jarak
***Centroid*1 Lama**	***Centroid*1 Baru**	0.43479
***Centroid*2 Lama**	***Centroid*2 Baru**	0.52886

Kemudian dilihat, apakah jarak antara *centroid* lama dengan *centroid* baru lebih dari *threshold* dikali dengan magnitude *centroid* lama.

$$\text{jarak C1Lama C1Baru} > \frac{20}{100} * |\text{C1}|$$

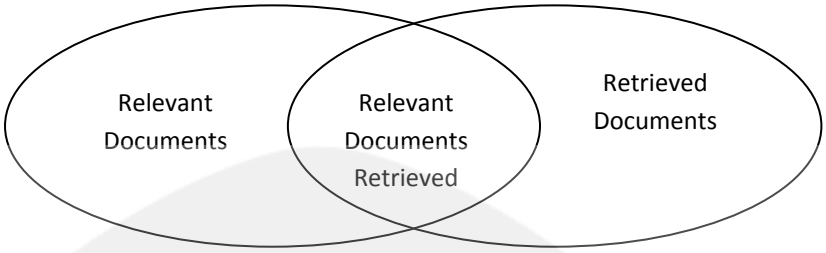
$$0.43479 > \frac{20}{100} * 0.9682$$

$$0.43479 > 0.1936$$

Karena bernilai benar, maka iterasi akan terus dilanjutkan, dimulai lagi dari menghitung jarak masing-masing dokumen dengan kedua *centroid cluster* yang baru. Proses *clustering* akan berhenti sampai jarak *centroid* lama dan *centroid* baru di bawah nilai *threshold* yang ditetapkan.

5. Precision dan recall

Pada ilmu IR, keakuratan menjadi hal yang penting karena *user* mengharapkan informasi yang didapat sesuai dengan yang diinginkan. Demikian pula dengan metode *clustering*, diperlukan evaluasi keakuratan model *clustering*, salah satunya menggunakan parameter *precision* dan *recall*. Nilai *precision* dan *recall* ini sudah dianggap menjadi salah satu parameter pengukuran keakuratan metode *information retrieval* yang tepat dibanding dengan metode pengukuran yang lain. (Su, 1994). Pengukuran akurasi sistem *information retrieval* menggunakan kedua parameter ini juga masih dipakai untuk pengujian *bibliographic database* seperti *MEDLINE*, *Google Scholar*, dan *AgeLine* (Walters, 2009).



$$precision = \frac{Relevant\ Documents\ Retrieved}{Retrieved\ Document}$$

$$recall = \frac{Relevant\ Documents\ Retrieved}{Relevant\ Document}$$

Gambar 2. Precision dan *Recall*

Precision merupakan salah satu parameter pengukuran hasil *retrieval* terhadap dokumen. Dengan kata lain, *precision* dapat diartikan sebagai kecocokan antara permintaaan informasi dengan respon dari permintaan tersebut. *Precision* dapat dihitung dengan:

$$Precision = \frac{jumlah\ dokumen\ relevan\ yang\ terretrieve}{jumlah\ seluruh\ dokumen} \quad \ldots\ldots \quad (5)$$

Sedangkan *recall* merupakan parameter yang didapat dari jumlah dokumen terambil yang relevan dibagi dengan keseluruhan jumlah dokumen yang relevan. *Recall* digunakan untuk melakukan pengukuran terhadap tingkat keberhasilan sistem dalam mengenali suatu dokumen yang relevan terhadap kueri.

$$recall = \frac{jumlah\ dokumen\ relevan\ yang\ terretrieve}{jumlah\ seluruh\ dokumen\ relevan\ dalam\ koleksi} \quad \ldots\ldots\ldots \quad (6)$$

Berikut ini adalah contoh perhitungan *precision* dan *recall*. Terdapat 10 dokumen yang diberi nama D1, D2, D3, D4, D5, D6, D7, D8, D9, dan

D10. Setelah dilakukan proses *retrieval*, keluaran sistem adaah D2, D4, D5, dan D9. Menurut pengguna hanya terdapat 3 dokumen yang relevan, yaitu D2, D4, dan D9. Sedangkan pada koreksi dokumen seharusnya terdapat 5 dokumen yang relevan. Sehingga diperoleh:

Jumlah dokumen relevan terambil = 3

Jumlah seluruh dokumen terambil = 4

Jumlah seluruh dokumen relevan dalam koreksi = 5

Hasil Perhitungan *precision* dan *recall* :

Precision = 3/4 = 75%

Recall = 3/5 = 60%

Dalam evaluasi sistem *clustering*, *precision* dan *recall* dihitung dengan membandingkan dokumen yang dikelompokkan secara manual dengan yang dilakukan oleh sistem.

C. Hipotesis

Penelitian ini diprediksi akan memberikan hasil sebagai berikut:

1. integrasi metode pembobotan TF-IDF ke dalam *clustering k-Means* dapat digunakan untuk melakukan pengelompokan dokumen teks berdasarkan kemiripan isi dokumen.
2. Akan didapatkan nilai *threshold* yang optimal untuk proses *clustering* secara umum.