BAB 2
LANDASAN TEORI

2.1 Data Mining

2.1.1 Definisi Data Mining

Menurut (Hand, Heikki dan Padhraic 2001) *data mining* merupakan proses analisis terhadap sekumpulan data yang berskala besar untuk menemukan keterkaitan yang tidak terduga kemudian menyimpulkannya, sehingga muncul nilai kebaruan informasi yang berguna bagi pemiliknya. Pada pengertian lain disebutkan *data mining* merupakan bidang kajian antar berbagai disiplin ilmu yang meliputi *machine learning*, *pattern recognition*, statistik, basis data, dan visualisasi yang bertujuan untuk mengekstraksi informasi baru dari basis data dalam skala besar (Cabena, et al. 1998).

Dari dua pengertian di atas dapat disimpulkan bahwa *data mining* merupakan sebuah proses untuk menggali informasi baru yang bermakna dari sekumpulan *dataset* dalam skala besar. Informasi yang bermakna bisa berupa keterkaitan antar data, pola serta tren yang dimiliki oleh kumpulan data tersebut. Teknik yang digunakan pada *data mining* meliputi : teknologi pengenalan pola, statistika, matematika serta visualisasi.

Berbagai aspek dalam kehidupan modern telah menerapkan mekanisme penyimpanan data ke dalam basis data. Setiap harinya data yang terkumpul akan berkembang. Perkembangan ini berpotensi terjadinya peledakan jumlah data. Data dalam skala besar tersebut tidak seharusnya disimpan begitu saja, kumpulan data ini dapat dipelajari lebih lanjut, sehingga akan muncul pengetahuan-pengetahuan baru yang berguna bagi pemiliknya. Pengetahuan baru yang muncul dapat berupa prediksi, korelasi, Kluster serta bentuk pengetahuan lainnya yang bisa dimanfaatkan untuk kemajuan organisasi pemiliknya.

2.1.2 Teknik Data Mining

Berdasarkan definisi dari *data mining*, terdapat gabungan beberapa disiplin ilmu yang digunakan untuk menggali informasi baru dari sekumpulan *dataset* antara lain :

a. Statistik.

   Merupakan cikal bakal lahirnya *data mining*. Bisa dikatakan tanpa adanya bidang ilmu statistik, mungkin *data mining* tidak akan pernah ada. Statistik merupakan teknik yang digunakan untuk menemukan hubungan yang sistematis antar beberapa variabel dalam suatu *dataset*. Teknik ini sering pula disebut dengan *Exploratory Data Analysis* (*EDA*). Secara umum teknik *EDA* yang sering digunakan pada proses *Data mining* meliputi :

   - Metode Komputasi : statistik deskriptif, korelasi, analisis multivariat, , pemodelan linier/non-linier, dan lain sebagainya.
   - Visualisasi Data : teknik sering digunakan untuk menyajikan informasi dalam bentuk visual. Beberapa teknik visualisasi data antara lain : histogram, *box plot, scatter, plot, contour plot, matriks plot, icon plot* dan lain sebagainya.

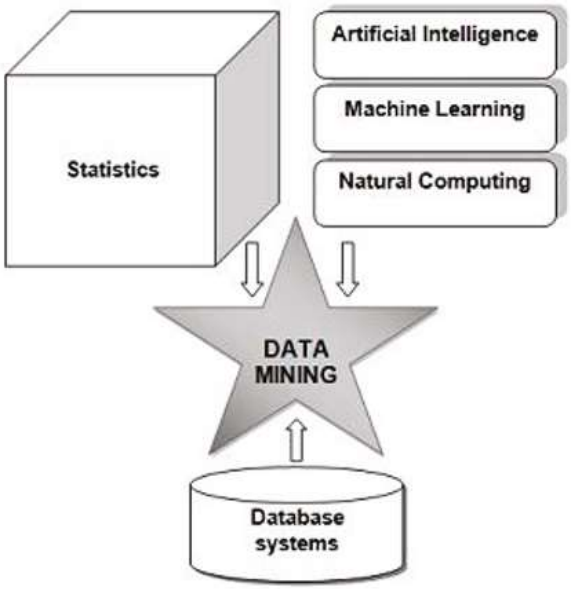
b. Kecerdasan Buatan (*Artificial Intelligence*).

   Tidak seperti statistik, *Artificial Intelligence* (*AI*) didasarkan pada data yang bersifat heuristis. Oleh sebab itulah *AI* berkontribusi dalam teknik pemrosesan informasi yang didasarkan pada model penalaran manusia. Teknik lain yang masih berhubungan dengan *AI* adalah *machine learning* yang memungkinkan komputer untuk melakukan pembelajaran (*Training*) terhadap suatu *dataset*.

c. Sistem Basis data.

   Basis data dipandang sebagai dasar ketiga dari *data mining* yang menyediakan kumpulan data yang nantinya akan diproses dengan menggunakan dua teknik di atas. Basis data merupakan repositori atau gudang data yang akan diproses dalam *data mining*.

Keterkaitan antar teknik yang digunakan dalam proses data mining dapat digambarkan pada gambar berikut :



Gambar 2.1 : Keterkaitan teknik-teknik dalam data mining

Sumber : Gorunescu (2011)

2.1.3 Permasalahan yang Diselesaikan dengan *Data Mining*

Beberapa permasalahan yang dapat diselesaikan dengan *data mining* antara lain :

- Deskripsi, jenis permasalahan dalam data mining yang bertujuan untuk mencari pola dan tren yang terkandung dalam suatu *dataset*. Permasalahan ini biasanya diselesaikan dengan menggunakan teknik EDA dan visualisasi data (untuk membantu pembaca memahami pola dan tren *dataset*)
- Klasifikasi, merupakan upaya pencarian variabel target suatu rekaman, dimana variabel target yang diinginkan berupa variabel kategorik. Dalam permasalahan *data mining* melakukan pengujian terhadap *dataset* skala besar, masing-masing rekaman terdiri dari variabel target dan juga *predictor*. Setelah diketahui pola komposisi dari *predictor*, selanjutnya akan dilakukan pencarian variabel target dari rekaman baru.
- Estimasi, merupakan permasalahan yang hampir mirip dengan klasifikasi, namun memiliki sedikit perbedaan yakni variabel target yang dicari berupa variabel numerik. Model *dataset* yang dibutuhkan tersusun dari rekaman data yang lengkap, dimana di dalamnya sudah terdapat variabel target dan

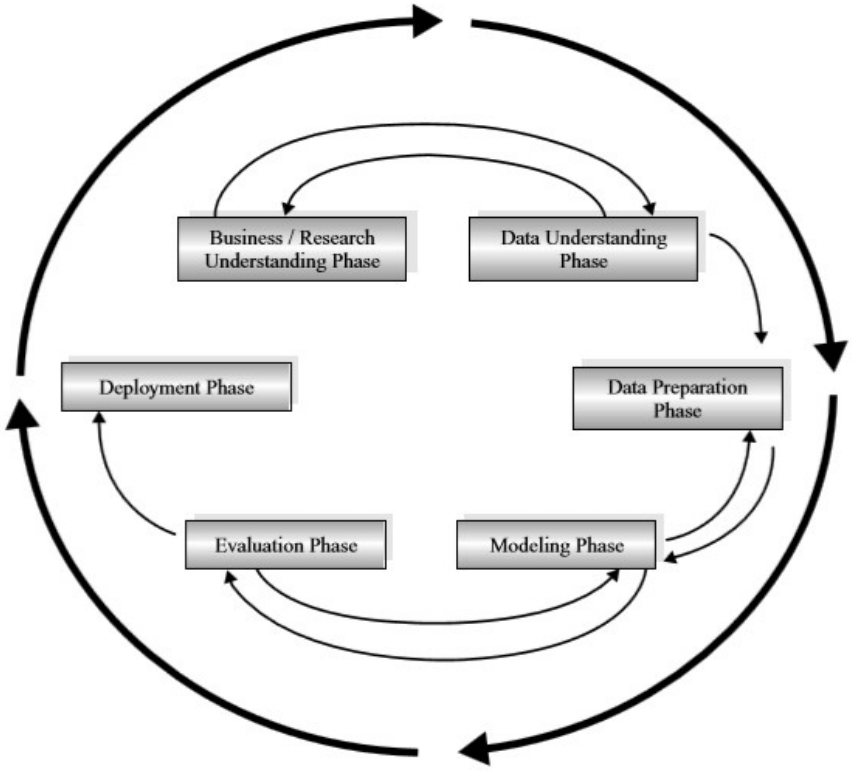
juga *predictor* (atribut). Selanjutnya untuk data baru, akan dilakukan estimasi variabel target berdasarkan nilai *predictor*.

- Prediksi, permasalahan yang sama dengan klasifikasi dan estimasi, namun hasil dari prediksi digunakan dimasa mendatang (untuk keperluan peramalan).
- Kluster, merupakan upaya untuk mengelompokkan rekaman, observasi atau kasus ke dalam kelompok-kelompok yang sejenis. Rekaman memiliki kesamaan karakteristik dengan rekaman lain pada Kluster yang sama, serta memiliki perbedaan dengan rekaman lain pada Kluster yang berbeda. *Clustering* berbeda dengan klasifikasi, dimana di dalamnya tidak terdapat variabel target. Clustering berupaya untuk menyegmentasi rekaman-rekaman ke dalam beberapa kelompok yang bersifat homogen.
- Asosiasi, merupakan permasalahan pencarian atribut-atribut yang saling berhubungan dalam suatu *dataset* dengan aturan "jika anteseden, maka konsekuen", dimana aturan tersebut dilengkapi dengan ukuran *support* dan *confidence*.

2.1.4 Daur Hidup *Data Mining*

Menurut CRISP-DM (*Cross-Industry Standard Process for Data Mining*) terdapat 6 fase daur hidup proyek *data mining* dimana masing-masing tahapan dalam fase tersebut bersifat adaptif. Artinya terdapat beberapa fase yang bergantung pada fase sebelumnya (tahapan bisa kembali lagi ke fase sebelumnya jika ditemukan ketidak sesuaian atau diperlukan perbaikan). Sebagai contoh ketika berada dalam fase pemodelan, tahapan ini merupakan salah tahapan yang didasarkan pada perilaku dan karakteristik dari model, kita bisa kembali lagi ke tahapan persiapan data untuk perbaikan dalam pemodelan data. Jika tahap pemodelan sudah mencapai titik yang optimal baru bisa dilanjutkan ke tahapan evaluasi.

Perilaku iteratif dari CRISP dapat ditunjukkan pada lingkaran luar pada gambar 2.2. Gambaran secara jelas daur hidup *data mining* dapat ditunjukkan pada gambar berikut :



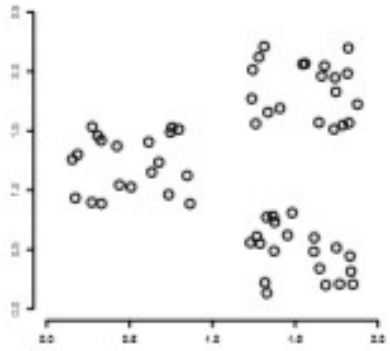
Gambar 2.2 : Daur hidup data mining,

sumber : Larose, (2005)

2.2 *Clustering*

Sebagaimana yang sudah dijelaskan pada pembahasan sebelumnya, bahwa *clustering* merupakan bentuk *data mining* untuk mengelompokkan *dataset* ke dalam kelompok-kelompok yang homogen. Homogen di sini memiliki makna bahwa data-data dalam Kluster yang sama cenderung memiliki kemiripan satu dengan lainnya, sedangkan data-data pada Kluster yang berbeda memiliki karakteristik yang berbeda pula. *Clustering* merupakan salah satu bentuk pembelajaran *unsupervised*. Proses clustering bukanlah proses untuk klasifikasi, estimasi maupun prediksi variabel target, melainkan proses untuk memisah-misahkan dataset ke dalam kelompok-kelompok yang sejenis.

Kunci utama dalam sebuah proses *clustering* adalah ukuran jarak yang digunakan. Pada gambar 2.3 ukuran jarak yang digunakan adalah jarak dalam bidang 2 dimensi. Jelas terlihat berdasarkan jarak antar data, terdapat 3 Kluster (kelompok). Semakin dekat jarak antar data, maka semakin mirip karakteristik dari data tersebut.



Gambar 2.3 : Contoh dataset yang telah terbagi menjadi 3 Kluster**,**

sumber : Manning, et al., (2009)

Beberapa ukuran jarak yang bisa digunakan dalam proses clustering antara lain : *Euclidean Distance, Manhattan Distance, Bit-Vector Distance, Hamming Distance, Jaccard Index, Cosine Index* dan *Dice Index*. Dari beberapa ukuran jarak tersebut, menurut (Pandit dan Gupta 2011) yang sesuai untuk digunakan dalam proses *text clustering* adalah *cosine index*.

Berdasarkan dari hasil yang dikeluarkan (*outcome*), proses clustering dapat dibedakan menjadi 2, yaitu *flat clustering* dan *hierarchical clustering*. Flat clustering akan menghasilkan kumpulan Kluster yang secara eksplisit tidak memiliki keterkaitan antar Kluster satu dengan lainnya, sedangkan *hierarchical clustering* merupakan proses *kluster* dimana suatu Kluster bisa menjadi anggota dari Kluster lain di atasnya (Kluster yang lebih besar).

2.3 *Clustering* Dokumen

Kumpulan Dokumen dapat dipandang sebagai suatu *dataset* yang bisa kita Kluster-kan. Sesuai dengan prinsip dasar dari Kluster, bahwa untuk dapat mengkluster *dataset* terlebih dahulu kita harus bisa memahami ciri yang terkandung dalam data tersebut, begitu juga dengan *clustering* dokumen, agar kumpulan dokumen dapat di-*cluster* maka hal mendasar yang harus kita pahami adalah bagaimana mempelajari ciri dari sebuah dokumen (*Feature Extraction*). Ciri suatu dokumen dapat dilihat dari topik yang dibahas di dalamnya . Sehingga dapat dikatakan *clustering* dokumen merupakan sebuah upaya untuk mengelompokkan dokumen-dokumen sesuai dengan topiknya.

Clustering dokumen telah diaplikasikan dalam banyak hal. Beberapa contoh aplikasi *clustering* dokumen antara lain Kluster untuk hasil pencarian, dimana item-item hasil pencarian dari sebuah mesin pencari dipandang sebagai kumpulan *dataset.* Beberapa pengguna mesin pencari terkadang merasa lebih mudah ketika mereka cukup melihat daftar kumpulan dokumen yang mirip dibandingkan jika mereka harus melihat item-item hasil pencarian satu persatu. Contoh lain penerapan *clustering* dokumen adalah untuk pengklasifikasian artikel berdasarkan topiknya, sehingga pembaca mendapatkan kemudahan dalam melihat artikel-artikel dengan topik sejenis.

2.3.1 *Preprocessing*

Kumpulan artikel yang akan dikluster merupakan data skala besar, diambil berasal dari berbagai sumber yang sifatnya heterogen. Karena sifatnya ini, maka data skala besar cenderung tidak konsisten dan mengandung banyak *noise* (Kumar dan Chandrasekhar 2012). Jika data yang akan diproses tidak konsisten, maka kemungkinannya proses clustering akan mendapatkan hasil yang tidak akurat. Agar data yang diproses menjadi akurat, maka diperlukan tahapan *preprocessing*. Tujuan dari tahapan tersebut adalah meningkatkan kualitas data dengan cara mengurangi bagian-bagian yang tidak diperlukan dalam proses Kluster.

Berikut ini merupakan langkah-langkah dalam tahapan *pre-processing* :

1. *Tokenization*, tujuan dari tahap ini adalah untuk mempermudah eksplorasi kata pada artikel yang akan diproses dengan cara membuang tanda-tanda pungtuasi, seperti tanda kurung, petik dan lain sebagainya.
2. *Stopword Removal*, kata yang terlalu sering muncul pada seluruh artikel biasanya tidak terlalu banyak berperan dalam mengidentifikasi makna suatu artikel, sehingga tidak diperlukan dalam proses *clustering*. Contoh kata yang sering muncul pada seluruh artikel adalah kata depan, kata keterangan, kata bantu dan lain sebagainya. Kata-kata ini sebaiknya tidak dilibatkan dalam proses *clustering*.
3. *Stemming*, merupakan teknik untuk mengonversi kata-kata pada artikel menjadi kata dasar. Satu kata dasar biasanya digunakan pada beberapa kata dengan perbedaan imbuhan, dengan mengembalikan seluruh kata menjadi kata

dasar, hal ini dapat mereduksi jumlah kata yang harus diproses, sehingga proses Kluster akan berjalan lebih efisien.

2.3.2 Model Representasi Data

Setelah tahap *preprocessing*, akan didapat kumpulan kata dasar yang mewakili makna dari suatu artikel. Kumpulan data tekstual tersebut tidak dapat begitu saja dikluster. Agar kumpulan kata tersebut dapat dikluster, maka data perlu disajikan dalam format yang memungkinkan untuk dikluster. Salah satu contoh model representasi data yang memungkinkan untuk dikluster adalah model representasi *Vector Space Model (VSM)*.

Setiap elemen dalam *VSM* mengindikasikan seberapa banyak suatu kata muncul dalam dokumen. Contoh *VSM* yang paling sederhana adalah matriks kata dan dokumen, dimana baris matriks dianggap sebagai kata-kata yang muncul sementara kolom matriks sebagai dokumen. Contoh *VSM* sederhana dapat dilihat pada tabel berikut ini :

Tabel 2.1 : Contoh VSM sederhana yang menyimpan vektor frekuensi kata

	Term1	Kata2	Kata3	Kata4	KataN
Dok1	F1,1	F1,2	F1,3	F1,4	F1,N
Dok2	F2,1	F2,2	F2,3	F2,4	F2,N
DokN	FN,1	FN,2	FN,3	FN,4	FN,N

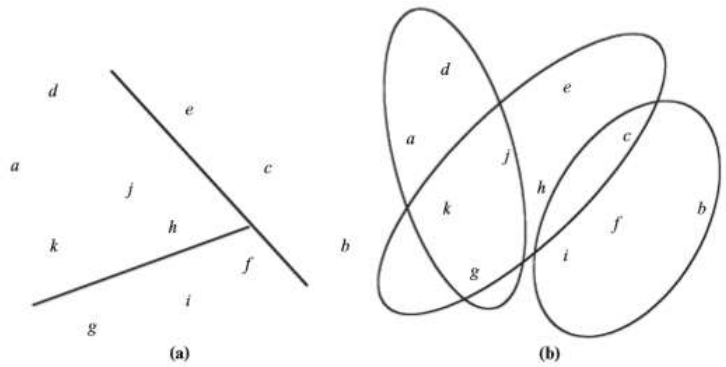
F1,1 merupakan nilai bilangan bulat yang menyatakan frekuensi kemunculan *Term1* pada Dok1. Dari contoh *VSM* di atas jelas terlihat bahwa sebuah dokumen dianggap sebagai vektor kumpulan kata. Selanjutnya vektor-vektor inilah yang akan diukur kemiripannya satu dengan lainnya. Vektor-vektor yang mirip akan masuk ke dalam Kluster yang sama, sementara vektor-vektor yang berbeda akan berada pada Kluster yang berbeda pula.

2.3.3 Algoritme *Kluster*

Luaran dari sebuah proses Kluster dapat disajikan dalam bentuk diagram yang menunjukkan seperti apa sebuah objek masuk ke dalam suatu Kluster. Artinya, algoritme Kluster adalah algoritme - algoritme yang mampu menyajikan model-model luaran tersebut. Terdapat beberapa jenis diagram yang dapat menggambarkan keanggotaan objek dalam suatu Kluster.

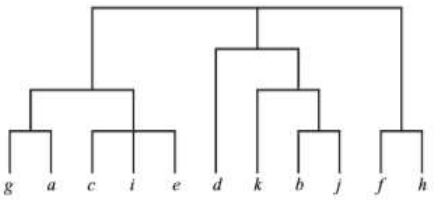
Dalam kasus yang paling sederhana algoritme Kluster mampu menghasilkan luaran berupa partisi-partisi kumpulan objek, dimana objek dalam satu partisi tidak menjadi anggota dari partisi lainnya. Beberapa algoritme Kluster lainnya memungkinkan satu objek untuk menjadi anggota dari beberapa Kluster, sehingga objek tersebut terletak pada dua area irisan Kluster, model Kluster seperti ini dapat disajikan dalam bentuk diagram venn.

Pada beberapa kasus terkadang juga diperlukan algoritme Kluster yang dapat mengelompokkan objek-objek ke dalam Kluster sesuai dengan derajat keanggotaannya, dimana setiap objek akan memiliki total derajat keanggotaan terhadap semua Kluster sebesar 1. Pada kasus lainnya juga terdapat model Kluster yang disajikan secara hierarki, dimana terdapat Kluster teratas, kemudian Kluster atas ini akan memiliki *sub*-Kluster, dan akan turun lagi sampai ke level tertentu, sehingga membentuk struktur pohon (*dendogram*). Berikut merupakan diagram-diagram yang mampu dihasilkan oleh algoritme Kluster :



	1	2	3
a	0.4	0.1	0.5
b	0.1	0.8	0.1
c	0.3	0.3	0.4
d	0.1	0.1	0.8
e	0.4	0.2	0.4
f	0.1	0.4	0.5
g	0.7	0.2	0.1
h	0.5	0.4	0.1
...			

(c)



Gambar 2.4: Berbagai diagram untuk merepresentasikan hasil Kluster

Sumber : Gorunescu, (2011)

Dari beberapa algoritme Kluster yang ada, secara prinsip metodologi Kluster meliputi 2 kategori, yakni :

1. *Hierarchical Clustering*, dimana Kluster terbentuk secara hierarkis membentuk struktur pohon sampai suatu kondisi terpenuhi. Algoritme yang termasuk dalam kategori ini adalah : *single linkage, complete linkage, average linkage, centroid method dan ward's method*.
2. *Non-hierarchical/Partitional/Flat Clustering*, merupakan pendekatan Kluster yang memasukkan objek persis ke dalam sebuah Kluster. Yang termasuk dalam kategori algoritme ini adalah *K-Means* dan *SOM*.

2.4 *Vector Space Model*

Pada pembahasan sebelumnya telah dijelaskan representasi data yang memungkinkan kumpulan dokumen untuk bisa dikluster, yakni dengan mengonversi kumpulan dokumen menjadi struktur data *space vector model*. Contoh sederhana VSM adalah dengan menganggap sebuah artikel sebagai sebuah vektor frekuensi kata. Meskipun frekuensi kata mampu untuk menggambarkan topik suatu dokumen, namun dalam beberapa kondisi hal ini tidak selamanya dapat digunakan, khususnya ketika terdapat kata yang sering muncul pada satu dokumen, namun juga sering muncul pada dokumen lainnya. Kata dengan frekuensi tinggi tersebut belum bisa menggambarkan ciri dokumen secara unik. Sehingga diperlukan bobot lain yang bisa memberikan identifikasi suatu dokumen secara unik.

Term Frequency/Inverse Document Fequency (TF/IDF) merupakan sebuah mekanisme pembobotan yang mampu menjadi solusi dari permasalahan frekuensi kata. Secara sederhana bobot *TF/IDF* akan memberikan nilai yang tinggi untuk *term* yang sering muncul di satu dokumen namun jarang muncul di dokumen lainnya, serta nilai yang rendah untuk term yang berfrekuensi tinggi pada suatu dokumen namun juga sering muncul di dokumen lain. Persamaan bobot *TF/IDF* dapat dilihat pada persamaan berikut :

$$tf - idf_{t,d} = tf_{t,d} \times idf_t \qquad (2.1)$$

$tf - idf_{t,d}$ merupakan bobot *TF/IDF* $term_t$ pada $dokumen_d$, $tf_{t,d}$ merupakan frekuensi $term_t$ pada $dokumen_d$ dan idf_t merupakan nilai idf dari $term_t$, Dimana :

$$idf_{t,d} = lod\frac{N}{df_t} \tag{2.2}$$

N merupakan jumlah dokumen yang akan dikluster dan df_t adalah jumlah dokumen yang mengandung $term_t$. Sehingga sebuah *VSM* dapat dipandang sebagai kumpulan dari vektor bobot *TF/IDF* seperti berikut ini :

Tabel 2.2 : Contoh VSM yang menyimpan vektor *TF/IDF*

	Term1	Kata2	Kata3	Kata4	KataN
Dok1	$tf - idf_{1,1}$	$tf - idf_{2,1}$	$tf - idf_{3,1}$	$tf - idf_{4,1}$	$tf - idf_{n,1}$
Dok2	$tf - idf_{1,2}$	$tf - idf_{2,2}$	$tf - idf_{3,2}$	$tf - idf_{4,2}$	$tf - idf_{n,2}$
DokN	$tf - idf_{1,n}$	$tf - idf_{2,n}$	$tf - idf_{3,n}$	$tf - idf_{4,n}$	$tf - idf_{n,n}$

Sebuah proses Kluster adalah proses pengelompokan objek-objek sesuai dengan kemiripannya ke dalam beberapa Kluster. Dari konsep tersebut jelas sekali diperlukan mekanisme pengukuran yang dapat digunakan untuk mengidentifikasi seberapa besar kemiripan antar objek, dalam hal ini seberapa besar kemiripan antar vektor dalam *VSM*. Salah satu instrumentasi yang dapat digunakan untuk mengukur kemiripan antar objek dokumen adalah *Cosine Index*.

Cosine Index, merupakan ukuran kemiripan antara 2 vektor pada dimensi n dengan cara mencari sudut di antaranya, ukuran kemiripan ini sering digunakan pada proses *text mining*. Pada 2 vektor atribut A dan B, *cosine similarity* di antara keduanya disajikan dengan menggunakan perkalian titik di antara keduanya :

$$\theta = arccos\frac{A.B}{||A||\,||B||} \tag{2.3}$$

2.5 *Latent Symantec Analysis*

Ciri kata yang terkandung dalam VSM, menggambarkan bobot kata berdasarkan frekuensi kemunculannya. Oleh sebab itu model ini dapat digunakan untuk membedakan ciri antar dokumen. Di satu sisi dalam memaknai sebuah kalimat ada pertimbangan lain yang juga perlu untuk diperhatikan, yakni sinonim

dan polisemi. Sinonim menggambarkan persamaan kata, sedangkan polisemi menggambarkan perbedaan makna terhadap kata tunggal yang bergantung pada konteksnya. Oleh sebab itulah selain menggunakan frekuensi kemunculan kata sebagai penanda ciri dokumen, penting pula untuk dipertimbangkan penggunaan ciri semantik dalam memaknai suatu dokumen.

Latent Symantic Analysis (LSA) merupakan teknik yang sering digunakan dalam *Natural Language Processing (NLP)* dalam mendeteksi ciri semantik dokumen dengan cara menggali struktur laten kemunculan kata. LSA memanfaatkan teknik aljabar linear *Singular Value Decomposition (SVD)*. Berikut ini beberapa terminologi yang berhubungan dengan teknik *SVD*.

Singular Value Decomposition (SVD) merupakan salah satu teorema pemfaktoran matriks A menjadi 3 buah matriks, yaitu ortogonal U, matriks diagonal S dan transpose dari matriks ortogonal V. Teorema tersebut dapat ditunjukkan di bawah ini :

$$A_{m\times n} = U_{m\times k} S_{k\times k} V_{k\times n} \tag{2.4}$$

Dimana $U^T U = I, V^T V = I$; Kolom-kolom U merupakan vektor eigen yang ortonormal dari AA^T, kolom-kolom V merupakan vektor eigen yang ortonormal dari $A^T A$ dan S merupakan matriks diagonal yang mengandung akar pangkat dari nilai eigen U atau V dalam posisi yang terurut. Hasil SVD juga dapat ditulis dengan interpretasi yang berbeda. Jika himpunan vektor yang menyusun U adalah $\vec{u}_1, \vec{u}_2 \ldots \vec{u}_n$, kemudian entri diagonal utama S adalah $\sigma_1, \sigma_2 \ldots \sigma_n$ dan himpunan vektor penyusun V adalah $\vec{v}_1, \vec{v}_2 \ldots \vec{v}_n$, maka teorema SVD dapat ditulis sebagai berikut :

$$A = \sum_{i=1}^{k} \sigma_i \vec{u}_i \vec{v}_i^T \tag{2.5}$$

Nilai-nilai $\sigma_1$diurutkan dari yang terbesar sampai dengan yang terkecil. Apabila beberapa nilai σ_i yang besar diambil dan nilai σ_i yang mendekati nol dibuang, maka akan diperoleh aproksimasi dari matriks A. Terdapat beberapa tahapan yang bisa dilakukan untuk mencari *SVD* :

a. Mencari Nilai U
   a.1. Mencari nilai AA^T
   a.2. Mencari vektor eigen dan nilai eigen dari AA^T
   a.3. Mengurutkan vektor eigen berdasarkan nilai eigen dari yang terbesar ke yang terkecil
   a.4. Melakukan proses ortonormalisasi gram-schmidt terhadap vektor-vektor eigen yang telah terurut.
b. Mencari Nilai V
   b.1. Mencari nilai $A^T A$
   b.2. Mencari vektor eigen dan nilai eigen dari $A^T A$
   b.3. Mengurutkan vektor eigen berdasarkan nilai eigen dari yang terbesar ke yang terkecil
   b.4. Melakukan proses ortonormalisasi gram-schmidt terhadap vektor-vektor eigen yang telah terurut.
   b.5. Melakukan proses transpose terhadap matriks ortonormal vektor-vektor eigen.
c. Mencari Nilai S
   Membuat matriks diagonal S dengan cara mengakar pangkatkan nilai-nilai eigen dari U dari yang terbesar ke yang terkecil.

2.6 *Self Organizing Map* (SOM)

SOM merupakan salah satu metode dalam jaringan saraf tiruan yang dapat digunakan untuk jenis pembelajaran *unsupervised*. Berbeda dengan metode *backpropagation* yang memerlukan label atau target, pendekatan SOM tidak bertujuan untuk mencari label, melainkan mengelompokkan observasi ke dalam grup-grup yang mirip, oleh sebab itulah pendekatan ini termasuk dalam kategori *unsupervised learning*. Metode ini pertama kali dikemukakan oleh Prof. Tuevo Kohonen pada tahun 1982.

Prinsip dasar dari SOM adalah mencoba untuk mentransformasikan *input* observasi yang berdimensi banyak menjadi bentuk yang lebih sederhana (biasanya berdimensi sedikit, bisa 1, 2 atau 3), sehingga karakteristik dari objek yang diobservasi lebih mudah dipahami. SOM memiliki keunggulan dalam hal

kemudahan implementasi dan mampu menyelesaikan permasalahan kompleks yang non linier.

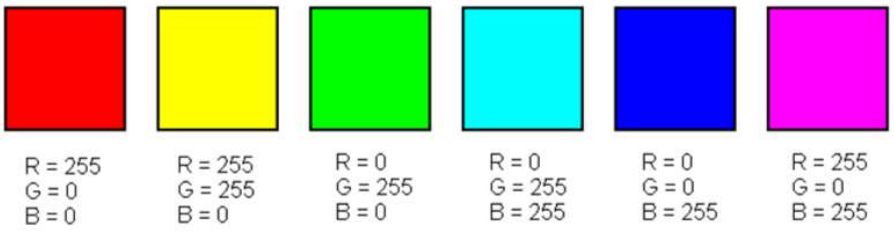
Berikut ini merupakan ilustrasi sederhana bagaimana metode SOM dapat digunakan untuk memecahkan suatu permasalahan :

- Kasus

Terdapat beberapa data yang berisi informasi warna. Atribut informasi warna yang terkandung dalam setiap rekaman data memiliki 3 atribut, yakni *red, green* dan *blue*. Data-data tersebut rencana akan dikelompokkan sesuai dengan kedekatan unsur RGB yang terkandung di dalamnya . Hasil dari pengelompokan data-data warna selanjutnya disajikan dalam bentuk matriks 2 dimensi (Kotak RGB).

- *Input* Data

Atribut warna yang dimiliki oleh setiap data dapat dianggap sebagai dimensi *input* (3 dimensi), sedangkan hasil penyajian pengelompokan data dapat dianggap sebagai dimensi *output* (2 dimensi). Secara sederhana SOM akan digunakan untuk menyederhanakan karakteristik data yang semula 3 dimensi menjadi 3 dimensi. Contoh *input* data adalah sebagai berikut :



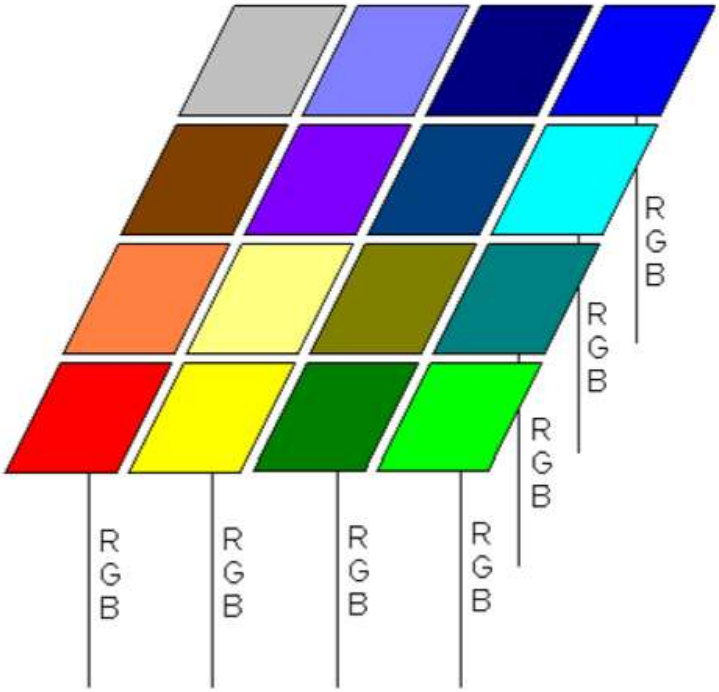
Gambar 2.5 : *Input* data RGB

Sumber : (Pang 2003)

- Jaringan Saraf (*Neural Network*)

Jaringan saraf dari SOM itu sendiri adalah grid dari neuron. Tiap-tiap neuron mengandung bobot vektor(*weight vector*) yang direpresentasikan dari nilai RGB dan lokasi geometri grid tersebut. Bobot vektor akan digunakan untuk menentukan "winning" neuron dari tiap *input* dan akan meng-update berdasarkan lokasi selama proses pembelajaran(*Training*). Inisialisasi dari jaringan saraf untuk permasalahan di atas dapat dilihat pada gambar 2.6. Kotak persegi pada gambar tersebut merepresentasikan neuron dan garis vertikal di masing-masing kotak

menunjukkan inisial dari nilai RGB. Bobot neuron dapat berupa inisial acak atau yang dihasilkan sebelumnya :

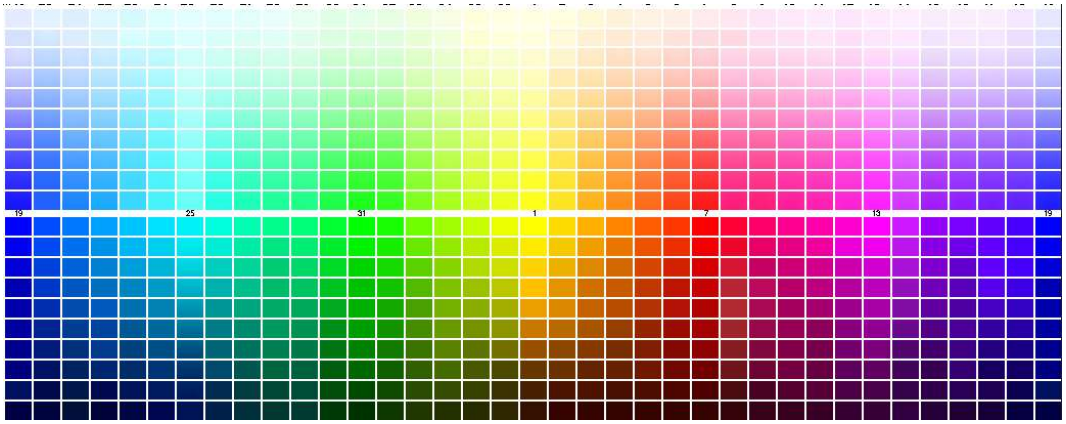


Gambar 2.6 : Inisialisasi jaringan saraf pada kotak RGB

Sumber : (Pang 2003)

- *Output* Data

Setelah dilakukan inisialisasi jaringan saraf, SOM akan melakukan pengelompokan data-data masukan sesuai dengan karakteristiknya (atribut RGB), sehingga didapatkan luaran dengan visualisasi yang kurang lebih sebagai berikut :

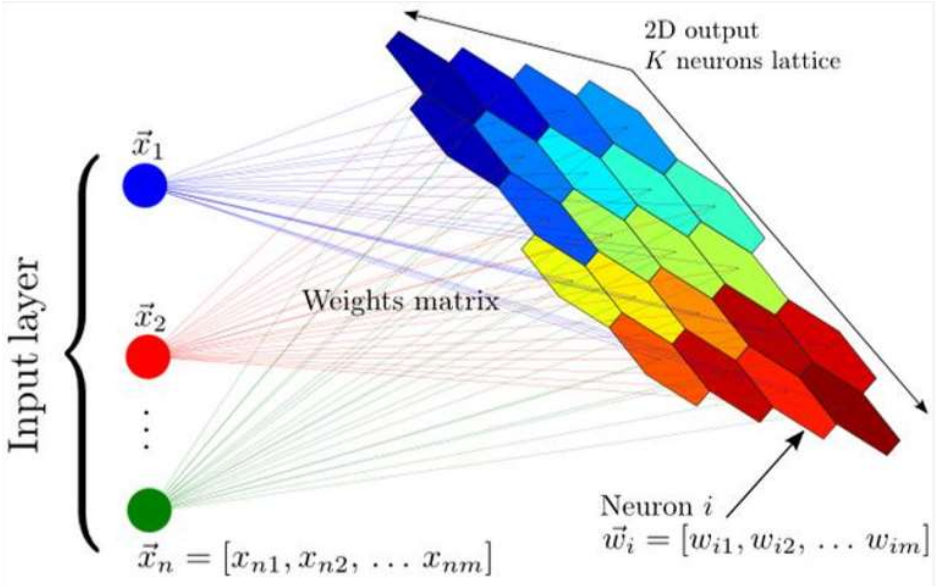


Gambar 2.7 : Contoh luaran SOM untuk vektor RGB acak.

Sumber : (Pang 2003)

2.6.1 Struktur SOM

Secara sederhana struktur SOM terdiri atas 2 *layer*, yakni *layer input* dan *layer output* (*neuron*). Setiap *node* pada layer *input* akan terhubung keseluruh *node* pada *neuron*, namun setiap *neuron* tidak terhubung satu sama lain, begitu juga dengan setiap *node* pada *layer input* juga tidak terhubung antara satu dengan lainnya.



Gambar 2.8 : Struktur SOM dengan *input* dan *output layer*.

Sumber : (Pang 2003)

Layer output dapat berdimensi 1 atau 2. *Layer* ini juga dapat disebut sebagai *computational layer atau competitive layer* karena *node -node* pada *layer* ini berkompetisi dan akan selalu di-*update* bobotnya. Baik *input* maupun *output node* merupakan *array* yang menyimpan informasi karakteristik, atribut maupun ciri. Bobot (*weight*) ini merepresentasikan hal-hal tersebut.

2.6.2 Proses Pemetaan SOM

Untuk dapat memetakan data *input* menjadi *output*, SOM dibagi menjadi 2 tahapan. Berikut merupakan tahapan-tahapan pada SOM :

a. Pembuatan Topologi

Misal S adalah himpunan *node output* yang diinginkan, sedangkan I adalah himpunan vektor *node input* $v \in I$. Untuk masing-masing posisi *node* $n_i \in S$ harus ditata terlebih dahulu sesuai dengan peta *output* yang diinginkan. Begitu

juga bobot pada masing-masing n_i harus diberikan dengan kaidah $\dim(n_{i_{bobot}}) = \dim(I)$.

b. Pengorganisasian

Agar SOM dapat beroperasi, terdapat beberapa parameter yang harus diatur terlebih dahulu. Secara umum keseluruhan proses dilakukan selama beberapa kali daur hidup pembelajaran, pada beberapa referensi parameter ini sering disebut dengan *epoch*. Selanjutnya, perilaku adaptasi S terhadap I dipengaruhi oleh 2 hal, yakni laju (μ) pembelajaran dan radius ketetanggaan (σ) n_i terhadap *node* pemenang ($n_{pemenang}$). Berikut ini merupakan rincian proses pengorganisasian :

1. Penentuan *Node* Pemenang

   Node pemenang $n_{pemenang} \in S$ ditentukan oleh jarak antara I terhadap n_i. $n_{pemenang}$ dipilih dari n_i yang memiliki jarak terendah terhadap I. $n_{pemenang}$ harus memenuhi kondisi :

   $$\forall n_i \in S : diff(n_{pemenang_{bobot}}, v) \leq diff(n_{i_{bobot}}, v) \qquad (2.6)$$

   Oleh sebab itu fungsi $bmu(v)$ (*best matching unit*) yang dapat menentukan $n_{pemenang}$ pada vektor *input* v dapat didefinisikan sebagai :

   $$bmu(v) = \arg\min diff(v, n_{i_{bobot}}), i = 0, 1, 2, \dots, |S| - 1 \qquad (2.7)$$

   Dimana $diff(x, y)$ merupakan selisih antara x dan y , atau jarak antara x terhadap y. Jarak euclide merupakan fungsi jarak yang umum digunakan pada algoritme SOM :

   $$diff(x, y) = \sqrt{\sum_i (x_i - y_i)^2} \qquad (2.8)$$

2. Adaptasi Bobot

   Penentuan *node* pemenang disertai dengan adaptasi dari seluruh bobot S menurut persamaan berikut :

$$n_{i_{bobot}}(t+1) = n_{i_{bobot}}(t) + \mu(t).h\left(\left|n_{pemenang} - n_i\right|, t\right).(v_i - n_{i_{bobot}}(t)) \quad (2.9)$$

Persamaan di atas menunjukkan bahwa penyesuaian bobot dipengaruhi oleh laju pembelajaran μ dan fungsi ketetanggaan h.

Laju pembelajaran merupakan bilangan real yang bernilai $0 < \mu \leq 1$, dan akan selalu meluruh setiap pada siklus pembelajaran, sehingga $\mu(t+1) < \mu(t)$. Agar nilai μ dapat selalu meluruh setiap pergantian siklus, maka nilai μ dapat dibangkitkan dengan fungsi peluruhan eksponensial :

$$\mu(t) = \mu_0.k^{\left(\frac{t}{t_{max}}\right).\left(\log\left(\frac{\mu_f}{\mu_0}\right).\frac{1}{\log(k)}\right)} \quad (2.10)$$

Dimana μ_0 merupakan nilai inisial μ dan μ_f merupakan nilai μ akhir yang diinginkan. Untuk peningkatan performa algoritme, Haykin menyederhanakan persamaan di atas menjadi :

$$\mu(t) = \mu_0.e^{-\frac{t}{t_{max}}} \quad (2.11)$$

Fungsi ketetanggaan menentukan seberapa besar jumlah *node* yang terpengaruh oleh proses penyesuaian yang dialami oleh *node* pemenang. Secara sederhana, bukan hanya *node* pemenang yang mengalami proses penyesuaian, namun *node -node* yang berada di sekitar *node* pemenang juga ikut mengalami penyesuaian. Semakin jauh jarak sebuah *node* dengan *node* pemenang, maka semakin kecil pula peluang *node* tersebut untuk beradaptasi. Salah satu fungsi ketetanggaan yang umum digunakan adalah fungsi Gaussian :

$$h(d) = e^{-\frac{d^2}{2.\sigma^2}} \quad (2.12)$$

Dari persamaan di atas, fungsi ketetanggaan tidak hanya bergantung pada jarak d saja, namun juga bergantung pada radius σ. Disisi lain ukuran radius σ seharusnya selalu mengecil setiap siklus pembelajaran, sehingga

notasi σ pada siklus t dapat diganti dengan $\sigma(t)$ dan fungsi ketetanggaan pada jarak d dan siklus t dapat dinotasikan dengan $h(d,t)$:

$$h(d,t) = e^{-\frac{d^2}{2.\sigma(t)^2}} \quad (2.13)$$

Nilai dari radius $\sigma(t)$ dapat diperlakukan sama dengan nilai dari μ yang juga selalu meluruh setiap pergantian siklus pembelajaran. Sehingga :

$$\sigma(t) = \sigma_0 . e^{\left(-\frac{t.\log \sigma_0}{t_{max}}\right)} \quad (2.14)$$

2.7 Pengujian *Kluster*

Terdapat beberapa cara dalam mengevaluasi kualitas Kluster. Sebuah Kluster dapat dikatakan memiliki kualitas yang bagus apabila mampu mengelompokkan dokumen-dokumen yang homogen ke dalam Kluster yang sama serta memisahkan dokumen-dokumen yang heterogen ke dalam beberapa Kluster yang berbeda. Kualitas dari proses *clustering* dapat diuji secara eksternal maupun internal. Pengujian internal dimaksudkan untuk mengukur kualitas homogenitas suatu Kluster sedangkan pengujian dimaksudkan untuk menguji perbandingan antara pengelompokan hasil Kluster dengan pengelompokan yang sebenarnya.

F-Measure merupakan pengujian yang sering digunakan dalam mengukur kualitas proses *clustering*. Untuk menghitung *F-Measure* ini terdapat beberapa parameter yang harus diketahui terlebih dahulu, yakni :

a. Precission

Presisi merupakan rasio jumlah objek yang tepat dalam suatu Kluster dengan jumlah seluruh anggota dalam Kluster tersebut. Nilai presisi dapat diketahui melalui persamaan :

$$P = \frac{TP}{TP + FP} \quad (2.15)$$

Dimana *True Positif (TP)* merupakan dua dokumen yang sama, terletak pada Kluster yang sama. Sedangkan False Positif (FP) merupakan dua dokumen yang berbeda, terletak pada Kluster yang sama.

b. Recall

Recall merupakan rasio jumlah objek yang tepat dalam suatu Kluster dengan seluruh jumlah objek yang seharusnya menjadi anggota pada Kluster tersebut. Nilai *Recall* dapat diketahui melalui persamaan :

$$R = \frac{TP}{TP + FN} \tag{2.16}$$

Dimana *True Positif (TP)* merupakan dua dokumen yang sama, terletak pada Kluster yang sama. Sedangkan False Negatif (FN) merupakan dua dokumen yang sama, terletak pada Kluster yang berbeda.

Setelah nilai presisi dan *recall* diketahui, selanjutnya nilai *F-Measure* dapat dicari dengan persamaan :

$$F_\beta = \frac{(\beta^2 + 1)PR}{\beta^2 P + R} \tag{2.17}$$

Nilai *F-Measure* dapat memberikan bobot yang kuat terhadap nilai *FN* dibanding *FP* dengan cara mengatur nilai $\beta > 1$.