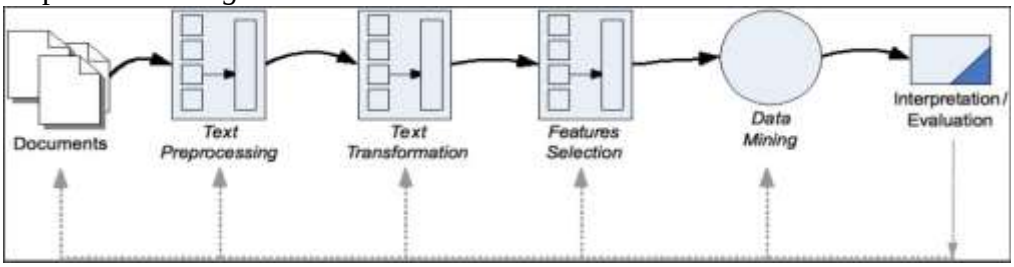
BAB II
LANDASAN TEORI

2.1 Text Mining

Text mining memiliki definisi menambang data yang berupa teks dimana sumber data biasanya didapatkan dari dokumen, dan tujuannya adalah mencari kata-kata yang dapat mewakili isi dari dokumen sehingga dapat dilakukan analisa keterhubungan antar dokumen. Jenis masukan untuk penambangan teks ini disebut data tak terstruktur dan merupakan pembeda utama dengan penambangan data yang menggunakan data terstruktur atau basis data sebagai masukan. Penambangan teks dapat dianggap sebagai proses dua tahap yang diawali dengan penerapan struktur terhadap sumber data teks dan dilanjutkan dengan ekstraksi informasi dan pengetahuan yang relevan dari data teks terstruktur ini dengan menggunakan teknik dan alat yang sama dengan penambangan data. Proses yang umum dilakukan oleh penambangan teks diantaranya adalah perangkuman otomatis, kategorisasi dokumen, penggugusan teks, dan lain-lain. Tujuan dari *text mining* adalah untuk mendapatkan informasi yang berguna dari sekumpulan dokumen. Jadi, sumber data yang digunakan pada *text mining* adalah kumpulan teks yang memiliki format yang tidak terstruktur atau minimal semi terstruktur. Adapun tugas khusus dari *text mining* antara lain yaitu pengkategorisasian teks (*text categorization*) dan pengelompokan teks (*text clustering*). *Text mining* merupakan penerapan konsep dan teknik *data mining* untuk mencari pola dalam teks, yaitu proses penganalisisan teks guna menyarikan informasi yang bermanfaat untuk tujuan tertentu. Berdasarkan ketidakteraturan struktur data teks, maka proses *text mining* memerlukan beberapa tahap awal yang pada intinya adalah mempersiapkan agar teks dapat diubah menjadi lebih terstruktur [7].

Tahapan *text mining*



Gambar 1.1 Proses *Text Mining [7]*

Area penerapan *text mining* yang paling populer adalah:

1. Ekstraksi informasi (*information extraction*): Identifikasi frasa kunci dan keterkaitan di dalam teks dengan melihat urutan tertentu melalui pencocokan pola.
2. Pelacakan topik (*topic tracking*): Penentuan dokumen lain yang menarik seorang pengguna berdasarkan profil dan dokumen yang dilihat pengguna tersebut.
3. Perangkuman (*summarization*): Pembuatan rangkuman dokumen untuk mengefisiensikan proses membaca.
4. Kategorisasi (*categorization*): Penentuan tema utama suatu teks dan pengelompokan teks berdasarkan tema tersebut ke dalam kategori yang telah ditentukan.
5. Penggugusan (*clustering*): Pengelompokan dokumen yang serupa tanpa penentuan kategori sebelumnya.
6. Penautan konsep (*concept linking*): Penautan dokumen terkait dengan identifikasi konsep yang dimiliki bersama sehingga mambantu pengguna untuk menemukan informasi yang mungkin tidak akan ditemukan dengan hanya menggunakan metode pencarian tradisional.
7. Penjawaban pertanyaan (*question answering*): Pemberian jawaban terbaik terhadap suatu pertanyaan dengan pencocokan pola berdasarkan pengetahuan [8].

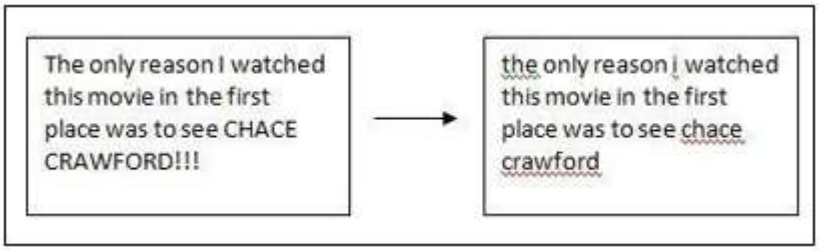
2.1.1 *Text Preprocessing*

Struktur data yang baik dapat memudahkan proses komputerisasi secara otomatis. Pada *Text Mining*, informasi yang akan digali berisi informasi-informasi yang strukturnya sembarang. Oleh karena itu, diperlukan proses pengubahan bentuk menjadi data yang terstruktur sesuai kebutuhannya untuk proses dalam *data mining*, yang biasanya akan menjadi nilai - nilai numerik. Proses ini sering disebut *Text Preprocessing*. Setelah data menjadi data terstruktur dan berupa nilai numerik maka

data dapat dijadikan sebagai sumber data yang dapat diolah lebih lanjut [2]. Berberapa proses yang dilakukan adalah sebagai berikut:

1. *Case Folding*

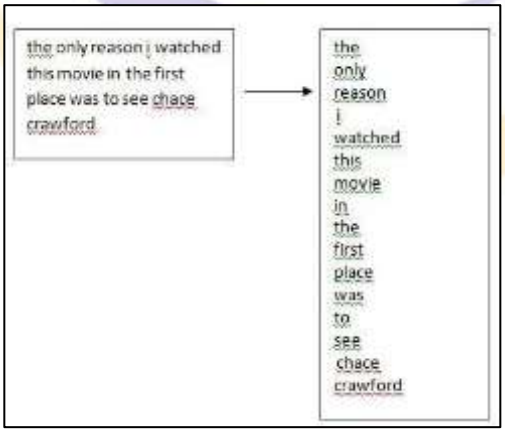
   Case folding adalah mengubah semua huruf dalam dokumen menjadi huruf kecil. Hanya huruf 'a' sampai dengan 'z' yang diterima. Karakter selain huruf dihilangkan dan dianggap delimiter [2].



Gambar 1.2 Proses Case Folding [8]

2. *Tokenizing*

   Tahap *Tokenizing* adalah tahap pemotongan *string input* berdasarkan tiap kata yang menyusunnya [2].

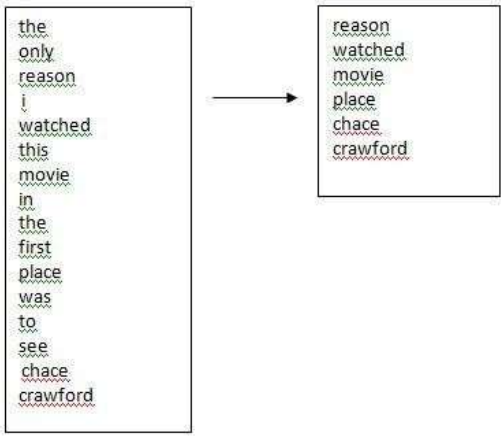


Gambar 1.3 Proses Tokenizing [8]

2.1.2 *Text Transformation*

1. *Stopwordremoval* atau *filering*

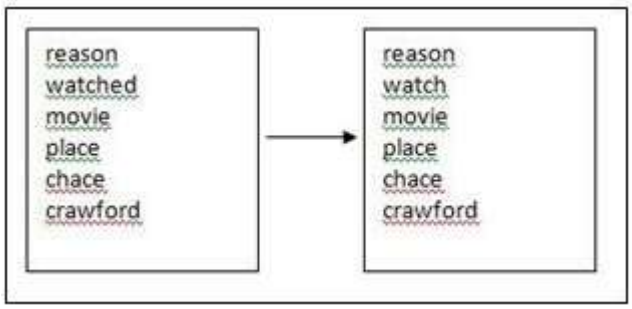
   Tahap *filtering* adalah tahap mengambil kata - kata penting dari hasil token. Bisa menggunakan algoritma *stoplist* (membuang kata yang kurang penting) atau *wordlist* (menyimpan kata penting). *Stoplist* / *stopword* adalah kata-kata yang tidak deskriptif yang dapat dibuang dalam pendekatan *bag-of-words* [2].



Gambar 1.4 Proses Stopwordremoval atau filtering [8]

2. *Stemming*

   Tahap *stemming* adalah tahap mencari *root* kata dari tiap kata hasil *filtering*. Pada tahap ini dilakukan proses pengembalian berbagai bentukan kata ke dalam suatu representasi yang sama [2].



Gambar 1.5 Proses Stemming [8]

2.2 Aturan Peluruhan Kata Dasar

Ada beberapa kata dasar yang apabila dilekati oleh awalan "me(N)-", "pe(N)-", "pe(R)-", "te(R)-", "be(R)-," akan mengelami peluruhan atau perubahan pada karakter awal dari kata dasar tersebut. Sebagai contoh kata "tanya", karakter awal dari kata "tanya" akan berubah apabila ditambahkan awalan "me-"dan menjadi "menanya". Begitu juga untuk beberapa kata dasar lainnya. Untuk melakukan proses *stemming* pada kata-kata tersebut harus mengikuti aturan peluruhan yang telah ditetapkan oleh algoritma [9]. Aturan- aturan tersebut dijelaskan pada gambar berikut.

Aturan	Bentuk Awalan	Peluruhan
1	berV…	ber-V… | be-rV…
2	belajar…	bel-ajar
3	beC_1erC_2…	be-C_1erC_2…dimana C_1!={'r'|'l'}
4	terV…	ter-V…|te-rV…
5	terCer…	ter-Cer…dimana C!='r'
6	teC_1erC_2	te-C_1erC_2…dimana C_1!='r'
7	me{l|r|w|y}V…	me-{l|r|w|y}V…
8	mem{b|f|v}…	mem-{b|f|v}…
9	mempe…	mem-pe…
10	mem{rV|V}…	me-m{rV|V}…|me-p{rV|V}…

Gambar 1.6 Aturan peluruhan kata dasar [8]

2.3 Algoritma *Stemming* Nazief-Adriani

Algoritma ini berdasarkan pada aturan morfologi bahasa Indonesia yang luas dan dikumpulkan menjadi satu grup serta dienkapsulasi pada imbuhan yang diperbolehkan dan imbuhan yang tidak diperbolehkan. Algoritma ini memiliki hasil kebenaran sekitar 93%. Langkah-langkah algoritma Nazief-Adriani:

1. Kata yang belum di *stemming* dicari pada kamus. Jika ditemukan berarti kata tersebut merupakan kata dasar sehingga kata tersebut dikembalikan dan algoritma dihentikan.
2. Hilangkan *Inflectional suffixes* terlebih dahulu. Jika berhasil dan *suffix* adalah partikel ("lah" atau "kah"), langkah ini dilakukan lagi untuk menghilangkan *Inflectional possessive pronoun suffixes* ("ku", "mu" atau "nya").
3. *Derivational prefix* kemudian dihilangkan. Langkah dilanjutkan untuk mengecek apakah masih ada *Derivational prefix* yang tersisa, jika ada maka dihilangkan. Jika tidak ada maka lakukan langkah selanjutnya.
4. Setelah tidak ada lagi imbuhan yang tersisa, algoritma dihentikan dan kata dasar dicari pada kamus, jika kata dasar tersebut ditemukan berarti algoritma ini berhasil tetapi jika tidak ketemu, maka dilakukan recoding.
5. Jika semua langkah telah dilakukan tetapi kata dasar tersebut tidak ditemukan pada kamus juga maka algoritma ini mengembalikan kata yang asli sebelum dilakukan *stemming* [10].

Berdasarkan penelitian sebelumnya tentang *stemming* yang membandingkan delapan algoritma *stemming* bahasa Indonesia yang meliputi algoritma Nazief Adriani, algoritma Yussof & Sembok, algoritma Idris & Mustofa, algoritma Vega, algoritma Arifin & Setiono, algoritma Porter Bahasa Indonesia, algoritma *Confix Stripping*, dan algoritma *Enhanced Confix Stripping*, didapatkan hasil bahwa algoritma Nazief Adriani memiliki *Word Conflation Class* (WCC) dan *Index Compression Factor* (ICF) lebih tinggi dibandingkan tujuh algoritma *stemming* lainnya. Berikut hasil WCC dan ICF [11].

Tabel 1.1 Tabel hasil WCC dan ICF [11]

Nama algoritma	Jumlah *term* unik (*distinct*) sebelum *stemming* & *stopword*	Jumlah *term* unik (*distinct*) setelah *stemming* & *stopword*	*Stemmer Strength*	
			avg word per *conflation class* (wc)	*index compression* (icf)
Nazief & Andriani	13515	8665	1.56	35.89
Confix		8710	1.55	35.55
ECS		8817	1.53	34.76
Arifin & Setiono		8992	1.50	34.47
Idris & Mustofa		9031	1.50	33.18
Yusof & Sembok		9196	1.47	31.96
Porter		9262	1.46	31.47
Vega		9603	1.41	28.95

Dari Tabel 2.1, dapat dilihat bahwa nilai WCC dan ICF yang dihasilkan algoritma Nazief Andriani memiliki nilai lebih tinggi dibandingkan dengan algoritma Confix, algoritma ECS, algoritma Arifin & Mustofa, algoritma Idris & Mustofa, algoritma Yusof & Sembok, algoritma Idris & Mustofa, algoritma Porter, dan algoritma Vega. Semakin tinggi nilai WCC dan ICF yang dihasilkan maka, algoritma *stemming* semakin baik, dan begitupun sebaliknya. Nilai WCC dan ICF tersebut dipengaruhi oleh beberapa alasan:

1. Nilai WCC dan ICF pada *stemming* Nazief Andriani yang lebih tinggi menunjukkan bahwa algoritma Nazief & Andriani lebih agresif dalam mereduksi kelompok kata. Misalkan kelompok kata "menyayangi", "disayangi", dan "penyayang" akan dianggap sebagai satu kata setelah dilakukan proses *stemming*, karena ketiganya memiliki kata dasar yang sama, yaitu "sayang". Semakin banyak kelompok kata yang dapat direduksi maka akan berpengaruh pada jumlah *term* yang dihasilkan oleh masing-masing algoritma.
2. Perbedaan pada jumlah *term* yang dihasilkan. Jumlah *term* sebelum di-*stemming* & *stopword* adalah 13.515 *term*. Kemudian setelah dilakukan proses *stemming* dengan algoritma Nazief & Andriani menghasilkan *term* sebanyak 8.665, algoritma

Confix menghasilkan 8.710 *term,* algoritma ECS menghasilkan 8.817 *term*, algoritma Arifin & Setiono menghasilkan 8.992 *term*, algoritma Idris & Mustofa menghasilkan 9.031 *term*, algoritma Yusof & Sembok menghasilkan 9.196 *term*, algoritma Porter menghasilkan 9.262 *term*, dan algoritma Vega menghasilkan 9.603 *term*. Dari hasil pengujian pada tabel 2.1 dapat disimpulkan bahwa semakin sedikit jumlah kata unik (*distinct*) setelah *stemming* maka semakin baik pula nilai WCC dan ICF yang dihasilkan, atau dengan kata lain semakin sedikit jumlah kata unik (*distinct*) setelah proses *stemming*, maka semakin banyak kelompok kata yang direduksi [11].

Berikut ini hasil performansi klasifikasi menggunakan *Support Vector Machine Classifier* (SVM) yang dilakukan dengan menggunakan algoritma *stemming* yang berbeda.

Tabel 1.2 Tabel hasil performansi [11]

Nama algoritma	Kategori				Ratarata
	Edukasi	Hukum	Olahraga	Politik	
Nazief & Andriani	0.922	0.847	0.968	0.598	0.856
Confix	0.912	0.843	0.967	0.585	0.85
ECS	0.912	0.84	0.965	0.565	0.845
Arifin & Setiono	0.912	0.835	0.962	0.544	0.838
Idris & Mustofa	0.912	0.828	0.96	0.537	0.834
Yusof & Sembok	0.912	0.826	0.96	0.492	0.826
Porter	0.912	0.815	0.952	0.469	0.815
Vega	0.912	0.812	0.951	0.446	0.809

Berdasarkan tabel 2.2, terlihat dengan jelas bahwa nilai rata-rata *F1-Measure stemming* Nazief Adriani berhasil mengungguli nilai rata-rata *F1-Measure* dari metode-metode lainnya. Nilai rata-rata *F1-Measure stemming* Nazief Adriani sebesar 0.856 dari skala 0-1 yang merupakan nilai tertinggi diantara metode lainnya. Tingginya nilai rata-rata *F1-Measure* pada *stemming* Nazief Adriani mengidintifikasikan bahwa rasio perbandingan antara jumlah atau total prediksi dokumen uji dari masing-masing kategori dapat diklasifikasikan dengan baik sesuai dengan kategorinya.

Berdasarkan hasil dari percobaan yang telah dilakukan, besar atau kecilnya nilai *F1Measure* dipengaruhi oleh beberapa faktor. Antara lain adalah *classifier* dan algoritma *stemming* yang digunakan. Nilai *F1-Measure* akan bagus jika *classifier*-nya juga bagus (mampu menebak atau memprediksi dengan baik). Bagus atau tidaknya tebakan/prediksi suatu *classifier* akan sangat bergantung pada jenis algoritma *stemming* yang digunakan. Semakin baik algoritma *stemming* yang digunakan, maka semakin baik pula hasil tebakan/prediksi suatu *classifier*. Sehingga dapat disimpulkan bahwa proses *stemming* memiliki peranan yang sangat penting dalam proses klasifikasi dokumen [11].

2.4 Pembobotan TF-IDF

Tahapan ini merupakan tahapan penting dalam *text mining* khususnya dalam proses klasifikasi menggunakan metode *K-Nearest Neighbor*. Salah satu fungsi penting yang disediakan adalah untuk dapat memilih term atau kata apa saja yang dapat dijadikan sebagai wakil penting untuk kumpulan dokumen yang kita analisis dengan kata lain melakukan pembobotan terhadap setiap term. Pembobotan yang paling umum digunakan dalam *text mining* adalah *Term Frequency-Inverse Document Frequency* (TF-IDF) [8].

Pembobotan TF-IDF adalah jenis pembobotan yang sering digunakan dalam *information retrieval* dan *text mining*. Pembobotan ini adalah suatu pengukuran statistik untuk mengukur seberapa penting sebuah kata dalam kumpulan dokumen. Tingkat kepentingan meningkat ketika sebuah kata muncul beberapa kali dalam sebuah dokumen tetapi diimbangi dengan frekuensi kemunculan kata tersebut dalam sebuah dokumen. TF-IDF dapat dirumuskan sebagai berikut:

$$TF\ IDF\ (t_k, d_j) = TF(t_k, d_j)\ *\ IDF(d_j) \quad (1)$$

Dimana sebelumnya dihitung terlebih dahulu *Term Frequency* (TF) yaitu frekuensi kemunculan suatu term di tiap dokumen. Kemudian dihitung *Inverse Document Frequency* (IDF) yaitu nilai bobot suatu *term* dihitung dari seringnya suatu

term muncul dibeberapa dokumen. Semakin sering suatu term muncul di banyak dokumen, maka nilai IDF nya akan kecil. Berikut rumus TF dan IDF [8].

$$TF\ (t_k, d_j) = f(t_k, d_j) \quad (2)$$

$$IDF\ (t_k) = log \frac{N}{df(t)} \quad (3)$$

2.5 Klasifikasi

Klasifikasi merupakan suatu pekerjaan menilai objek data untuk memasukkannya ke dalam kelas tertentu dari sejumlah kelas yang tersedia. Dalam klasifikasi ada dua pekerjaan utama yang dilakukan, yaitu: pertama, pembangunan model sebagai *prototype* untuk disimpan sebagai memori dan kedua, penggunaan model tersebut untuk melakukan pengenalan / klasifikasi / prediksi pada suatu objek data lain agar diketahui di kelas mana objek data tersebut dalam model yang mudah disimpan [2].

Contoh aplikasi yang sering ditemui adalah pengklasifikasian jenis hewan, yang mempunyai sejumlah atribut. Dengan atribut tersebut, jika ada hewan baru, kelas hewannya bisa langsung diketahui. Contoh lain adalah bagaimana melakukan diagnosis penyakit kulit kanker melanoma, yaitu dengan melakukan pembangunan model berdasarkan data latih yang ada, kemudian menggunakan model tersebut untuk mengidentifikasi penyakit pasien baru sehingga diketahui apakah pasien tersebut menderita kanker atau tidak [1].

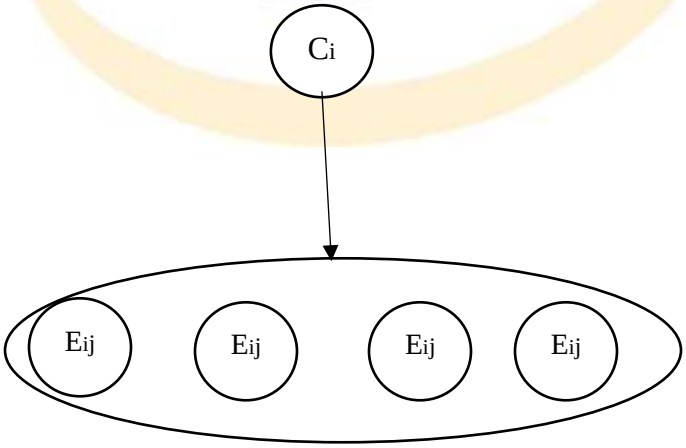
2.6 Naïve Bayes

Naïve Bayes merupakan suatu metode klasifikasi yang menggunakan perhitungan probabilitas. Konsep dasar yang digunakana pada *Naïve Bayes Classifier* adalah teorema bayes yang dinyatakan pertama kali oleh Thomas Bayes [5]. Metode klasifikasi *Naïve Bayes* adalah metode pembelajaran Bayesian yang ditemukan sangat berguna dalam berbagai aplikasi. *Naïve Bayes* merupakan salah satu metode *supervised document classification.* Metode ini sering digunakan dalam menyelesaikan masalah

dalam bidang *machine learning* karena metode ini dikenal memiliki tingkat akurasi yang tinggi dengan perhitungan sederhana serta proses pembelajaran yang cukup cepat [12]. Metode ini disebut *naïve* karena menggabungkan asumsi yang menyederhanakan bahwa nilai setiap atribut adalah saling bebas bersyarat atau *conditionally independent* terhadap kelasnya. Ketika asumsi ini terpenuhi, *Naïve Bayes* menghasilkan klasifikasi *maximum a posteriori* (MAP). Bahkan ketika asumsi ini tidak terpenuhi, seperti pada kasus klasifikasi teks, metode klasifikasi *Naïve Bayes* tetap efektif [13].

Dalam sebuah aturan yang mudah, sebuah klasifikasi diasumsikan bahwa ada atau tidaknya ciri tertentu dari sebuah kelas tidak ada hubungannya dengan ciri ciri kelas lainnya. Untuk contohnya, buah akan dianggap sebagai buah apel jika berwarna merah, berbentuk bulat dan berdiameter sekitar 6 cm. Walaupun jika ciri-ciri tersebut bergantung satu sama lainnya atau keberadaannya merupakan ciri dari kelas lainnya, klasifikasi Naïve Bayes tetap menganggap bagian-bagian dari kelas tersebut masing-masing memberikan jawaban bahwa kelas itu adalah apel.

Berdasarkan dari ciri alami dari sebuah model probabilitas, klasifikasi Naïve Bayes bisa dibuat lebih efisien dalam bentuk pembelajaran. Dalam beberapa bentuk praktiknya, parameter untuk perhitungan model Naïve Bayes menggunakan metode *maximum likelihood* atau kemiripan tertinggi [8].



Gambar 1.7 Ilustrasi naïve bayes pada maximum likelihood [3]

Jadi sebuah data akan dilakukan klasifikasi berdasarkan sekelompok data yang sudah mengalami proses pembelajaran untuk menentukan termasuk data yang mana jika data tersebut diklasifikasikan.

Dengan aturan Bayes maka probabilitas sebuah dokumen untuk berada pada suatu kelas dapat dihitung sebagai berikut.

$$P(c|d) \propto P(c) \prod_{1 \le k \le n_d} P(t_k|c) \tag{4}$$

Pada klasifikasi teks, tujuan utama dari pendekatan Bayes adalah menemukan kelas terbaik untuk suatu dokumen. Kelas terbaik dalam metode klasifikasi *Naïve Bayes* adalah kelas yang memiliki probabilitas tertinggi atau *maximum a posteriori* (MAP) [13].

$$c_{map} = \arg\max_{c \in C} P(c|d) = \arg\max_{c \in C} P(c) \prod_{1 \le k \le n_d} P(t_k|c) \tag{5}$$

dimana,

c_{map} : Kategori yang memiliki probabilitas tertinggi.

$P(c|d)$: Probabilitas kelas c terhadap dokumen d (*posteriori probability*)

$P(c)$: Probabilitas kelas c pada data *training* (*prior probability*)

$P(t_k|c)$: Probabilitas bersyarat *term* t_k muncul pada dokumen di kelas c.

2.7 K-Nearest Neighbor

Algoritma *K-Nearest neighbor* (KNN) adalah sebuah metode untuk melakukan klasifikasi terhadap objek berdasarkan data pembelajaran yang jaraknya paling dekat dengan objek tersebut. KNN termasuk algoritma *supervised learning* dimana hasil dari *query instance* yang baru diklasifikan berdasarkan mayoritas dari kategori pada KNN. Nanti kelas yang paling banyak muncul yang akan menjadi kelas hasil klasifikasi [6].

Tujuan dari algoritma ini adalah mengklasifikasikan obyek baru bedasarkan atribut dan *training sample*. *Classifier* tidak menggunakan model apapun untuk dicocokkan dan hanya berdasarkan pada memori. Diberikan titik *query*, akan ditemukan sejumlah k obyek atau (titik *training*) yang paling dekat dengan titik *query*. Klasifikasi menggunakan *voting* terbanyak diantara klasifikasi dari k obyek algoritma KNN menggunakan klasifikasi ketetanggaan sebagai nilai prediksi dari *query instance* yang baru [14].

Algoritma metode *K-Nearest Neighbor* (KNN) sangatlah sederhana, bekerja berdasarkan jarak terpendek dari *query instance* ke *training sample* untuk menentukan KNN-nya. *Training sample* diproyeksikan ke ruang berdimensi banyak, dimana masing-masing dimensi merepresentasikan fitur dari data. Ruang ini dibagi menjadi bagian-bagian berdasarkan klasifikasi training sample. Sebuah titik pada ruang ini ditandai kelas c jika kelas c merupakan klasifikasi yang paling banyak ditemui pada k buah tetangga terdekat dari titik tersebut [8]. Dekat atau jauhnya tetangga biasanya dihitung menggunakan penghitung kemiripan dokumen diantaranya adalah menggunakan *Cosine Similarity*.

Cosine Similarity digunakan untuk mengukur tingkat kedekatan antar dua buah vektor. Berikut adalah rumus *cosine similarity* [15]:

$$\cos(x, y) = \frac{x * y}{||x|| \quad ||y||} \tag{6}$$

dimana,

$x * y$ = vektor dari x dan y, dihitung dengan $\sum_{k=1}^{n} x_k y_k$

$||x||$ = panjang vektor dari x, dihitung dengan $\sqrt{\sum_k^n {x_k}^2}$

$||y||$ = panjang vektor dari y, dihitung dengan $\sqrt{\sum_k^n {y_k}^2}$

Ketepatan algoritma *K-Nearest Neighbor* sangat dipengaruhi oleh ada atau tidaknya fitur-fitur yang tidak relevan atau jika bobot fitur tersebut tidak setara dengan relevansinya terhadap klasifikasi. Riset terhadap algoritma ini sebagian besar membahas bagaimana memilih dan memberi bobot terhadap fitur agar performa klasifikasi menjadi lebih baik [16].

Langkah-langkah untuk menghitung metode *K-Nearest Neighbor*:

1. Menentukan parameter K (jumlah tetangga paling dekat).
2. Menghitung kuadrat jarak (*query instance*) masing-masing obyek terhadap data sampel yang diberikan.
3. Kemudian mengurutkan objek-objek tersebut kedalam kelompok yang mempunyai jarak terkecil.
4. Mengumpulkan kategori Y (Klasifikasi *nearest neighbor*).
5. Dengan menggunakan kategori *nearest neighbor* yang paling mayoritas maka dapat dipredisikan nilai *query instance* yang telah dihitung.

2.8 *Evaluation Model*

Diperlukan cara yang sistematis untuk mengevaluasi kinerja dari suatu metode / model. Evaluasi klasifikasi didasarkan pengujian pada objek yang benar dan salah. Validasi data digunakan untuk menentukan jenis terbaik dari skema pembelajaran yang digunakan, berdasarkan data pelatihan untuk melatih skema pembelajaran.

1. *Confusion Matrix*

   Confusion matrix menurut Kohavi dan Provost dalam Visa, Ramsay, Ralescu, dan Van Der Knaap berisi informasi mengenai hasil klasifikasi aktual dan yang telah diprediksi oleh sistem klasifikasi. Performa dari sistem tersebut biasanya dievaluasi menggunakan data dalam sebuah matriks [17]. Tabel dibawah ini menampilkan sebuah *confusion matrix* untuk pengklasifikasian ke dalam dua kelas.

Tabel 1.3 Confusion matrix 2 kelas

#	*Relevant*	*Not Relevant*
Retrieved	*True Positive(TP)*	*False Positive(FP)*
Not Retrieved	*False Negative(FN)*	*True Negative(TN)*

Keterangan:

True Positive(TP)	:	jumlah kelas positif (kelas yang menjadi fokus klasifikasi) dan diklasifikasikan dengan benar sebagai kelas positif oleh sistem.
True Negative(TN)	:	jumlah kelas bukan positif (kelas selain kelas yang menjadi fokus klasifikasi) dan diklasifikasikan dengan benar sebagai kelas negatif oleh sistem.
False Positif(FP)	:	jumlah kelas bukan positif yang diklasifikasikan sebagai kelas positif oleh sistem.
False Negative(FN)	:	jumlah kelas positif yang diklasifikasikan sebagai kelas bukan positif oleh sistem.

Pengukuran performansi sistem akan dilakukan dengan menghitung akurasi, *precision*, *recall* dan *F1-Measure*. Menurut Christopher Manning, akurasi sudah tidak relevan lagi digunakan untuk masalah *Information Retrieval*. Hampir semua kasus terkadang data sangat condong terhadap satu kelas dari pada kelas yang lain. Umumnya lebih dari 99.9% dokumen ada di kategori yang tdak relevan. Sistem yang di atur untuk memaksimalkan akurasi dapat menghasilkan performansi yang bagus dengan menganggap semua dokumen tidak relevan terhadap semua *query*. Meskipun sistem yang dibangun bagus, tetapi sistem gagal mengenali dokumen yang relevan, sehingga sistem akan memberi label tidak relevan terhadap dokumen yang relevan tersebut [18]. Berikut penjelasan tentang akurasi, *precision*, *recall* dan *F1-Measure*.

Akurasi adalah ketepatan sistem melakukan proses klasifikasi dengan benar.

$$\text{Akurasi} = \frac{TP + TN}{TP + FP + TN + FN} \tag{7}$$

Precision adalah rasio jumlah dokumen relevan dengan total jumlah dokumen yang ditemukan oleh *classifier*.

$$Precision = \frac{TP}{TP + FP} \tag{8}$$

Recall adalah rasio jumlah dokumen dokumen yang ditemukan kembali oleh *classifier* dengan total jumlah dokumen yang relevan.

$$Recall = \frac{TP}{TP + FN} \tag{9}$$

F-Measure adalah kombinasi rata-rata harmonik dari *precision* dan *recall* yang berbanding lurus dengan nilai keduanya.

$$F_1 = \frac{2 \,.\, Precision \,.\, Recall}{Precision + Recall} \tag{10}$$