BAB II
LANDASAN TEORI

2.1 Berita

Berita adalah laporan mengenai fakta atau ide terbaru yang benar, dan penting bagi sebagian besar khalayak, melalui media berkala seperti surat kabar, radio, televisi, atau media *online* internet.

Berita (*news*) adalah laporan mengenai suatu peristiwa atau kejadian yang terbaru (aktual), laporan mengenai fakta-fakta yang aktual, menarik perhatian, dinilai penting, atau luar biasa (Budiman, 2005). Sedangkan menurut (Budyatna, 2005), berita adalah informasi aktual atau terbaru tentang fakta-fakta dan opini yang menarik perhatian orang.

Secara umum, unsure-unsur berita yang selalu ada pada berita adalah : *headline, deteline, lead,* dan *body* (Budiman, 2005).

1. Judul atau kepala berita (*headline*)

   Headline mewakili isi berita yang ingin disampaikan dan memiliki daya tarik yang kuat.

2. Baris tanggal (*dateline*)

   Dateline ada yang terdiri atas nama media massa, tempat kejadian dan tanggal kejadian. Ada pula yang terdiri atas nama media massa, tempat kejadian dan tanggal kejadian.Tujuannya adalah untuk menunjukkan tempat kejadian dan inisial media.

3. Teras berita (*lead* atau *intro*)

   Lead biasanya ditulis pada paragraf pertama sebuah berita. *Lead* merupakan unsur yang paling penting dari sebuah berita, yang menentukan apakah isi berita akan dibaca atau tidak. *Lead* merupakan saripati berita, yang melukiskan seluruh berita secara singkat. *Lead* biasanya berisi hal yang paling penting

dengan mengedepankan unsur 5W + 1H (*what*, *who*, *when*, *where*, *why*, dan *how*)

4. Tubuh berita (*body*)

Body isinya menceritakan peristiwa yang dilaporkan dengan bahasa yang singkat, padat, dan jelas baik yang sudah dikemukakan dalam teras maupun yang belum diungkapkan. Dengan demikian *body* merupakan perkembangan berita.

Bagian-bagian tersebut membentuk anatomi berita yang tersusun sebagai sebuah struktur yang utuh dan terpadu, yang sering dinamakan sebagai gaya piramida terbalik (*inverted pyramid style*). Disebut demikian karena bagian tubuh berita disusun dengan pola pengembangan *umum-khusus* (dimulai dari hal umum, lalu secara berangsur-angsur menuju ke hal-hal yang semakin khusus) atau *klimaks-antiklimaks* (dari yang paling pokok atau penting beralih secara berturut-turut ke yang kurang pokok atau penting). tujuannya adalah untuk memudahkan atau mempercepat pembaca dalam mengetahui apa yang diberitakan; juga untuk memudahkan para redaktur memotong bagian yang tidak atau kurang penting yang terletak di bagian paling bawah dari tubuh berita.

2.2 Metode Klasifikasi Topik Berita

Pendekatan utama dalam menentukan topik berita yaitu, pendekatan supervised learning.

2.2.1 Supervised Learning

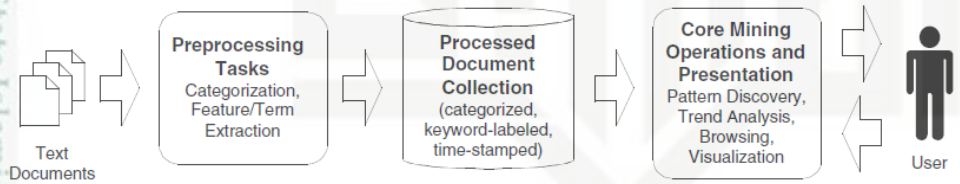
Terdapat empat isu yang harus dipertimbangkan dalam menggunakan teknik *supervised learning* (Feldman, 2007) yaitu perlunya memutuskan kategori yang akan digunakan untuk mengklasifikasikan kasus. Kedua, dibutuhkan satu set pelatihan untuk masing-masing kategori. Ketiga, perlu menentukan fitur dari setiap kategori. Biasanya, lebih baik untuk menghasilkan fitur sebanyak mungkin karena sebagian besar algoritma akan dapat fokus hanya pada fitur yang relevan. Terakhir, perlu memutuskan algoritma yang akan digunakan untuk kategorisasi tersebut.

Beberapa algoritma yang biasa digunakan terhadap pendekatan *supervised learning* (Liu, 2012), diantaranya *naïve bayes*, dan *support vector machines* (SVM). *Supervised learning* bergantung pada data pelatihan. Model klasifikasi berdasarkan data latih yang telah diberi label dalam satu domain, sering berkinerja buruk dengan domain yang berbeda. Meskipun adaptasi domain telah dipelajari oleh para peneliti, namun teknologi ini masih jauh dari sempurna (Liu, 2012).

2.3 Text Mining

Text mining dapat didefinisikan secara luas sebagai proses pengetahuan intensif di mana pengguna berinteraksi dengan koleksi dokumen dari waktu ke waktu dengan menggunakan seperangkat alat analisis. *Text mining* berusaha untuk mengekstrak informasi yang berguna dari sumber data melalui identifikasi dan eksplorasi pola yang menarik. *Text mining* banyak mengarah pada bidang penelitian *data mining*. Oleh karena itu, tidak mengherankan bahwa *text mining* dan *data mining* akan berada pada tingkat arsitektur yang sama (Feldman, 2007)

Berikut gambaran sistem arsitektur *text mining* yang dicantumkan pada buku (Feldman, 2007) Gambar 2.1.



Gambar 2.1 Sistem Arsitektur *Text Mining*

Penelitian di bidang *text mining* menangani masalah yang berkaitan dengan representasi teks, klasifikasi, *clustering*, ekstraksi informasi atau pencarian dan pemodelan pola. Dalam hal ini pemilihan karakteristik, juga domain penelitan dan prosedur penelitian menjadi peran penting. Oleh karena itu, adaptasi dari algoritma *data mining* dari teks yang diketahui sangat diperlukan. Maka dari itu untuk

mencapai hal ini seringkali berdasarkan penelitian sebelumnya *text mining* bergantung pada *information retrieval*, *natural language processing* dan *information extraction*. Selain itu juga penerapan metode *data mining* dan statistik juga diterapkan untuk menangani masalah ini (Hotho, 2005).

Information Retrieval (IR) adalah menemukan bahan (biasanya dokumen) dari suatu keadaan yang tidak terstruktur (biasanya teks) yang memenuhi kebutuhan informasi dari dalam kumpulan data yang besar (biasanya disimpan didalam komputer) (Manning, dkk. 2009). *Natural Language Processing* (NLP) bertujuan untuk mencapai hasil yang lebih baik dalam pemahaman bahasa alami dengan menggunakan komputer. Sedangkan Ekstraksi Informasi (IE). Bertujuan untuk menemukan informasi tertentu dari dokumen teks yang kemudian Ini disimpan dalam basis data seperti pola sehingga dapat digunakan dan dimanfaatkan, juga mengatakan bahwa pada penelitian *text mining* diperlukan tahapan *text preprocessing* pada koleksi dokumen dan menyimpan informasi tersebut dalam struktur data (Hotho, 2005).

Pendekatan *text mining* didasarkan pada pemikiran bahwa dokumen teks dapat diwakili oleh satu *set* kata-kata, yaitu dokumen teks digambarkan berdasarkan pada set kata-kata yang terkandung di dalamnya.

2.3.1 Pembangunan Index

Untuk mendapatkan kata-kata yang digunakan dalam teks tertentu, dibutuhkan proses *tokenization*, yaitu dimana dokumen teks dibagi menjadi aliran kata-kata yang terpisah kemudian dengan menghapus semua tanda baca dan dengan mengganti tab dan karakter non-teks lain dengan spasi tunggal (Hotho, 2005).

Selanjutnya (Hotho, 2005) juga mengatakan untuk dapat mengurangi ukuran koleksi dokumen dapat dilakukannya proses *filtering, lemmatization dan stemming.*

Ketiga tahapan tersebut dapat dijelaskan (Hotho, 2005) sebagaimana berikut:

1. *Filtering* atau *stop-words* yaitu, menghapus kata-kata pada dokumen dimana penyaringan untuk menghapus kata-kata yang mengganggu informasi konten,

seperti konjungsi, dan preposisi. Kata-kata tersebut cenderung tidak memiliki relevansi statistik tertentu dan dapat dihapus dari kamus.

2. *Lemmatization*, yaitu mencoba untuk memberikan pola pada kata kerja dan kata benda tunggal. Namun, untuk mengambarkan hal ini, bentuk kata harus diketahui, yaitu *part of speech* (POS) dari setiap kata dalam dokumen teks harus ditentukan. Karena proses penandaan ini biasanya cukup memakan waktu dan masih rawan kesalahan, dalam penggunaanya metode *stemming* yang diterapkan.
3. Metode *stemming* mencoba untuk membangun bentuk-bentuk dasar dari kata-kata. Dengan cara ini, diperoleh kelompok kata yang mempunyai makna serupa tetapi berbeda wujud sintaktis satu dengan lainnya. Sehingga proses steaming tidak akan merubah makna dari sebuah dokumen. Namun justru meningkatkan relevansi kemiripan dokumen. Ada beberapa algoritma yang dapat digunakan untuk stemming dalam bahasa indonesia, yaitu algoritma Nazief dan Andriani, algoritma Arifin dan Setiono, algoritma Vega dan algoritma Ahmad, Yussof dan Sembok. Algoritma Nazief dan Andriani adalah algoritma yang paling efektif untuk stemming bahasa Indonesia (Agusta, 2009).

Terdapat lima langkah pembangunan inverted index (Bintana, 2012), yaitu:

1. Membangun dokumen yang kemudian akan di-index pada tahapan ini hasil dari kumpulan dokumen sering disebut corpus.
2. Penghapusan format dan markup dari dalam dokumen.

   Pada dokumen yang mempunyai banyak tag markup dan format seperti dokumen (X) HTML semua format *Tag Markup* dihapus.
3. Pemisahan rangkaian kata (*tokenization*).

   Pada tahap ini seluruh kata (term) pada dokumen dipisahkan menjadi potongan kata tunggal. Selanjutnya tahapan ini juga akan menghilangkan karakter-karakter tertentu, yang tidak mewakili atau dapat mengurangi relevansi seperti tanda baca dan mengubah bentuk huruf menjadi kecil.
4. Melakukan *linguistic preprocessing* untuk menghasilkan daftar kata (*term*) yang ternormalisasi. Dua hal yang dilakukan dalam tahap ini adalah:

a. Penyaringan (*filtration*)

Pada tahapan ini ditentukan *term* mana yang akan digunakan untuk merepresentasikan dokumen sehingga dapat mendeskripsikan isi dokumen dan membedakan dokumen tersebut dari dokumen lain di dalam koleksi. Kemudian di indetifikasi *term* yang dianggap tidak berguna. Hal ini memiliki beberapa alasan pertama, dokumen relevan terhadap *query* merupakan bagian kecil dari koleksi dokumen. Sementara itu *term* yang dianggap berguna atau mewakili relevansi *query* dengan koleksi dokumen yaitu kemungkinan besar adalah *term* yang muncul pada sedikit dokumen. Ini berarti bahwa *term* dengan frekuensi kemunculan tinggi bersifat *poor descriminator*. Kedua, *term* yang muncul dalam banyak dokumen tidak mencerminkan definisi dari topik atau sub-topik dokumen. Karena itu, *term* yang sering digunakan dianggap sebagai *stop-words* dan dihapus dari dokumen.

b. Konversi *term* ke bentuk akar (*stemming*)

Stemming adalah proses konversi *term* ke bentuk akarnya.

5. Mengindeks dokumen (*indexing*)

Pengindeksan dilakukan dengan membuat *inverted index* yang terdiri dari *dictionary* dan *postings*. *Inverted index* merupakan konversi dari dokumen asli yang mengandung sekumpulan kata ke dalam daftar kata (*dictionary*) yang memiliki hubungan dengan dokumen terkait dimana kata-kata tersebut muncul (*postings*). *Dictionary* adalah daftar kata yang diperoleh dari hasil pengindeksan koleksi dokumen.

2.3.2 Algoritma Nazief Adriani

Algoritma Nazief dan Adriani, algoritma *stemming* untuk teks berbahasa Indonesia yang mempunyai tingkat keakuratan yang lebih baik dari algoritma lainnya (Agusta, 2009). Algoritma Nazief dan Adriani mengacu pada aturan morfologi bahasa Indonesia yang mengelompokkan imbuhan, yaitu imbuhan yang diperbolehkan atau imbuhan yang tidak diperbolehkan. Pengelompokan ini termasuk

imbuhan di depan (awalan), imbuhan kata di belakang (akhiran), imbuhan kata di tengah (sisipan) dan kombinasi imbuhan pada awal dan akhir kata (konfiks) (Sahroni, R, 2012).

Berikut ini adalah langkah-langkah yang dilakukan oleh algoritma Nazief dan Adriani (Agusta, 2009) :

1. Kata yang belum di-*stemming* dicari pada kamus KBBI. Jika kata itu langsung ditemukan, berarti kata tersebut adalah kata dasar. Kata tersebut dikembalikan dan algoritma dihentikan.
2. Hilangkan *inflectional suffixes* terlebih dahulu. Jika hal ini berhasil dan *suffix* adalah partikel (“lah” atau ”kah”), langkah ini dilakukan lagi untuk menghilangkan *inflectional possessive pronoun suffixes* (“ku”, “mu” atau ”nya”).
3. *Derivational suffix* kemudian dihilangkan. Lalu langkah ini dilanjutkan lagi untuk mengecek apakah masih ada *derivational suffix* yang tersisa, jika ada maka dihilangkan. Jika tidak ada lagi maka lakukan langkah selanjutnya.
4. Kemudian *derivational prefix* dihilangkan. Lalu langkah ini dilanjutkan lagi untuk mengecek apakah masih ada *derivational prefix* yang tersisa, jika ada maka dihilangkan. Jika tidak ada lagi maka lakukan langkah selanjutnya.
5. Setelah tidak ada lagi imbuhan yang tersisa, maka algoritma ini dihentikan kemudian kata dasar tersebut dicari pada kamus, jika kata dasar tersebut ketemu berarti algoritma ini berhasil tapi jika kata dasar tersebut tidak ketemu pada kamus, maka dilakukan *recoding*.
6. Jika semua langkah telah dilakukan tetapi kata dasar tersebut tidak ditemukan pada kamus juga maka algoritma ini mengembalikan kata yang asli sebelum dilakukan *stemming*.

Kelebihan pada algoritma Nazief dan Andriani ini adalah bahwa algoritma ini memperhatikan kemungkinan adanya partikel-partikel yang mungkin mengikuti suatu kata berimbuhan. Sehingga dapat melihat pada rumus untuk algoritma ini yaitu

adanya penempatan *possesive pronoun* dan juga partikel yang mungkin ada pada suatu kata berimbuhan. Akhir dari algoritma ini yaitu apabila pemotongan semua imbuhan telah berhasil dan hasil pemotongan imbuhan tersebut terdapat pada kamus maka algoritma ini dapat dikatakan berhasil dalam penentuan kata dasarnya. Dan apabila sebaliknya bahwa algoritma ini setelah dilakukan pemotongan kata dan tidak terdapat pada kamus maka kata berimbuhan yang telah mengalami pemotongan dikembalikan ke keadaan semula.

Algoritma yang dibuat oleh Bobby Nazief dan Mirna Adriani ini memiliki tahap-tahap sebagai berikut :

1. Cari kata yang akan di*stemming* dalam kamus. Jika ditemukan maka diasumsikan bahwa kata tesebut adalah *root word*, maka algoritma berhenti.
2. *Inflection suffixes* (“-lah”, “-kah”, “-ku”, “-mu”, atau “-nya”) dibuang. Jika berupa partikel (“-lah”, “-kah”, “-tah” atau “-pun”) maka langkah ini diulangi lagi untuk menghapus *possesive pronouns* (“-ku”, “-mu”, atau “-nya”), jika ada.
3. Hapus *Derivation suffixes* (“-i”, “-an” atau “-kan”). Jika kata ditemukan di kamus, maka algoritma berhenti. Jika tidak maka ke langkah 3a
   a. Jika “-an” telah dihapus dan huruf terakhir dari kata tersebut adalah “-k”, maka “-k” juga ikut dihapus. Jika kata tersebut ditemukan dalam kamus maka algoritma berhenti. Jika tidak ditemukan maka lakukan langkah 3b.
   b. Akhiran yang dihapus (“-i”, “-an” atau “-kan”) dikembalikan, lanjut ke langkah 4.
4. Hapus *Derivation prefix*. Jika pada langkah 3 ada *sufiks* yang dihapus maka pergi ke langkah 4a, jika tidak pergi ke langkah 4b.

a. Periksa kombinasi awalan-akhiran yang tidak diijinkan. Jika ditemukan maka algoritma berhenti, jika tidak pergi ke langkah 4b.
b. For i = 1 to 3, tentukan tipe awalan kemudian hapus awalan. Jika *root word* belum juga ditemukan lakukan langkah 5. Jika sudah maka algoritma berhenti. Catatan: jika awalan kedua sama dengan awalan pertama maka algoritma berhenti.

5. Melakukan *recoding*.
6. Jika semua langkah telah selesai tetapi tidak juga berhasil maka kata awal diasumsikan sebagai *root word*. Proses selesai.

Tipe awalan ditentukan melalui langkah-langkah berikut:

a. Jika awalannya adalah: “di-”, “ke-”, atau “se-” maka tipe awalannya secara berturut-turut adalah “di-”, “ke-”, atau “se-”.
b. Jika awalannya adalah “te-”, “me-”, “be-”, atau “pe-” maka dibutuhkan sebuah proses tambahan untuk menentukantipe awalannya.
c. Jika dua karakter pertama bukan “di-”, “ke-”, “se-”, “te-”, “be-”, “me-”, atau “pe-” maka berhenti.
d. Jika tipe awalan adalah “tidak ada” maka berhenti. Jika tipe awalan adalah bukan “tidak ada” . Hapus awalan jika ditemukan.

2.3.3 Pembobotan Kata

Setiap *term* yang telah di-*index* diberikan bobot sesuai dengan struktur pembobotan yang dipilih, apakah pembobotan lokal, global atau kombinasi keduanya. Jika menggunakan pembobotan lokal maka, pembobotan *term* diekspresikan sebagai *tf* (*term frequency*). Namun, jika pembobotan global yang digunakan maka, pembobotan *term* didapatkan melalui nilai *idf* (*inverse document frequency*). Beberapa aplikasi juga ada yang menerapkan pembobotan kombinasi keduanya yaitu, dengan mengalikan bobot lokal dan global (*tf* . *idf*) (Bintana, 2012).

1. *Term Frequency*

   Empat cara yang dapat digunakan untuk memperoleh nilai *term frequency* (*tf*), yaitu:

   a. *Raw term frequency*. Nilai *tf* sebuah *term* diperoleh berdasarkan jumlah kemunculan *term* tersebut dalam dokumen. Contoh kasus dimana *term* muncul sebanyak dua kali dalam suatu dokumen maka, nilai *tf term* tersebut adalah 2.

   b. *Logarithm term frequency*. Hal ini untuk menghindari dominasi dokumen yang mengandung sedikit *term* dalam *query*, namun mempunyai frekuensi yang tinggi. Cara ini menggunakan fungsi logaritmik matematika untuk memperoleh nilai *tf*.

      $tf = 1 + \log(tf)$ .....(2.1)

   c. *Binary term frequency*. Hanya memperhatikan apakah suatu *term* ada atau tidak dalam dokumen. Jika ada, maka *tf* diberi nilai 1, jika tidak ada diberi nilai 0. Pada cara ini jumlah kemunculan *term* dalam dokumen tidak berpengaruh.

   d. *Augmented Term Frequency*. Nilai *tf* adalah jumlah kemunculan suatu *term* pada sebuah dokumen, sedangkan nilai *max(tf)* adalah jumlah kemunculan terbanyak sebuah *term*pada dokumen yang sama.

      $tf = 0.5 + 0.5 \times \frac{tf}{\max(tf)}$ .....(2.2)

2. *Inverse Document Frequency*

   Inverse document frequency (*idf*) digunakan untuk memberikan tekanan terhadap dominasi *term* yang sering muncul di berbagai dokumen. Hal ini diperlukan karena *term* yang banyak muncul di berbagai dokumen, dapat dianggap sebagai *term* umum (*common term*) sehingga tidak penting nilainya. Pembobotan akan memperhitungkan faktor kebalikan frekuensi dokumen yang mengandung suatu *term* (*inverse document frequency*).

$$idf(t) = \log\left(\frac{N}{df(t)}\right) \quad \text{.....(2.3)}$$

Keterangan:

N : jumlah dokumen dalam *corpus*.

df(t) : *document frequency* atau jumlah dokumen dalam *corpus* yang mengandung *term t*.

2.4 Metode *Naive Bayes*

Untuk menjelaskan klasifikasi Naive Bayes, dimisalkan (Kabir, et al, 2011 dikutip oleh Ulysses, 2013) : sebuah kasus dalam suatu database X ={$x_1, x_2, \ldots, x_n$}, yang memiliki nilai fitur data pada sebuah kumpulan n atribut. Dengan menjadikan H sebagai hipotesis, sehingga data X menjadi sebuah kelas spesifik Ci dimana, $H = X \in C_i$. Dalam klasifikasi Naive Bayes, mengkalkulasi sample X adalah bagian Kelas Ci, dengan memberikan nilai fitur dari X. Dengan teorema Bayes bisa dituliskan :

$$P(H|X) = \frac{P(H|X).P(H)}{P(X)} \quad \text{.....(2.4)}$$

Keterangan :

X : Data dengan kelas yang belum diketahui

H : Hipotesis data X merupakan suatu kelas spesifik

P(H|X) : Probabilitas hipotesis H berdasarkan kondisi X (*posteriori probability*)

P(H) : Probabilitas hipotesis H (*prior probability*)

P(X|H) : Probabilitas X berdasarkan kondisi pada hipotesis H

P(X) : Probabilitas X

Untuk Menjelaskan teorema Naive Bayes, Perlu diketahui bahwa proses klasifikasi memerlukan sejumlah petunjuk untuk menentukan kelas apa yang cocok bagi sampel yang dianalisa tersebut. Karena itu, teorema bayes diatas disesuaikan sebagai berikut :

$$P\,(C|F_{i....}F_n) = \frac{P\,(C)P(F_1 \ldots\ldots F_n|C)}{P\,(F_1 \ldots . F_n)}$$

Dimana Variabel *C* merepresentasikan kelas, sementara variabel *F1 ... Fn* merepresentasikan karakteristik petunjuk yang dibutuhkan untuk melakukan klasifikasi. Maka rumus tersebut menjelaskan bahwa peluang masuknya sampel karakteristik tertentu dalam kelas *C* (*Posterior*) adalah peluang munculnya kelas *C* (sebelum masuknya sampel tersebut, seringkali disebut *prior*), dikali dengan peluang kemunculan karakteristik karakteristik sampel pada kelas *C* (disebut juga *likelihood*), dibagi dengan peluang kemunculan karakteristik karakteristik sampel secara global (disebut juga *evidence*). Karena itu, rumus diatas dapat pula ditulis secara sederhana sebagai berikut :

$$Posterior = \frac{Prior\ x\ likelihood}{evedence}$$

Nilai *Evidence* selalu tetap untuk setiap kelas pada satu sampel. Nilai dari *posterior* tersebut nantinya akan dibandingkan dengan nilai nilai *posterior* kelas lainnya untuk menentukan ke kelas apa suatu sampel akan diklasifikasikan. Penjabaran lebih lanjut rumus *Bayes* tersebut dilakukan dengan menjabarkan $(C|F_{1,....}F_n)$ menggunakan aturan perkalian sebagai berikut :

$$P\ (\ C\ |F_1, \ldots F_n) = P(C)P\ (F_1, .., F_n|C)$$

$$= P(C)P(F_1|C)P(F_2, \ldots, F_n|C, F_1)$$

$$= P(C)P(F_1|C)P(F_2|C, F_1)P(F_3, \ldots, F_n|C, F_1, F_2)$$

$$= P(C)P(F_1|C)P(F_2|C,F_1)P(F_3|C,F_1,F_2),P(F_4,\dots,F_n|C,F_1,F_2,F_3)$$

$$= P(C)P(F_1|C)P(F_2|C,F_1)P(F_3|C,F_1,F_2)\dots P\ (F_n|C,F_1,F_2,F_3,\dots,F_{n-1})$$

Dapat dilihat bahwa hasil penjabaran tersebut menyebabkan semakin banyak dan semakin kompleksnya faktor-faktor syarat yang mempengaruhi nilai probabilitas, yang hampir mustahil untuk dianalisa satu persatu. Akibatnya, perhitungan tersebut menjadi sulit untuk dilakukan. Disinilah digunakan asumsi independensi yang sangat tinggi (*naif*), bahwa masing masing petunjuk $(F_1,F_2,\dots\ F_n)$ saling bebas *(independen)* satu sama lain. Dengan asumsi tersebut, maka berlaku suatu kesamaan sebagai berikut:

$$P\ (P_i|F_j) = \frac{P(F_1 \cap F_j)}{P\ (F_j)} = \frac{P(F_i \cap F_j)}{P\ (F_j)} = P\ (f_j)$$

Untuk $i\ \neq j$, sehingga

$$P(F_i|C,F_j) = P\ (F_i|C)$$

Dari persamaan diatas dapat disimpulkan bahwa asumsi independensi naif tersebut membuat syarat peluang menjadi sederhana, sehingga perhitungan menjadi mungkin untuk dilakukan. Selanjutnya, penjabaran $P(C|F_1,\dots,F_n)$ dapat disederhanakan menjadi :

$$P\ (C|F_1,\dots,F_n) = P\ (C)P(F_1|C)P(F_2|C)P(F_3|C)\dots$$

$$P\ (C)\prod_{i}^{n} = 1\ P(F_i|C)$$

Persamaan diatas merupakan model dari teorema *Naive Bayes* yang selanjutnya akan digunakan dalam proses klasifikasi.

2.4.1 K-fold cross validation

K-*fold cross validation* adalah teknik yang dapat digunakan jika memiliki jumlah data yang terbatas (jumlah *instance* tidak banyak). Cara kerja K-*fold cross validation* adalah sebagai berikut:

1. Seluruh data dibagi menjadi K bagian.
2. *Fold* ke-1 adalah ketika bagian ke-1 menjadi data uji (*testing data*) dan sisanya menjadi data latih (*training data*). Selanjutnya, hitung akurasi berdasarkan porsi data tersebut.
3. *Fold* ke-2 adalah ketika bagian ke-2 menjadi data uji (*testing data*) dan sisanya menjadi data latih (*training data*). Selanjutnya, hitung akurasi berdasarkan porsi data tersebut.
4. Demikian seterusnya hingga mencapai *fold* ke-K.
5. Hitung rata-rata akurasi dari N buah akurasi di atas. Rata-rata akurasi ini menjadi akurasi final.

Metode k-*fold cross validation* melakukan generalisasi dengan membagi data ke dalam k bagian berukuran sama. Selama proses berlangsung, salah satu dari partisi dipilih untuk data uji, dan sisanya digunakan untuk dta latih. Langkah ini diulangi k kali sehingga setiap partisi digunakan untuk data uji tepat satu kali. Metode k-*fold cross validation* menetapkan k = N, ukuran dari data set.

Pendekatan ini memiliki keuntungan dalam pengunaan data sebanyak mungkin untuk *training*. *Test set* secara efektif mencakup keseluruhan data set. Kekurangan dari pendekatan ini adalah banyaknya komputasi untuk mengulangi prosedur sebanyak N kali. K-*fold cross validation* adalah salah satu teknik untuk mengevaluasi keakuratan model (Mustika 2015).

2.4.2 Confusion Matrix

Confusion matrix merupakan alat yang berguna untuk menganalisis seberapa baik *classifier* mengenali *tuple* dari kelas yang berbeda. TP dan TN memberikan

informasi ketika *classifier* benar, sedangkan FP dan FN memberikan informasi ketika *classifier* salah (Han, dkk, 2012). Gambar 2.2 adalah contoh dari *confusion matrix*.

		Actual Class	
		Ya	Tidak
Predictive Class	Ya	TP	FN
	Tidak	FP	TN
	Total	P'	N'

Gambar 2.2 *Confusion Matrix* (Han et al., 2012)

Akurasi merupakan persentase dari data yang diprediksi secara benar. Perhitungan akurasi adalah :

$$Akurasi = \frac{(TP+TN)}{(TP+TN+FP+FN)} \quad (2.5)$$

Keterangan :

TP : *True positives*, merupakan jumlah data dengan kelas positif yang diklasifikasikan positif.

TN : *True negatives*, merupakan jumlah data dengan kelas negatif yang diklasifikasikan negatif.

FP : *False positives*, merupakan jumlah data dengan kelas positif diklasifikasikan negatif.

FN : *False negatives*, merupakan jumlah data dengan kelas negatif diklasifikasikan positif