BAB II
TINJAUAN PUSTAKA DAN DASAR TEORI

2.1 Tinjauan Pustaka

Berbagai penelitian yang menganalisis teks dengan metode penambangan teks telah dilakukan oleh beberapa peneliti sebelumnya. Seperti yang dilakukan peneliti Ur-Rahman dan Harding (2012) dalam penelitiannya menggunakan metode *textual data mining* untuk media pengelolaan pengetahuan industrial dan klasifikasi teks dengan model pendekatan berbasis bisnis. Penelitian ini mengaplikasikan teknik *clustering* yang kemudian dilakukan klasifikasi menggunakan beberapa algoritma *classifier* untuk dibandingkan hasil yang terbaik. *Classifier* yang digunakan diantaranya *C4.5*, *K-NN*, *Naïve* Bayes dan *Support vector machine*. Penelitian ini menyimpulkan bahwa algoritma C4.5 menunjukkan hasil akurasi terendah dibandingkan dengan yang lain (Ur-Rahman dan Harding, 2012).

Penelitian Ordenes dkk. (2014), melakukan analisis umpan balik pengalaman pelanggan dengan metode *text mining* melalui pendekatan berbasis linguistik untuk memperoleh wawasan persepsi pelanggan dan memberikan rekomendasi untuk perusahaan dari sisi terpusat pada pelanggan. Salah satu kesimpulan yang didapatkan bahwa ulasan pelanggan dapat dibagi dalam beberapa tipe: langsung/tidak langsung, terstrutur/tidak terstruktur, eksplisit/implisit.

Secara spesifik penelitian dengan menggunakan metode *text mining* untuk memproses data teks dari sumber TripAdvisor juga telah dilaksanakan sebelumnya. Penelitian Moro dkk. (2017), menggunakan pendekatan *data mining* untuk pemodelan skor pada TripAdvisor dengan sumber data masukan sejumlah 504 ulasan pelanggan pada 21 hotel. *Support Vector Machine* (SVM) digunakan untuk melakukan komputasi sembilan belas karakteristik fitur kuantitatif yang mencakup ulasan, hotel dan pengguna, yang menghasilkan pengetahuan berupa faktor-faktor yang memberikan pengaruh (*influence*) pada pengguna untuk memberikan skor tertentu (Moro dkk., 2017). Sezgen dkk (2019) menginvestigasi isu utama kepuasan

penumpang dalam kategori penerbangan *full-services* dan *low-cost carriers*, juga penumpang *economy-cabin* dan *premium-cabin*.

Metode *Latent Semantic Analysis* (LSA) digunakan untuk menganalisis lebih dari lima ribu ulasan penumpang dari total lima puluh perusahaan penerbangan. Hasil yang didapatkan berupa kumpulan kata-kata utama yang terhubung dengan kategori value, *staff*, food dan services. Kata-kata ini mengungkapkan faktor-faktor kepuasan atau ketidakpuasan oleh penumpang (Sezgen dkk., 2019). Cozza dkk (2018) menganalisis sekumpulan *dataset* ulasan pelanggan hotel pada laman TripAdvisor menggunakan metode *Association Rule Mining* (ARM) dan perhitungan statistikal untuk mengekstraksi ulasan pelanggan dan mendapatkan wawasan tentang faktor penentu yang mempengaruhi pelanggan dengan karakter spesifik selalu memberikan *review* spesifik dengan karakter hotel yang disukai (Cozza dkk., 2018).

Penelitian Isa dkk. (2008), menggunakan pendekatan metode hibrid yaitu kombinasi formula *Naïve* Bayes untuk vektorisasi data teks yang selanjutnya digunakan untuk komputasi dengan *Support Vector Machine*, yang akan menghasilkan klasifikasi sesuai kategori yang terdapat pada ulasan pelanggan. Penelitian tersebut juga membandingkan penggunaan beberapa algoritma untuk mencari nilai akurasi dan kecepatan yang terbaik, diantaranya penggunaan algoritma *Naive* Bayes, SVM, *NB-SVM hybrid*, *Lsquare*, dan *TFIDF-SVM hybrid*. Dari hasil perbandingan menggunakan matriks *classification ranking techniques* menunjukkan secara rata-rata pendekatan algoritma *NB-SVM hybrid* mencapai nilai akurasi terbaik hingga 97,08%.

Dari berbagai literatur dan penelitian sebelumnya menunjukkan bahwa analisis data tekstual dalam jumlah besar dapat dilakukan dengan berbagai metode algoritma seperti *C4.5*, *K-NN*, *Naïve* Bayes *Classifier* dan SVM. Setiap perhitungan algoritma tersebut menghasilkan nilai akurasi sesuai dengan karakteristik masing-masing. Model algoritma Bayes dan SVM dinyatakan mendapatkan akurasi yang lebih tinggi dibanding algoritma lainnya, hasil dari penelitian Isa dkk (2008) menyatakan metode hybrid Bayes – SVM bahkan mendapat nilai akurasi sangat

tinggi. Namun penerapan metode hibrid rumus Bayes dan SVM tersebut masih bekerja secara independen dan belum terintegrasi dalam satu sistem informasi.

Penelitian ini menjembatani kendala yang ada dalam penelitian – penelitian sebelumnya dimana satu atau lebih metode algoritma telah digunakan secara terpisah, berdiri sendiri dan bekerja secara individu untuk menyelesaikan permasalahan klasifikasi data teks. Hal ini menyebabkan beberapa kendala seperti waktu pemrosesan data meningkat sehingga tingkat efektivitas dan efisiensi untuk mendapatkan hasil keluaran menjadi berkurang. Penelitian sistem informasi analisis dengan metode rumus Bayes hybrid dan SVM ini mengajukan suatu metode baru dimana dua algoritma digabungkan dalam satu modul pemrosesan sistem untuk memproses data masukan secara paralel dan menyajikan hasil keluaran secara simultan dalam keluaran sistem ini sendiri sehingga proses klasifikasi data teks lebih efektif dan efisien.

2.2 Dasar Teori

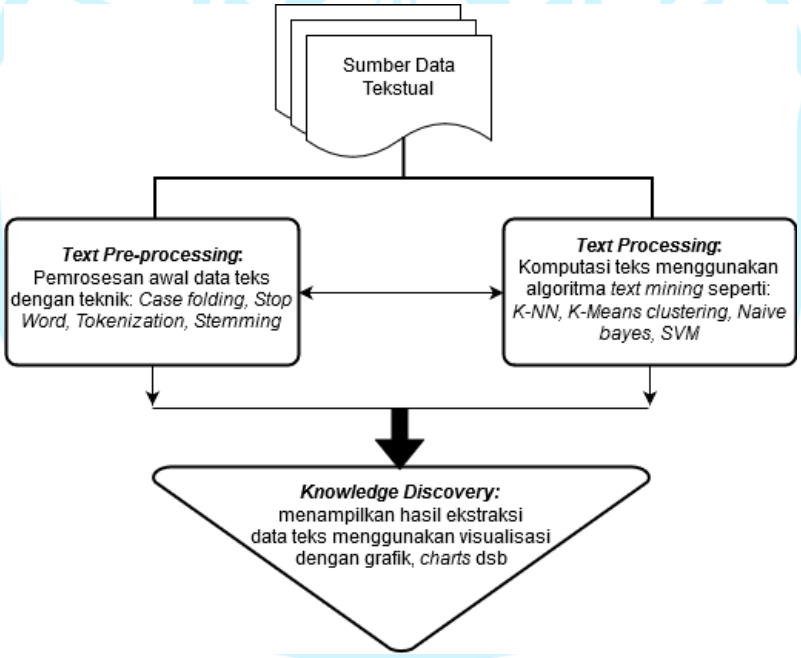
2.2.1 *Data Mining* dan *Text Mining*

Data mining adalah proses untuk mengekstraksi informasi implisit yang berasal dari massa, bersifat tidak lengkap, tidak terstruktur dan acak, sehingga dapat menjadi pengetahuan yang berpotensi memberi manfaat (Agrawal dkk., 1993). Penambangan data (*data mining*) merupakan salah satu proses yang dilakukan untuk menemukan pola dan pengetahuan dari data besar (Virk dan Chauhan, 2018). Perbedaan mendasar antara *data mining* dan analisis data secara tradisional misalnya proses pelaporan, permintaan, dan aplikasi analisis *online* adalah proses *data mining* yang dilakukan untuk menambang data dan menguak pengetahuan mendalam tentang premis asumsi yang tidak jelas (Chen dkk., 1996).

Di dalam proses *data mining* seringkali terdapat kendala ketika menangani basis data yang bersifat tidak terstruktur seperti misalnya data teks, gambar, audio dan lainnya. *Text mining* merupakan variasi metode dari *data mining* yang bertujuan untuk menemukan suatu pola dari kumpulan besar data tekstual. Metode

text mining dapat memberikan nilai tambah bagi suatu bisnis dengan memfasilitasi proses pengambilan keputusan dengan lebih efektif dan efisien dibandingkan teknik pemrosesan teks lainnya (Spinakis dan Chatzimakri, 2005).

Text mining juga berfungsi untuk melakukan pengelompokan/klasterisasi, ekstraksi informasi dan pengambilan kembali informasi (Berry dan Kogan, 2010). Di dalam proses *text mining* terdapat beberapa tahapan proses untuk menganalisis data tidak terstruktur yang memuat banyak derau, untuk kemudian diproses dengan algoritma *text mining* sehingga diperoleh hasil keluaran berupa penemuan pengetahuan. Gambar 2.1 menunjukkan tahapan proses di dalam *text mining* (Ur-Rahman dan Harding, 2012).



Gambar 2.1. Tahapan proses di dalam *text mining*

Pada Gambar 2.1 ditunjukkan tahapan proses penambangan teks, dimulai dari sumber *dataset* yang akan diolah. Setelah melalui tahap *pre-processing*, selanjutnya data teks diproses dengan algroitma seperti *Naïve* Bayes *Classifier* (NBC) dan *Support vector machine* (SVM). Sebagai hasil akhir dari pemrosesan *text mining* diperoleh ekstraksi pengetahuan yang dapat ditampilkan dalam bentuk grafik, *charts, table* dan sebagainya.

2.2.2 Prapengolahan Teks dan *Bag-of-Words model*

Di dalam penambangan teks, dibutuhkan prapengolahan teks (*text pre-processing*) sebagai tahapan awal untuk menyiapkan *dataset* sebelum diproses oleh algoritma. Secara definitif, *text pre-processing* merupakan suatu tahapan untuk mengubah data tekstual yang tidak terstruktur (*unstructured*) menjadi data terstruktur (*structured*) sehingga dapat diproses dengan algoritma klasifikasi selanjutnya (Miner, 2012). Data awal dipersiapkan (*data preparation*) dan diubah (*data transformation*) pada tahap *pre-processing*. Sebelum menggunakan algoritma *machine learning* untuk data latih, diperlukan cara untuk merepresentasikan sebuah dokumen teks sebagai fitur vektor.

Pemodelan yang umum digunakan dalam *Natural Language Processing* (NLP) untuk melakukan vektorisasi salah satunya adalah dengan menggunakan *Bag-of-Words* model (Rachka., 2014). Langkah – langkah dalam penerapan BoW model terdiri dari:

a.) Membuat *vocabulary*.

Langkah pertama dalam *bag-of-words* (BoW) adalah membuat *vocabulary*, yaitu kumpulan kata-kata yang berbeda yang muncul dalam dokumen latih, dan setiap kata diasosiasikan dengan jumlah kemunculannya. Urutan kata dalam *vocabulary* ini dapat diabaikan. Sebagai contoh, terdapat dua data dokumen $\boldsymbol{D_1}$ dan $\boldsymbol{D_2}$ dalam data latih sebagai berikut:

[$\boldsymbol{D_1}$]: "*Each plane has its own seats*."
[$\boldsymbol{D_2}$]: "*Every airlines has its own passengers*."

Berdasarkan data pada dua dokumen di atas, dapat ditulis *vocabulary* sebagai berikut: V = {*each*: 1, *plane*: 1, *has*: 2, *its*: 2, *own*: 2, *seats*: 1, *every*: 1, *airlines*: 1, *passengers*: 1}. Selanjutnya *vocabulary* tersebut kemudian digunakan untuk membangun *d-dimensional feature vectors* untuk setiap individu dokumen dimana dimensionalitas sama dengan jumlah kata yang berbeda dalam *vocabulary*

($d = |V|$). Proses ini yang dinamakan dengan vektorisasi (*vectorization*). Tabel 2.1 menunjukkan representasi dua contoh dokumen $\boldsymbol{D_1}$ dan $\boldsymbol{D_2}$ sebagai berikut:

Tabel 2.1 Penerapan model *bag-of-words* dalam dokumen D_1 dan D_2

Masukan	*each*	*plane*	*has*	*its*	*own*	*seats*	*every*	*airlines*	*passengers*
$\mathbf{X}_{D1}$	1	1	1	1	1	1	0	0	0
$\mathbf{X}_{D2}$	0	0	1	1	1	0	1	1	1
$\sum$	1	1	2	2	2	1	1	1	1

Dalam Tabel 2.1 di atas dinyatakan *features vector* dari setiap kata dalam angka biner 1 apabila kata tersebut muncul dalam dokumen, dan 0 apabila kata tersebut tidak muncul dalam dokumen. Pemilihan masukan kata dalam dokumen untuk menyusun sebuah *vocabulary* di atas dapat dilakukan dengan proses tokenisasi teks.

b.) Tokenisasi

Tokenisasi (*tokenization*) merupakan proses untuk menjabarkan sebuah *text corpus* kedalam elemen individual yang dapat menjadi masukan untuk berbagai varian algoritma NLP. Untuk mendapatkan hasil perhitungan optimal, tokenisasi dilakukan bersamaan dengan langkah pemrosesan lain, diantaranya penghapusan *stop words* (*stop words removal*) dan karakter penghubung (*punctuation characters*), proses *stemming* dan *lemmatizing*, serta membangun *N-grams*. Tabel 2.2 menunjukkan proses tokenisasi sebuah kalimat dalam dokumen sebagai berikut:

Tabel 2.2. Contoh penerapan tokenisasi

	Contoh kalimat dalam dokumen						
Masukan	*A traveller likes travelling, thus he travels.*						
Keluaran	*A*	*traveller*	*likes*	*travelling*	*thus*	*he*	*travels*

Tabel 2.2 menunjukkan contoh sederhana tokenisasi yang membagi kalimat menjadi kata per kata, menghilangkan karakter *punctuation* dan konversi semua kata dalam huruf ke*Ci*l.

c.) *Stop Words Removal*

Stop words merupakan bagian umum dari *text corpus* yang dipertimbangkan sebagai kata tidak informatif dan tidak berpengaruh signifikan pada sebuah dokumen (misalnya kata: *a, so, or, the,* dan sebagainya). Salah satu pendekatan untuk menghapus *stop words* adalah dengan mencari daftar kata-kata dalam bahasa inggris pada *stop words removal library*. Alternatif lainnya adalah dengan membuat sendiri daftar *stop words* yang akan dihapus. *Stop words list* tersebut kemudian digunakan untuk menghapus seluruh kata tersebut yang muncul dalam masukan dokumen, dalam urutan ranking *n words* tertinggi dalam daftar *stop words*. Tabel 2.3 menunjukkan contoh penerapan *stop words removal* pada masukan dokumen.

Tabel 2.3. Contoh penerapan *stop words* removal

	Contoh kalimat dalam dokumen				
Masukan	*A traveller likes travelling, thus he travels.*				
Keluaran	*traveller*	*likes*	*travelling*	*,*	*travels*

Tabel 2.3 di atas menunjukkan proses *stop words removal* yaitu menghapus kata "*a*", "*thus*", dan "*he*". Kata-kata yang dihapus tersebut tidak berpengaruh terhadap hasil akhir klasifikasi.

d.) *Stemming* dan *Lemmatization*

Stemming, pertama kali dikembangkan oleh Martin F Porter pada 1979 dan dikenal dengan istilah *Porter Stemmer* (Porter., 1979), dapat diartikan sebagai proses untuk mengubah (*transforming*) sebuah kata menjadi kata aslinya (*root form*). Tabel 2.4 menunjukkan proses stemming sebagai berikut:

Tabel 2.4 Contoh proses *Porter stemming*

Masukan	*A traveller likes travelling, thus he travels.*								
Keluaran	*A*	*traveller*	*like*	*travel*	*,*	*thu*	*he*	*travel*	*.*

Proses *stemming* seperti ditunjukkan Tabel 2.4 di atas mampu mengubah kata "*likes*" menjadi kata asli "*like*", kata "*travelling*" menjadi "*travel*" dan kata "*travels*" menjadi kata "*travel*". Proses *stemming* juga dapat menghasilkan kata yang tidak bermakna, misalnya kata "*thu*" yang merupakan asal kata untuk "*thus*". Untuk menyeimbangkan proses tersebut, digunakan proses *lemmatization*. Tabel 2.5 menunjukkaan contoh proses *lemmatization*:

Tabel 2.5 Contoh proses *lemmatization*

Masukan	A traveller likes travelling, thus he travels.								
Keluaran	A	traveller	like	travel	,	thus	he	travel	.

Proses lemmatization bertujuan untuk mencapai *canonical* atau perbaikan *grammar* (*grammatically correction*). Secara komputasional, proses *lemmatization* lebih sulit dan lebih kompleks daripada *stemming*, dan dalam prakteknya kedua proses *stemming* maupun *lemmatization* hanya memberi dampak ke*Ci*l terhadap proses klasifikasi secara keseluruhan (Toman dkk., 2006).

e.) N-grams Model

Di dalam *N-grams model*, sebuah token dapat didefinisikan sebagai bagian dari *n items*. Contoh sederhana penerapan *unigram*, yaitu *N-grams* yang didefinisikan sebagai 1-gram dimana dalam kalimat dipenggal menjadi tepat satu kata, huruf, atau simbol. Begitu pula dengan penerapan bi-gram (2-gram), tri-gram (3-gram) dan seterusnya. Tabel 2.6 menunjukkan penerapan *N-grams* model.

Tabel 2.6. Penerapan *N-grams* model

Masukan	*A traveller likes travelling, thus he travels.*						
1-gram	*A*	*traveller*	*likes*	*travelling*	*thus*	*he*	*travels*
2-gram	*A traveller*	*traveller likes*	*likes travelling*	*travelling thus*	….		
3-gram	*A traveller likes*	*Traveller likes travelling*	*Likes travelling thus*	……			

Memilih jumlah *n* untuk pemodelan *n-grams* yang optimal tergantung dari bahasa yang digunakan atau penerapan aplikasi masing-masing. Sebagai contoh penerapan *n-grams* ukuran $4 \leq n \leq 8$ dinyatakan memiliki akurasi yang baik untuk klasifikasi buku berbahasa inggris (Keselj dkk., 2003), sementara penelitian (Kanaris dkk., 2007) menyatakan *n-grams* antara 3 dan 4 lebih optimal untuk klasifikasi *anti-spam filtering* pada pesan *e-mail*.

2.2.3 Rumus Bayes untuk Klasifikasi Kategori

Rumus Bayes merupakan salah satu metode algoritma klasifikasi yang populer untuk keperluan *data mining* karena penggunaannya yang mudah (Hall, 2006), waktu pemrosesan yang relatif lebih cepat, struktur yang sederhana sehingga mudah diimplementasikan dan memiliki tingkat efektifitas yang tinggi (Taheri dan Mammadov., 2013). Algoritma *Naïve* Bayes merupakan algoritma yang digunakan untuk mencari nilai peluang tertinggi dan melakukan klasifikasi data teks yang diujikan pada kategori yang paling tepat (Feldman dan Sanger., 2007). Model peluang *Naïve* Bayes merupakan klasifikasi linier yang menggunakan dasar teorema Bayes, dan penggunaan istilah *Naïve* datang dari asumsi bahwa fitur-fitur dalam *dataset* berdiri independen secara mutual (Rish, 2001).

Pada penelitian Isa dkk. (2008) menggunakan pendekatan hibrid yaitu integrasi antara teknik rumus Bayes dengan *Support Vector Machine* (SVM). Sebuah dokumen teks elektronik dapat mengandung sejumlah besar fitur yang bermanfaat dalam klasifikasi teks, seperti misalnya kata kunci (*keywords*). Sehingga untuk mengatasi ketidakmampuan SVM dalam memproses format teks mentah (*raw text format)*, kata kunci yang terkandung dalam dokumen tersebut perlu untuk diubah menjadi nilai vektor yang dapat diproses oleh SVM. Metode

rumus Bayes dapat diusulkan untuk melakukan proses vektorisasi dan klasifikasi sesuai kategori yang sudah ditentukan. Namun pada penelitian ini terdapat batasan dimana algoritma Bayes digunakan untuk klasifikasi kategori dokumen saja, sementara SVM digunakan untuk analisis sentimennya.

Ketika digunakan untuk memproses *dataset* dalam jumlah besar *Naïve* Bayes classifier (NBC) terbukti memiliki tingkat akurasi yang cukup baik (Isa dkk., 2008; Aggarwal dkk., 2015). Algoritma *Naïve* Bayes memiliki kesederhanaan dalam proses komputasi, dan nilai akurasi yang dihasilkan akan semakin baik berbanding lurus dengan *dataset* yang dilatih, sementara algoritma SVM memiliki akurasi yang baik dengan data latih yang sedikit tetapi memerlukan komputasi yang lebih lama dan memakan sumber daya yang relatif lebih banyak (Isa dkk., 2008).

Dasar dari teorema Bayes dan turunannya seperti *Naive* Bayes *Classifier* dan sebagainya, merupakan formulasi yang disusun oleh Thomas Bayes (1701-1761) yang dapat dijabarkan sebagai berikut:

$$posterior\ probability = \frac{conditional\ probability\ .\ prior\ probability}{evidence}$$

Posterior probability, atau peluang akhir bersyarat dari sebuah hipotesis dapat diketahui dengan mengalikan nilai peluang bersyarat (*conditional probability*) dengan peluang awal (*prior probability*) dibagi nilai peluang suatu bukti (*evidence*). Atau dalam notasi umum persamaan di atas dapat dijabarkan sebagai berikut:

$$P(H|E) = \frac{P(E|H)\ .\ P(H)}{P(E)}$$

Posterior probability atau *P(H|E)* dapat didefinisikan sebagai peluang akhir bersyarat suatu hipotesis "*H*" terjadi jika diberikan bukti atau *evidence* "*E*" yang terjadi. Sedangkan nilai *conditional probability* atau *P(E|H)* merupakan peluang sebuah bukti "*E*" akan mempengaruhi nilai hipotesis "*H*". Sementara kondisi *prior probability* atau *P(H)* merupakan peluang awal hipotesis "*H*" yang terjadi tanpa memandang bukti sebelumnya. Sedangkan nilai pembagi *probability evidence* atau

P(E) merupakan peluang awal sebuah bukti "*E*" terjadi tanpa memandang hipotesis atau bukti lainnya (Rachka, 2014).

Dari dasar teorema Bayes dan pengembangan penelitian oleh Isa dkk. (2008) maka untuk klasifikasi kategori data teks pada penelitian ini, persamaan Bayes dapat ditulis ulang sebagai berikut:

$$Pr(Category|Word) = \frac{Pr(Word|Category) \, . \, Pr(Category)}{Pr(Word)} \quad (2.1)$$

Persamaan di atas dapat ditulis ulang sebagai berikut:

$$Pr(Ci|Wj) = \frac{Pr(Wj|Ci) \, . \, Pr(Ci)}{Pr(Wj)} \quad (2.2)$$

Keterangan dari persamaan tersebut:

Ci : merupakan kelas label kategori yang ditentukan (C_1,C_2,C_3 ... C_k)

Wj : merupakan kemunculan kata dalam dokumen masukan *X* (W_1,W_2,W_3 ... W_k) sebagai *evidence*

Pr(Ci|Wj): merupakan peluang kata *Wj* dalam dokumen masukan *X* yang berpeluang masuk pada kategori *Ci*

Pr(Wj|Ci): merupakan peluang bersyarat untuk jumlah kata *Wj* dalam kategori *Ci*

Pr(Ci): merupakan nilai peluang kategori *Ci*

Pr(Wj): merupakan nilai peluang kata *Wj* dalam dokumen masukan *X* sebagai *evidence*

Ci, merupakan kelas label yang sudah ditentukan untuk klasifikasi dokumen teks ke dalam 5 (lima) label kategori, yaitu: *Airplane* (C_1), *Comfort* (C_2), *Staff Services* (C_3), *Food & Entertainment* (C_4) dan *Price* (C_5). Hasil klasifikasi dokumen ke dalam label kategori tersebut dipengaruhi oleh nilai peluang kemunculan kata (*Wj*) dalam dokumen masukan sebagai *evidence*. Sebagai contoh dalam satu dokumen masukan terdapat kalimat: "*the seats on this airplane is big and the food is deliCious*", artinya dalam kalimat tersebut terdapat *evidence* berupa kemunculan kata (*Wj*) "*airplane*" yang memiliki peluang untuk masuk pada kategori *Airplane*

(C_1) dan kata "*food*" yang memiliki peluang untuk masuk pada kategori *Food & Entertainment* (C_4).

Persamaan (2.2) di atas merupakan perhitungan utama untuk menghitung peluang dokumen masukan *X*, dimana nilai *Pr(Ci)* sebagai *prior probability*, dapat dihitung dengan persamaan:

$$Pr(Ci) = \frac{\text{total kata dalam kategori } Ci}{\text{total kata dalam dokumen latih } (training)} \quad (2.3)$$

Sementara nilai peluang kata *Wj* dalam dokumen *X* atau *Pr(Wj)* dihitung dengan persamaan:

$$Pr(Wj) = \frac{\text{jumlah kemunculan kata } Wj \text{ dalam semua kategori}}{\text{jumlah semua kata dalam semua kategori}} \quad (2.4)$$

Metode yang sama dapat diimplementasikan untuk mencari jumlah keseluruhan kemunculan semua kata dalam semua kategori pada basis data pelatihan. Selanjutnya, untuk menghitung kemiripan kategori tertentu (*Ci*) yang terkandung dalam kata tertentu (*Wj*), maka daftar kemunculan *Wj* di dalam *Ci* pada basis data latih dicari dan kemudian menjumlahkan semua kata dalam *Ci* untuk mendapatkan nilai *Pr(Wj|Ci)* sesuai persamaan berikut:

$$Pr(Wj|Ci) = \frac{\text{kemunculan kata } wj \text{ dalam kategori } Ci}{\text{total kata dalam kategori } Ci} \quad (2.5)$$

Dari penjabaran sebelumnya maka seluruh parameter formula Bayes untuk klasifikasi teks telah lengkap meliputi nilai *prior probability*, *Pr(Ci)*, *likelihood*, *Pr(Wj|Ci)* dan *evidence*, *Pr(Wj)*, menghasilkan *posterior probability*, *Pr(Ci|Wj)*, dalam setiap kata pada dokumen masukan. Kemudian nilai *posterior probability* pada setiap kata yang dinotasikan pada kategori tertentu dimasukkan pada Tabel 2.7.

Tabel 2.7 Daftar kemunculan kata dan peluang untuk perhitungan Bayes

Peluang Word	**Peluang Kategori *C1***	**Peluang Kategori *C2***	**.....**	**Peluang Kategori *Ck***
W_1	*Pr(C1|W1)*	*Pr(C2|W1)*	.....	*Pr(Ck|W1)*
W_2	*Pr(C1|W2)*	*Pr(C2|W2)*	.....	*Pr(Ck|W2)*
.....	.....	.....	.....	.....
Wn	*Pr(C1|Wn)*	*Pr(C2|Wn)*	.....	*Pr(Ck|Wn)*
Total	$\sum Pr(C1\|Wn)$	$\sum Pr(C2\|Wn)$	.....	$\sum Pr(Ck\|Wn)$
Peluang Dokumen	$\frac{\sum Pr\ (C1\|Wn)}{n}$	$\frac{\sum Pr\ (C2\|Wn)}{n}$	.....	$\frac{\sum Pr\ (Ck\|Wn)}{n}$

Setelah seluruh kolom peluang diisi, maka *Ci* dapat dihitung dengan membagi total kolom peluang dengan panjang *query n*, sesuai persamaan:

$$Pr(Dokumen\ X) = \frac{\sum Pr(Ci|w1, w2, w3 \ldots wn)}{n} \quad (2.6)$$

Dimana nilai $n = \sum_{j=1}^{k} Wj$

Sebagai contoh, nilai peluang untuk dokumen *X* termasuk pada kategori tertentu (*Ci*) dapat dihitung dengan mencari peluang bersyarat untuk kemunculan kata-kata yang terkandung dalam dokumen *X* (*Wj*) sesuai dengan label kata yang terkandung dalam kategori *Ci*, dihitung dengan persamaan *Pr(Ci|Wj)*. Daftar kategori *Ci* disusun sebagai: C_1, C_2, C_3, ... C_k, sementara kemunculan kata *Wj* pada dokumen *X* disusun sebagai: W_1, W_2, W_3 ... W_n. Jumlah kategori dan jumlah kata dinotasikan dengan n. Tabel kemunculan kata dan peluang sesuai diatas telah menghasilkan nilai peluang sebuah kata dalam setiap kategori. Kolom "*Word*" diisi oleh kata-kata hasil ekstraksi dari dokumen masukan.

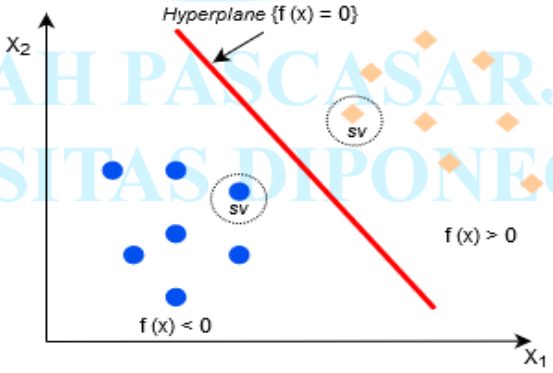
Sebelum algoritma menghitung peluang kata pada setiap kategori, diperlukan pelatihan dengan sekumpulan data latih yang telah dikategorikan secara manual. Setiap kata dari seluruh dokumen latih dalam kategori yang sama diektraksi dan dimunculkan dalam tabel kemunculan data sebagai variabel pelatihan. Metode *Naïve* Bayes *Classifier* mampu untuk menentukan kategori yang

tepat dari sebuah dokumen masukan dengan mengacu pada nilai peluang tertinggi yang dihitung oleh *trained classifier* sesuai formula Bayes. Kategori yang tepat direpresentasikan oleh kategori yang memiliki nilai *posterior probability*, *Pr(Category|Document)* tertinggi. Hasil akhir perhitungan rumus Bayes berupa klasifikasi dokumen sesuai kategori yang ditentukan, dan hasil vektorisasi pada Tabel 2.7 dapat digunakan sebagai data masukan untuk SVM.

2.2.4 *Support Vector Machine* untuk Analisis Sentimen

Support vector machine (SVM) merupakan algoritma klasifikasi yang menggunakan metode penerapan garis pembatas terbaik untuk melakukkan pemetaan data dari ruang masukan atau ruang parametrik ke dalam ruang fitur dimensional yang lebih tinggi (Ur-Rahman, 2012). Beberapa penelitian membuktikan bahwa SVM mampu meyajikan hasil klasifikasi dengan nilai akurasi yang cukup tinggi dibandingkan teknik klasifikasi lainnya (Isa dkk., 2008). Kendala dalam menerapkan SVM adalah ketidakmampuannya untuk memproses format teks mentah, sehingga untuk mengklasifikasi dokumen menggunakan SVM diperlukan proses vektorisasi yaitu proses untuk mengubah data teks menjadi *vector*.

Dalam klasifikasi dokumen, ada banyak cara untuk membuat garis pembatas klasifikasi yang linear, namun tujuan dari algoritma SVM adalah memilih pembatas *hyperplane* optimal (*optimal separating hyperplane*) untuk memaksimalkan *margin* diantara dua kelas yang berbeda (Vapnik, 1995). Gambar 2.2 menunjukkan *optimal separating hyperplane* yang mampu dilakukan oleh SVM.



Gambar 2.2 *Optimal separating hyperplane* pada SVM

Gambar 2.2 menunjukkan bahwa SVM merepresentasikan sebuah garis pembatas *hyperplane* yang dalam gambar ditunjukkan oleh garis berwarna merah dinotasikan oleh *f(x) = 0*, yang membagi ruang data secara geometris kedalam dua area berbeda sehingga menghasilkan klasifikasi data masukan menjadi dua kategori (Al-Amrani dkk., 2017). *Hyperplane* pemisah terbaik antara kedua kelas dapat ditemukan dengan mengukur *margin hyperplane* tersebut dan mencari titik maksimalnya. *Margin* merupakan jarak antara *hyperplane* dengan *pattern* terdekat dari masing- masing kelas. *Pattern* yang paling dekat ini disebut sebagai *support vector*, yang dalam gambar ditunjukkan oleh data point dengan lingkaran.

Dengan menggunakan SVM sebagai klasifikasi linier untuk membagi dua kelas data yang berbeda, asumsikan *data set* yang digunakan memiliki label x_i dan fitur y_i sebagai kelas label dengan $y_i \in \{-1, 1\}$, untuk $i = 1, 2, 3 \ldots n$, dimana n adalah jumlah data, dengan parameter W dan B, maka *hyperplane* dapat dicari dengan persamaan:

$$f(x) = W^T x + B \tag{2.7}$$

Dengan:
W: bobot (*weight*) vektor
x: masukan vektor
B: bias

Jika sebuah data *point* "*x*" termasuk dalam kategori positif apabila nilai *f(x)* > 0 dan termasuk kategori negatif apabila nilai *f(x)* < 0, dengan persamaan:

$$\begin{aligned} W^T x_i + B &\geq +1 \\ W^T x_i + B &\leq -1 \end{aligned} \tag{2.8}$$

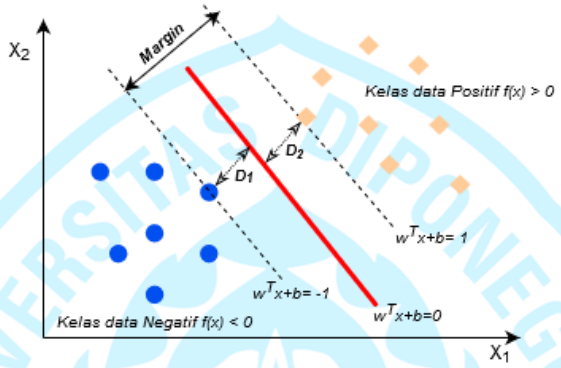
Persamaan 2.8 dapat ditulis ulang:

$$f(x) = sgn(W^T x + B) \tag{2.9}$$

Dimana *sgn( )* merupakan *sign function* yang secara matematis direpresentasikan dengan:

$$sgn(x) = \begin{cases} 1\ jika\ x > 0 \\ 0\ jika\ x = 0 \\ -1\ jika\ x < 0 \end{cases} \quad (2.10)$$

Selanjutnya Gambar 2.3 menunjukkan persamaan *hyperplane* tersebut:



Gambar 2.3 Persamaan *hyperplane* pada SVM

Untuk menghitung jarak D sebuah *data point x* dari *hyperplane* menggunakan persamaan:

$$D = \frac{|\ W^T x + B\ |}{|W|} \quad (2.11)$$

Selanjutnya *margin* antara *support vector* dihitung dengan persamaan:

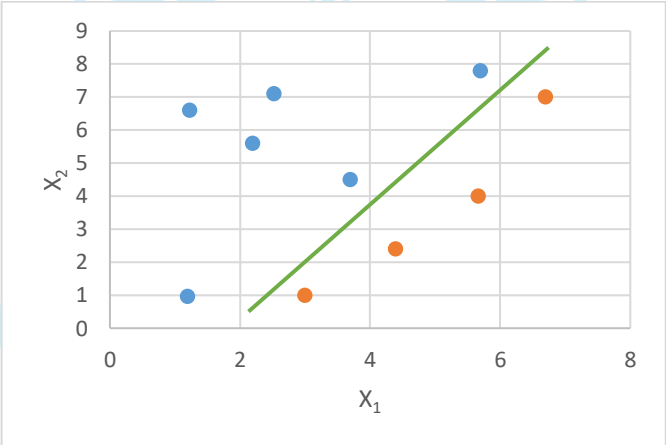
$$\frac{w}{\|w\|} \cdot (x_+ - x_-) = \frac{w^T(x_+ - x_-)}{\|w\|} = \frac{w^T\left(\left(\frac{+1-b}{w^T}\right) - \left(\frac{-1-b}{w^T}\right)\right)}{\|w\|} = \frac{2}{\|w\|} \quad (2.12)$$

Berikut ini merupakan contoh perhitungan SVM untuk klasifikasi linear dengan contoh vektor x_1 dan x_2 ditampilkan pada Tabel 2.8.

Tabel 2.8 Contoh perhitungan SVM untuk klasifikasi linear dua vektor

	Vektor x_1	**Vektor x_2**	**Label**	**Label (angka)**
D_1	1,23	6,6	positif	1
D_2	2,53	7,1	positif	1
D_3	1,2	0,97	positif	1
D_4	5,7	7,8	positif	1
D_5	2,2	5,6	positif	1
D_6	3,7	4,5	positif	1
D_7	4,4	2,4	negatif	-1
D_8	5,67	4	negatif	-1
D_9	6,7	7	negatif	-1
D_{10}	3	1	negatif	-1

Data vektor pada Tabel 2.8 dapat digambarkan dalam *hyperplane linear* seperti pada Gambar 2.4 berikut:



Gambar 2.4 *Hyperplane* klasifikasi *linear* pada vektor x_1 dan x_2

Menggunakan persamaan klasifikasi (2.8) dapat ditulis ulang sebagai berikut:

$$y = sign\ (w_1x_1 + w_2x_2 + b) \tag{2.13}$$

apabila $W_1x_1 + W_2x_2 + B >= 0$, maka kelasnya 1 (positif), apabila $W_1x_1 + W_2x_2 + B < 0$, maka kelasnya -1 (negatif). Maka tugas SVM adalah memprediksi

W_i dan B sehingga kelasnya tepat, asumsikan prediksi awal: $W_1 = (-2)$ $W_2 = 0{,}7$ $B = 2$, maka perhitungan SVM vektor x_1 dan x_2 ditunjukkan pada Tabel 2.9.

Tabel 2.9 Perhitungan SVM vektor x_1 dan x_2

Doc	x_1	x_2	$w_1x_1 + w_2x_2 + b$	$sign(w_1x_1 + w_2x_2 + b)$	**Label aktual**
			t	***y'***	***y***
D_1	1,23	6,6	4,16	1	1
D_2	2,53	7,1	1,91	1	1
D_3	1,2	0,97	0,279	1	1
D_4	5,7	7,8	-3,94	-1	1
D_5	2,2	5,6	1,52	1	1
D_6	3,7	4,5	-2,25	-1	1
D_7	4,4	2,4	-5,12	-1	-1
D_8	5,67	4	-6,54	-1	-1
D_9	6,7	7	-6,5	-1	-1
D_{10}	3	1	-3,3	-1	-1

Selanjutnya adalah menghitung *cost function* dengan persamaan:

$$J(\mathbf{w}) = \frac{\lambda}{2}\|\mathbf{w}\|^2 + \frac{1}{N}\sum_{i}^{n} \max\,(0,\ 1\ -\ y_i * (\mathbf{w} \cdot x_i + b)) \qquad (2.14)$$

Misalkan *lambda* = 1, maka *hinge loss* dihitung dengan persamaan:

$$\ell(y) = \max(0, 1 - t \cdot y) \qquad (2.15)$$

Tabel 2.10 Perhitungan *hinge loss* D_1 hingga D_{10}

Hinge loss	*Value*	*Hinge loss*	*Value*
D_1	0	**D_6**	3,25
D_2	0	**D_7**	0
D_3	0,721	**D_8**	0
D_4	4,94	**D_9**	0
D_5	0	**D_{10}**	0
Average loss			**0,8911**

Kemudian nilai *delta W* atau *J(W)* dapat dihitung dengan persamaan:

$$J(\mathbf{w}) = \frac{1}{N}\sum_{i}^{n}\left[\frac{1}{2}\|\mathbf{w}\|^2 + C\max(0,\ 1 - y_i * (\mathbf{w}\cdot x_i))\right]$$

$$\nabla_w J(\mathbf{w}) = \frac{1}{N}\sum_{i}^{n}\begin{cases}\mathbf{w} & \text{if } \max(0,\ 1 - y_i * (\mathbf{w}\cdot x_i)) = 0\\ \mathbf{w} - Cy_i\, x_i & \text{otherwise}\end{cases} \quad (2.16)$$

Jika *hinge loss* = 0, maka *W* bernilai tetap, jika *hinge loss* <= 0, maka *W* = *C.y.x* di mana *C = 1/lambda*. Sehingga didapatkan data baru atau transformasi untuk *delta* W_1, *delta* W_2 dan *delta B* seperti ditampilkan pada Tabel 2.11.

Tabel 2.11 Transformasi data baru *delta W* dan *B*

D	Delta (W_1)	Delta (W_2)	Delta (*B*)
D_1	-2	0,7	2
D_2	-2	0,7	2
D_3	-3,2	-0,27	1
D_4	-7,7	-7,1	1
D_5	-2	0,7	2
D_6	-5,7	-3,8	1
D_7	-2	0,7	2
D_8	-2	0,7	2
D_9	-2	0,7	2
D_{10}	-2	0,7	2
Nilai delta	**-3,06**	**-0,627**	**1,7**

Selanjutnya menentukan nilai *learning rate*, asumsikan *lr* = 0,01 maka didapatkan nilai baru dengan persamaan:

$$W' = W - lr * delta(W) \quad (2.17)$$

$\boldsymbol{W_1}'$ = -2 - 0.01 * (-3.06) = -1.9694
$\boldsymbol{W_2}'$ = 0.7 - 0.01 * (-0.627) = 0.70627
$\boldsymbol{B}'$ = 2 - 0.01 * (1.7) = 1.983

Dengan demikian perhitungan di atas dapat dilanjutkan ke iterasi selanjutnya hingga nilai *cost function* semakin rendah. Setelah ditemukan nilai *cost function* terendah, maka nilai weight terendah tersebut merupakan nilai weight optimal, yang akan menjadi model untuk dikalikan dengan setiap nilai atribut.

Apabila hasil dari perkalian tersebut lebih dari > 0 berarti nilainya positif, sedangkan jika hasil perkalian tersebut kurang dari < 0 berarti nilainya negatif.

Secara mendasar persamaan untuk mencari *hyperplane* di atas dapat digunakan untuk menyelesaikan permasalahan klasifikasi secara dua dimensi, seperti menganalisis sentimen yang terkandung dalam teks dokumen kedalam kategori positif atau negatif. Permasalahan dua dimensi tersebut dapat dibagi dengan sebuah *hyperplane linear*. Sementara permasalahan klasifikasi yang bersifat *non linear* memerlukan pemecahan yang lebih kompleks, yaitu dengan memasukkan fungsi *Kernel* pada pemetaan fungsi vektor ke ruang dimensi yang lebih tinggi. Fungsi *Kernel trick* yang umum digunakan pada SVM antara lain *Polynomial*, *Gaussian radial basis function* (*Gaussian* RBF), dan *Sigmoid*. SVM secara komputasional dianggap lebih efisien dan cepat dalam pemrosesan karena hanya menggunakan sebagian dari dokumen yang direpresentasikan sebagai fitur vektor untuk melakukan proses pembelajaran algoritma (Han dan Kamber, 2006).

2.2.5 Pengukuran Kinerja Klasifikasi

Setelah proses data pelatihan selesai, diperlukan pengukuran kinerja untuk menilai tingkat akurasi hasil klasifikasi. Tingkat performansi model klasifikasi berbanding lurus dengan hasil akurasi klasifikasi. Salah satu metode untuk mengukur tingkat akurasi klasifikasi dapat dilakukan dengan metode tabulasi silang atau matriks konfusi (Han dan Kamber, 2006). Matriks konfusi berisi informasi baris matriks dan kolom klasifikasi yang masing-masing merepresentasikan kelas data asli dan kelas data prediksi hasil perhitungan algoritma. Tabel 2.12 menampilkan contoh matriks konfusi hasil klasifikasi atas dua kelas biner yaitu kelas 0 dan kelas 1.

Tabel 2.12 Contoh matriks konfusi 2 kelas

f_{ij}		Kelas Prediksi (*j*)	
		Kelas = 0	Kelas = 1
Kelas Asli (*i*)	Kelas = 0	*True Positive* (TP) → f_{00}	*False Positive* (FP) → f_{01}
	Kelas = 1	*False Negative* (FN) → f_{10}	*True Negative* (TN) → f_{11}

Dalam matriks konfusi setiap sel f_{ij} menyatakan informasi jumlah data dari kelas *i* yang hasil prediksinya masuk ke dalam kelas *j*. Sebagai contoh dihasilkan *true positive* (TP) apabila data kelas asli 0 dipetakan dengan benar ke dalam kelas prediksi 0, sementara dihasilkan *false positive* (FP) apabila data kelas asli 1 dipetakan salah ke dalam kelas prediksi 0. *False negative* (FN) dihasilkan apabila data kelas asli 0 dipetakan salah ke dalam kelas prediksi 1, dan *true negative* (TN) dihasilkan apabila kelas asli 1 benar dipetakan ke dalam kelas prediksi 1. Dari informasi yang dihasilkan oleh matriks konfusi tersebut dapat dicari nilai *Precision*, *Recall*, *Accuracy* dan *F1 Score* untuk menentukan akurasi hasil klasifikasi terbaik.

Precision, merupakan tingkat ketepatan terhadap perkiraan proporsi kasus positif yang benar, dirumuskan dengan persamaan:

$$Precision = \frac{TP}{TP + FP} \tag{2.13}$$

Recall, merupakan tingkat ketepatan terhadap perkiraan proposi kasus positif yang diidentifikasi benar, dirumuskan dengan persamaan:

$$Recall = \frac{TP}{TP + FN} \tag{2.14}$$

Accuracy, merupakan tingkat ketepatan terhadap proporsi jumlah total prediksi yang benar, dirumuskan dengan persamaan:

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{2.15}$$

Pada suatu keadaan dapat ditemukan kondisi ketika terjadi ketidakseimbangan nilai antara *precision*, *recall* dan *accuracy*. Maka diperlukan perhitungan *F1 Score* untuk mencari keseimbangan dengan persamaan:

$$F1\ Score = 2\ \frac{Precision * Recall}{Precision + Recall} \tag{2.16}$$