BAB 2
LANDASAN TEORI

2.1 Analisis Sentimen

Analisis sentimen adalah sebuah teknik atau cara yang digunakan untuk mengidentifikasi bagaimana sebuah sentimen diekspresikan menggunakan teks dan bagaimana sentimen tersebut bisa dikategorikan sebagai sentimen positif maupun sentimen negatif. Pendapat yang hampir sama juga digunakan untuk memahami komentar yang diciptakan oleh pengguna (internet) dan menjelaskan bagaimana produk maupun brand diterima oleh mereka. Teknik tersebut dapat menganalisis pendapat, sentimen, evaluasi, penilaian, sikap, dan emosi masyarakat terhadap entitas seperti produk, layanan, organisasi, individu, isu, peristiwa, topik, dan atribut. Selain analisis sentimen, teknik tersebut dikenal juga dengan istilah berbeda, misalnya, *opinion mining*, *opinion extraction*, *sentimen mining*, *subjectivity analysis*, *affect analysis*, *emotion analysis*, *review mining*, dan lain-lain. Dari beragam istilah tersebut, kalangan akademisi dan industri lebih sering menggunakan istilah sentiment analysis [13].

Pada penelitian tugas akhir ini akan melakukan analisis level aspek. Level aspek ini yang biasanya disebut *feature level*, *feature-based opinion mining* dan *summarization* [14]. Tujuan pada level aspek ini adalah untuk menentukan sentimen pada setiap entitas dan setiap aspeknya yang berbeda.

2.1.1 Analisis Sentimen Berdasarkan Aspek

Analisis sentimen berbasis aspek atau *Aspect-based Sentiment Analysis* merupakan perkembangan dari analisis sentimen yang hanya mengacu pada sebuah kalimat [2]. Analisis sentimen berbasis aspek dari sebuah kalimat merupakan sebuah opini yang mengacu kepada entitas yang spesifik dan aspek yang dibahasnya. Analisis sentimen berbasis aspek bertujuan untuk mendeteksi polaritas teks tertulis berdasarkan dengan aspek tertentu. Penelitian *aspect-based sentiment analysis* terdiri dari beberapa task.

Berdasarkan penelitian yang dilakukan oleh Andi Suciati dan Indra Budi [6] memiliki 2 task, yaitu *Aspect Extraction* dan *Aspect Sentiment Classification*.

1. *Aspect Extraction*

   Pada tahap ini akan mengidentifikasi aspek yang sudah ditentukan sebelumnya. Contohnya adalah, untuk kalimat "pelayanannya baik tapi kok error mulu aplikasinya", aspek yang akan diketahui adalah "layanan" dan "sistem" karena mengacu kepada entitas "customer service" dengan aspek "layanan" dan entitas "aplikasi" dengan aspek "sistem".

2. *Aspect Sentiment Classification*

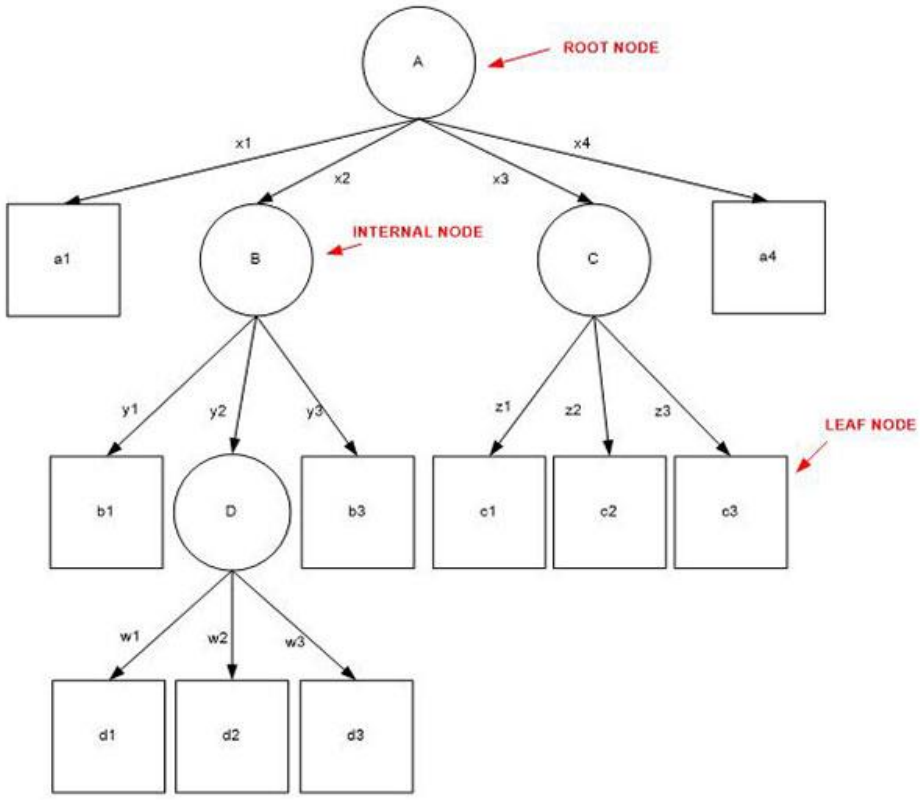
   Pada tahap ini akan menentukan polaritas sentimen kepada aspek yang telah diekstraksi dengan nilai polaritas biner yaitu, "positif" dan "negatif". Tujuan dari tahap ini adalah untuk mengidentifikasi nilai sentimen dari aspek yang telah di ekstraksi sebelumnya. Pada tahap identifikasi ini ada beberapa pendekatan yang bisa dilakukan yaitu, *rules-based*, *seeds-based*, *topic modelling*, dll. Untuk penentuan sentimen, umumnya ada dua pendekatan yaitu, *supervised learning* dan *lexicon based*. Dengan penelitian ini, identifikasi aspek yang digunakan adalah *rules-based* dan penentuan sentimen menggunakan *supervised learning*. Selain dari itu, ada juga penentuan label klasifikasi machine learning yang tergantung model digunakan. Penelitian ini menggunakan machine learning dan deep learning model untuk memberi nilai label "positif", "negatif" dan "netral" atau bisa disebut klasifikasi multi-label. Tahap lanjutannya adalah untuk memilih fitur yang akan dimasukkan tahap klasifikasi dan terdiri dari beberapa metode untuk digunakan, dalam penelitian ini digunakan fitur N-Grams dan memilih unigrams, bigram dan kombinasi unigram-bigram.

2.2 Extra Tree

Extra Tree atau bisa juga disebut Extremely Randomized Trees adalah jenis teknik pembelajaran mesin ansambel yang menggabungkan hasil dari beberapa pohon keputusan yang tidak berkorelasi yang dikumpulkan di "forest" untuk mengeluarkan hasil klasifikasinya [15].

Dalam paper dokumentasi algoritma Extra Tree oleh Pierre Geurts(et al., 2006) [16], algoritma ini sangat mirip dengan algoritma pendahulunya yaitu Random Forest dan hanya berbeda dari cara konstruksi pohon keputusan (decision tree) di hutannya (forest). Setiap pohon keputusan di *extra trees forest* dibangun dari seluruh sampel pelatihan asli (bukan replica boostrap) untuk menumbuhkan pohon, tidak ada *resampling* dan menggunakan tipe pohon CART (*Classification and Regression Trees,* Breiman et al., 1984) [17] untuk konstruksi pohon. Mekanisme dari pohon keputusan seperti struktur pohon, dimana setiap *internal node* menunjukan sebuah pengujian pada sebuah atribut, setiap cabang menunjukan hasil dari pengujian, dan *leaf node* menunjukan kelas atau kategori. Pada pohon keputusan terdapat tiga jenis *node*, yaitu:

1) *Root Node*; atau *node* akar merupakan *node* yang terletak paling atas dari suatu pohon, *node* ini tidak memiliki *parent* dan memiliki *child* lebih dari satu.
2) *Internal Node*; merupakan *node* percabangan, dimana pada *node* ini memiliki *parent* dan minimal dua *child*.
3) *Leaf Node*; merupakan *node* akhir, pada *node* ini memiliki *parent* dan tidak memiliki *child*.



Gambar 2.2.1 Arsitektur pohon keputusan

Selanjutnya adalah di setiap node pengujian, setiap pohon diberikan sampel acak dari *k* fitur dari set fitur di mana setiap pohon keputusan harus memilih fitur terbaik untuk memisahkan data berdasarkan beberapa kriteria matematika (biasanya Indeks Gini). Contoh fitur acak ini mengarah pada pembuatan beberapa pohon keputusan yang tidak berkorelasi. Jumlah pohon dalam 1 hutan secara default memiliki estimasi 100 pohon keputusan untuk 1 hutan, bisa disesuaikan jumlahnya terhadap parameter *n_estimators* dan setelah memiliki 100 pohon keputusan akan mengambil hasil klasifikasi secara suara terbanyak (majority vote) dari seluruh pohon. Untuk menghitung Indeks Gini digunakan rumus seperti tertera dalam persamaan (2.1):

$$Gini\ (D) = 1 - \sum_{i=1}^{m} (p_i)^2 \tag{2.1}$$

Keterangan:

D : Dataset dari kelas

m : Jumlah kelas variabel atribut

i : Kelas atribut

pi : Proporsi jumlah kelas dalam atribut i terhadap jumlah kelas C dalam atribut

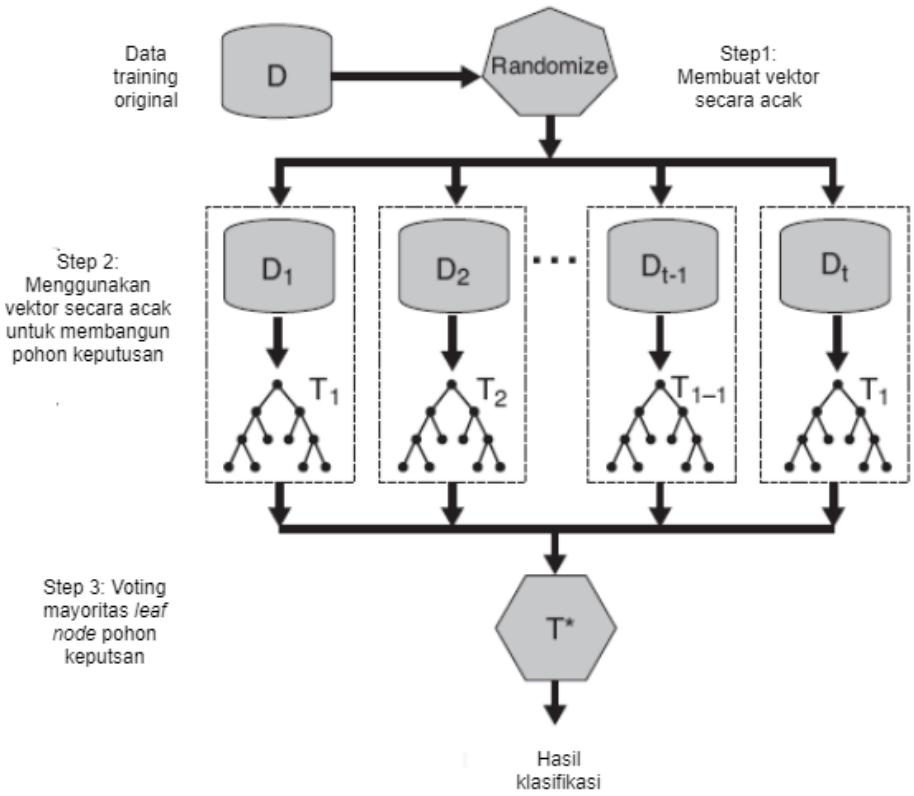
Setelah menghitung indeks gini perlu melakukan perhitungan selanjutnya untuk menentukan atribut mana yang akan masuk ke dalam node, yaitu menggunakan rumus pada persamaan *(2.2)*:

$$Gini_A(D) = \frac{|D_1|}{D} gini(D_1) + \frac{|D_2|}{D} gini(D_2) \qquad (2.2)$$

A : Jumlah bobot indeks gini untuk atribut

$D_{1,2}$: Jumlah dataset D jika di split ke A untuk menjadi dua subset D_1 dan D_2.

Dari persamaan tersebut menghasilkan nilai jumlah bobot dari indeks gini untuk fitur A dan seterusnya sebanyak fitur, setelah kalkulasi seluruh fitur untuk indeks gini berikutnya adalah mengambil nilai fitur terkecil sebagai keputusan pertama dalam pohon.



Gambar 2.2.2 Blok diagram alur pelatihan Extra Tree Classifier

Adapun tahap pelatihan untuk Extra Tree Classifier. Berdasarkan paparan pada *Gambar 2.2.2*. Tahap pertama adalah untuk mengambil seluruh data training asli untuk melakukan klasifikasi terhadap data pengujian. Selanjutnya adalah untuk memilih beberapa atribut dari data training secara acak untuk membangun pohon keputusan. Setelah memiliki atribut untuk membangun pohon keputusan adalah untuk melakukan pembangunan pohon keputusan dengan algoritma pohon keputusan CART (Classification and Regression Trees) dengan menggunakan indeks gini sebagai aturan potongan cabang pohon [17]. Proses pertama dan kedua dilakukan sebanyak 100 kali sampai semua cabang simpul mencapai leaf node atau simpul daun. Tahap terakhir adalah untuk melakukan *mayority voting* atau suara terbanyak terhadap data pengujian dari hasil pohon keputusan yang terbentuk, dari hasil suara terbanyak akan menghasilkan kelas klasifikasi untuk data yang diuji tersebut.

2.3 Decision Tree CART

Algoritma pembelajaran Pohon Keputusan CART adalah singkatan dari *Classification And Regression Trees* yang ditemukan oleh Breiman, dkk.[17] Dimana metode ini merupakan gabungan dari dua jenis pohon, yaitu classification tree dan juga regression tree. Jika variabel dependen yang dimiliki bertipe kategorik maka CART menghasilkan pohon klasifikasi (classification trees). Sedangkan jika variabel dependen yang dimiliki bertipe kontinu atau numerik maka CART menghasilkan pohon regresi (regression trees). [18]

Secara umum sangat mirip dengan C4.5, tetapi memiliki karakteristik utama sebagai berikut:

1. Daripada pohon umum yang dapat memiliki banyak cabang, CART menggunakan pohon biner, yang hanya memiliki dua cabang dari setiap simpul.
2. CART menggunakan Gini Impurity sebagai kriteria untuk membagi node, bukan Information Gain.
3. CART mendukung variabel target numerik, yang memungkinkan dirinya menjadi Pohon Regresi yang memprediksi nilai kontinu.

2.4 Google Play Store

Google Play Store, sebelumnya dikenal dengan nama Android Market, adalah layanan distribusi digital yang dioperasikan dan dikembangkan oleh Google. Aplikasi atau situs ini berfungsi sebagai toko aplikasi resmi untuk perangkat bersertifikat yang berjalan pada sistem operasi Android, memungkinkan pengguna untuk menjelajahi dan mengunduh aplikasi yang dikembangkan dengan kit pengembangan perangkat lunak (SDK) Android dan diterbitkan melalui Google. Google Play juga berfungsi sebagai toko media digital, menawarkan musik, buku, film, dan program televisi.

Aplikasi tersebut tersedia melalui Google Play secara gratis atau dengan biaya tertentu. Aplikasi ini dapat diunduh langsung di perangkat Android melalui

aplikasi seluler Play Store atau dengan menerapkan aplikasi ke perangkat dari situs web Google Play. Aplikasi yang memanfaatkan kemampuan perangkat keras perangkat dapat ditargetkan ke pengguna perangkat dengan komponen perangkat keras tertentu, seperti sensor gerak (untuk permainan yang bergantung pada gerakan) atau kamera depan (untuk panggilan video online). Google Play Store memiliki lebih dari 82 miliar unduhan aplikasi pada tahun 2016 dan mencapai lebih dari 3,5 juta aplikasi yang diterbitkan pada tahun 2017 [19], dengan jutaan pengguna akan lebih baik apa bila aplikasi-aplikasi tersebut dapat di analisa untuk mengurangi review buruk dan memperbaiki aplikasi terhadap ulasan pengguna.

2.5 *Web Scrapping*

Web Scrapping atau yang juga dikenal sebagai ekstraksi web atau *harvesting*, merupakan teknik mengekstrak data dari World Wide Web (WWW) dan simpan ke file sistem atau database untuk pengambilan atau analisis nanti. Biasanya, data web dihapus menggunakan Hypertext Transfer Protocol (HTTP) atau melalui web browser. Ini dilakukan baik secara manual oleh pengguna atau secara otomatis oleh bot atau *web crawler*. Fakta bahwa sejumlah data heterogen terus-menerus dihasilkan di WWW, web scraping secara luas dikenal sebagai teknik yang efisien dan kuat untuk mengumpulkan data besar [20][21].

2.6 *Preprocessing*

Preprocessing adalah tahapan untuk mempersiapkan teks menjadi data yang akan diolah di tahapan berikutnya. Teks yang akan dilakukan pada proses pada umumnya memiliki beberapa karakteristik, berdimensi tinggi, terdapat *noise* dan terdapat struktur yang tidak baik.

Tahapan *preprocessing* yang akan dilakukan di penelitian ini yaitu, *case folding*, *tokenizing*, *filtering*, *stemming*, *stopword removal*, *convert emoticon* dan *convert slangword*. [6]

2.6.1 *Case Folding*

Case Folding merupakan langkah dalam mengkonversi keseluruhan teks dalam dokumen menjadi suatu bentuk standar (biasanya huruf kecil atau *lowercase*). Contohnya adalah mengubah semua huruf dalam dokumen menjadi huruf kecil yang dimana dalam dokumen ditemui beberapa kata yang memiliki huruf besar. Hanya huruf "a" sampai dengan "z" yang diterima. Karakter selain huruf dihilangkan dan dianggap *delimiter*.

2.6.2 *Tokenizing*

Tokenizing merupakan langkah atau tahapan pemotongan string input berdasarkan tiap kata yang menyusunnya. Tokenisasi secara garis besar memecah sekumpulan karakter dalam suatu teks ke dalam satuan kata, bagaimana membedakan karakter-karakter tertentu yang dapat diperlakukan sebagai pemisah kata atau bukan. Ataupun dapat diartikan sebagai proses pemenggalan kata pada dokumen berdasarkan spasi dan tanda –(penghubung).

2.6.3 *Filtering*

Filtering, merupakan proses penghilangan kata-kata (yang dianggap) sebagai kata yang jarang dicari atau jarang digunakan sebagai keywords pada proses pencarian. Filtering dalam *machine learning* adalah tahap mengambil kata-kata penting dari hasil token. Bisa menggunakan algoritma *stoplist* (membuang kata kurang penting) atau *wordlist* (menyimpan kata penting). *Stoplist*/*stopword* adalah kata-kata yang tidak deskriptif yang dapat dibuang dalam pendekatan *bag-of-words*. Contoh stopwords adalah "yang", "dan", "di", "dari" dan seterusnya.

2.6.4 *Stemming*

Stemming adalah salah satu proses dari *text preprocessing* dimana proses *stemming* akan membuat term/kata yang ada pada tabel filtering menjadi kata dasar, dengan menghilangkan semua imbuhan yang ada pada kata tersebut (imbuhan meng-, me-, kan-, di-, i, pe-, peng-, a-, dll.) yang

meliputi awalan kata (prefixes), sisipan kata (infixes), akhiran kata (suffixes) dan atau menghilangkan awalan dan akhiran kata (confixes) pada kata turunan. Dalam pemrograman bahasa Python memiliki library yang dapat digunakan untuk proses *stemming* dalam bahasa Indonesia, yaitu dengan menggunakan *library python Sastrawi* yang menerapkan algoritma *Nazief & Adriani*.

2.6.5 *Stopword Removal*

Stopword Removal adalah proses untuk menghilangkan kata yang tidak relevan pada hasil *parsing* sebuah dokumen teks dengan cara membandingkan dengan *stoplist* yang ada. *Stoplist* berisi sekumpulan kata yang tidak relevan namun sering muncul dalam sebuah dokumen. *Stoplist* berisi sekumpulan *stopwords*.

Setiap kata akan diperiksa apakah ada dalam *stoplist* atau tidak, jika sebuah kata termasuk kedalam *stoplist* maka kata tersebut tidak akan diproses lebih lanjut dan akan dihilangkan. Sebaliknya jika sebuah kata tidak termasuk kedalam *stoplist* maka kata tersebut akan masuk ke proses berikutnya.

2.6.6 *Convert Emoticon*

Convert emoticon adalah proses mengkonversikan emoticon kedalam string yang sesuai dengan ekspresi emoticon itu sendiri. Convert emoticon dilakukan karena pada data komentar yang diambil dari halaman review terkadang memiliki opini yang menggunakan emotikon untuk menyampaikan ekspresi sentimen terhadap topik diskusi tersebut. Hal ini dirasa mempunyai pengaruh terhadap pengklasifikasian sentimen, oleh karena itu convert emoticon. Pada setelah menggunakan scraper untuk mengambil data yang ada di youtube, emoticon yang diambil berubah menjadi simbol-simbol. Rangkaian dari simbol itu bisa di konversikan menjadi sebuah kata yang memiliki makna untuk membantu menganalisis sentimen.

2.6.7 *Convert Slangword*

Slang ragam bahasa tidak resmi dan tidak baku yang sifatnya musiman, dipakai oleh kaum remaja atau kelompok sosial tertentu untuk komunikasi intern dengan maksud agar yang bukan anggota kelompok tidak mengerti. Pada sosial media seperti youtube, data komentar terdapat banyak kata-kata yang tidak baku. Kata yang tidak baku akan mempengaruhi proses selanjutnya yang nantinya sulit dideteksi saat pelabelan di setiap katanya. Kata yang tidak baku tersebut akan diubah menjadi kata yang baku. Pada penelitian kali ini menggunakan fitur lokal, dimana kata-kata yang tidak baku yang terdapat dalam data akan diubah menjadi kata baku. Perubahan kata tidak baku menjadi kata baku didapat dari kitab gaul. Proses selanjutnya yaitu kata akan ditentukan kategorinya dan dilihat setiap stemming katanya yang menandakan bahwa kata itu sudah baku. Selanjutnya kata tersebut akan dianalisis melalui proses morfologi.

2.6.8 Convert Negation

Convert Negation adalah proses yang bertujuan untuk menggabungkan kata negasi dengan kata setelahnya, missal “ga mau” menjadi “gamau”. Kata yang termasuk kata negasi memiliki jumlah sebanyak 9 kata, dapat dilihat pada tabel. Proses ini dilakukan karena dapat menentukan hasil deteksi teks.

2.7 *Feature Extraction*

Ekstraksi fitur atau *feature extraction* merupakan langkah penting dari *data mining* dan pencarian informasi. Ekstraksi fitur mengukur fitur kata-kata yang diekstraksi dari teks untuk mewakili teks informasi, dan mengubahnya dari sumber asli yang tidak terstruktur teks ke informasi terstruktur yang komputer dapat mengenali dan memproses [22][23]. Salah satu metode ekstraksi fitur adalah metode TF-IDF (*Term frequency – Inverse Document Frequency*). TF-IDF merupakan statistik numerik yang mencerminkan betapa pentingnya sebuah kata dalam sebuah dokumen dalam koleksi atau *corpus*. Metode ini memiliki dua perhitungan yaitu perhitungan TF dan IDF lalu menggabungkan dua perhitungan tersebut dengan

mengkalikannya, rumus, Pada persamaan (2.3) dapat dilihat rumus untuk menghitung TF:

$$tf_{ij} = \frac{f_d(i)}{max\ f_d(j)} \quad (2.3)$$

Dimana:

tf_{ij} = frekensi kemunculan term i pada dokumen j dibagi total term pada dokumen j

Selanjutnya adalah untuk menghitung IDF atau *Inverse Document Frequency*, pada dasarnya IDF mengukur jumlah informasi yang diberikan oleh sebuah kata, yaitu apakah term itu biasa atau jarang terjadi di semua dokumen. Berikut ini persamaan (2.4) untuk menghitung IDF:

$$IDF_t = log(\frac{D}{DF_t}) \quad (2.4)$$

Dimana:

IDF = Hasil *Inversed Document Frequency*

D = Jumlah dokumen

DF = Jumlah dokumen yang berisi term

Frekuensi kemunculan kata di dalam dokumen yang diberikan menunjukkan seberapa penting kata tersebut di dalam dokumen. Sehingga bobot hubungan antara sebuah kata dan sebuah dokumen akan tinggi apabila frekuensi keseluruhan dokumen yang mengandung kata tersebut akan rendah pada kumpulan dokumen. Setelah menghasilkan nilai perhitungan TF dan IDF tahap selanjutnya adalah untuk menggabungkan hasil perhitungan TF dan IDF supaya menghasilkan nilai TF-IDF. Berikut ini persamaan (2.5) untuk menghitung TF-IDF:

$$TF - IDF = TF\ x\ IDF \quad (2.5)$$

Dengan konvensi, nilai TF-IDF meningkat secara proporsional dengan sebuah kata berapa kali muncul dalam sebuah dokumen, namun diimbangi

oleh frekuensi kata di dalam corpus, yang membantu mengendalikan fakta bahwa beberapa kata lebih umum daripada yang lain.

2.8 Python

Python adalah bahasa pemrograman komputer, sama layaknya seperti bahasa pemrograman lain, misalnya C, C++, Pascal, Java, PHP, Perl dan lain-lain. Sebagai bahasa pemrograman, Python tentu memiliki varian, kosakata atau kata kunci, dan aturan tersendiri yang jelas berbeda dengan bahasa pemrograman lainnya.

Python menawarkan code yang ringkas dan udah dibaca. Meskipun algoritma kompleks dan alur kerja serbaguna mendukung pembelajaran mesin dan AI, kesederhanaan python memungkinkan pengembang untuk menulis sistem yang andal. Pengembang dapat mengerahkan semua upaya mereka untuk memecahkan masalah ML daripada berfokus pada nuansa teknis bahasa tersebut. Selain itu, Python menarik bagi banyak pengembang karena mudah dipelajari. Kode Python dapat dimengerti oleh manusia, yang membuatnya lebih mudah untuk membangun model untuk pembelajaran mesin.

Banyak programmer mengatakan bahwa Python lebih intuitif daripada bahasa pemrograman lainnya. Yang lain menunjukkan banyak kerangka kerja, pustaka, dan ekstensi yang menyederhanakan penerapan fungsi yang berbeda. Secara umum diterima bahwa Python cocok untuk implementasi kolaboratif ketika banyak pengembang terlibat. Karena Python adalah bahasa tujuan umum, Python dapat melakukan serangkaian tugas pembelajaran mesin yang kompleks dan memungkinkan Anda membuat prototipe dengan cepat yang memungkinkan Anda menguji produk untuk tujuan pembelajaran mesin.

2.9 Confusion Matrix

Confusion Matrix adalah sebuah metode perhitungan yang digunakan untuk mencari keakuratan pada hasil klasifikasi. Confusion Matrix

mengandung informasi yang membandingkan hasil klasifikasi yang dilakukan oleh sistem dengan klasifikasi seharusnya [24].

Pada evaluasi klasifikasi terdapat empat kemungkinan yang bisa terjadi dari hasil klasifikasi suatu data. Jika data positif dan diprediksi positif maka akan dihitung sebagai true positif dan jika data positif diprediksi negatif maka akan dihitung sebagai false negatif. Pada data negatif jika diprediksi negatif akan dihitung sebagai true negatif dan jika diprediksi positif maka akan dihitung sebagai false positif. Inilah yang dinamakan Matriks Konfusi dan untuk lebih jelasnya bisa dilihat pada Tabel 2.9.1.

Tabel 2.9.1 Confusion Matrix

	Aktual		
Prediksi	**Class**	**Positive**	**Negative**
	Positive	True Positive (TP)	False Positive (FP)
	Negative	False Negative (FN)	True Negative (TN)

2.9.1 Precision

Precision adalah tingkat ketepatan antara informasi yang diminta oleh pengguna dengan jawaban sistem, *precision* dapat dihitung dengan persamaan (2.6).

$$Precision = \frac{relevant\ item\ retrieved}{retrieved\ item} = \frac{TP}{TP + FP} \quad (2.6)$$

2.9.2 Recall

Recall adalah salah satu perhitungan keakuratan prediksi yang digunakan sebagai ukuran tingkat keberhasilan sistem dalam menemukan kembali sebuah informasi, *recall* dapat dihitung melalui persamaan (2.7).

$$Recall = \frac{relevant\ item\ retrieved}{retrieved\ item} = \frac{TP}{TP + FN} \quad (2.7)$$

2.9.3 F-Measure

F-Measure atau F-Score adalah *relative* hasil kombinasi nilai precision dengan nilai recall. F-measure dapat digunakan untuk mengukur kinerja dari sistem klasifikasi yang merupakan rata-rata harmonis dari *precision* dan *recall*. F-measure dapat memberikan penilaian yang lebih seimbang. F-measure dapat dihitung dengan persamaan (2.8).

$$F - Measure = \frac{2.(recall.precision)}{(recall.precision)} \quad (2.8)$$

2.9.4 Accuracy

Accuracy adalah tingkat kedekatan antara nilai prediksi dengan nilai aktual. Jika nilai akurasi tinggi maka sebuah sistem akan semakin bagus dalam melakukan prediksi, accuracy dapat dihitung dengan persamaan (2.9).

$$\boldsymbol{Accuracy} = \frac{\boldsymbol{Prediksi\ data\ benar}}{\boldsymbol{Total\ data}} = \frac{\boldsymbol{TP + TN}}{\boldsymbol{TP + TN + FP + FN}} \quad (2.9)$$

2.10 Unified Modeling Language

UML atau Unified Modeling Language merupakan pengganti dari metode analisis berorientasi objek dan design berorientasi objek (OOAD&D/object oriented analysis and design) yang dimunculkan sekitar akhir tahun 80-an dan awal tahun 90-an. UML memiliki beberapa jenis diagram untuk mempresentasikan proses pembuatan perangkat lunak berorientasi objek, diantaranya adalah *class diagram*, *sequence diagram, use case diagram*, dan *activity diagram*.

2.10.1 Use Case Diagram

Use Case Diagram adalah pemodelan untuk menggambarkan behavior / kelakuan sistem yang akan dibuat. Use case diagram

menggambarkan sebuah interaksi antara satu atau lebih aktor dengan sistem yang akan dibuat. Secara sederhana, diagram use case digunakan untuk memahami fungsi apa saja yang ada di dalam sebuah sistem dan siapa saja yang dapat menggunakan fungsi-fungsi tersebut [25].

Simbol-simbol dalam use case diagram memiliki arti masing-masing, definisi dari simbol-simbol use case dapat dilihat pada paparan ***Tabel 1.1.1***.

2.10.2 Use Case Scenario

Use case scenario merupakan penjelasan dari setiap *use case*. *Use case scenario* terbagi menjadi tiga bagian, diantaranya:

1. Identifikasi dan inisiasi
2. Langkah yang dilakukan
3. Kondisi, asumsi dan pertanyaan.

2.10.3 Activity Diagram

Activity Diagram merupakan rancangan aliran aktivitas atau aliran kerja dalam sebuah sistem yang akan dijalankan. Activity Diagram juga digunakan untuk mendefinisikan atau mengelompokan aluran tampilan dari sistem tersebut. Activity Diagram memiliki komponen dengan bentuk tertentu yang dihubungkan dengan tanda panah. Panah tersebut mengarah ke-urutan aktivitas yang terjadi dari awal hingga akhir.

Simbol-simbol dalam activity diagram memiliki arti masing-masing, definisi dari simbol-simbol activity diagram dapat dilihat pada paparan ***Tabel 1.1.2***.

2.10.4 Class Diagram

Class diagram adalah visual dari struktur sistem program pada jenis-jenis yang di bentuk. Class diagram merupakan alur jalannya database pada sebuah sistem. Class diagram merupakan penjelasan proses database

dalam suatu program. Dalam sebuah laporan sistem maka class diagram ini wajib ada.

Simbol-simbol dalam class diagram memiliki arti masing-masing, definisi dari simbol-simbol class diagram dapat dilihat pada paparan ***Tabel 1.1.3***.

2.10.5 Sequence Diagram

Sequence Diagram adalah suatu diagram yang menjelaskan interaksi objek dan menunjukkan (memberi tanda atau petunjuk) komunikasi diantara objek-objek tersebut. Sequence diagram digunakan untuk menggambarkan perilaku pada sebuah skenario dan mendeskripsikan bagaimana entitas dam sistem berinteraksi, termasuk pesan yang digunakan saat interaksi. Semua pesan dideskripsikan dalam urutan pada eksekusi. Sequence diagram berhubungan erat dengan Use Case Diagram, dimana 1 Use Case akan menjadi 1 Sequence Diagram.

Simbol-simbol dalam sequence diagram memiliki arti masing-masing, definisi dari simbol-simbol sequence diagram dapat dilihat dari paparan ***Tabel 1.1.4***.