BAB 2
LANDASAN TEORI

2.1. *Document Classification*

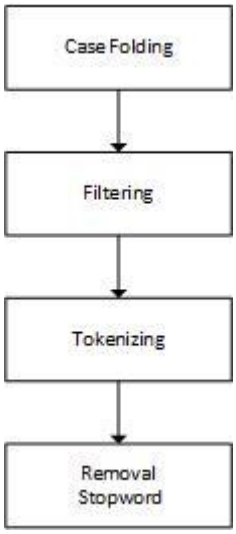
Document Classification (klasifikasi dokumen) adalah sebuah proses untuk menanggulangi munculnya sebuah masalah sederhana akan jumlah dokumen yang setiap hari semakin bertambah jumlahnya. Manfaat dari klasifikasi dokumen adalah untuk pengorganisasian dokumen[6]. Penggunaan klasifikasi dokumen tidak lain untuk membantu proses pencarian sebuah dokumen dengan cepat dan tepat. Klasifikasi dokumen mengelompokan dokumen yang sesuai dengan katergori yang terkandung pada dokumen tersebut.

Document classification pada dasarnya dibagi menjadi 2 metode dasar yaitu unsupervised document classification dan supervised document classification. Unsupervised document classification tidak memiliki pola atau aturan yang dicari sebagai pembelajaran. Sedangkan supervised document classification memiliki pola yang dijadikan sebagai patokan dalam mengklasifikasi dokumen baru, pola tersebut diperoleh dari proses pembelajaran terhadap dokumen training atau dokumen yang telah terklasifikasi. Pada supervised document classification digunakan dokumen training dan dokumen test, sedangkan unsupervised document classification tidak digunakan pembedaan dokumen training dan dokumen test.[7] Dalam klasifikasi dokumen terdapat banyak metode yang bisa digunakan. Untuk pengerjaan Tugas Akhir ini metode yang digunakan adalah metode K-Nearest Neighbor (KNN).

2.2 Text Pre-processing

Text pre-processing adalah bagian penting dari setiap sistem pemrosesan bahasa alami, karena karakter, kata, dan kalimat yang diidentifikasi pada tahap ini adalah unit dasar atau awal sebelum pemrosesan lebih lanjut [19]. *Text pre-processing* dilakukan karena data teks sering mengandung berbagai macam format khusus atau berbeda-beda seperti format angka, format tanggal dan kata-kata yang tidak membantu dan dapat dihilangkan sehingga memudahkan proses selanjutnya.

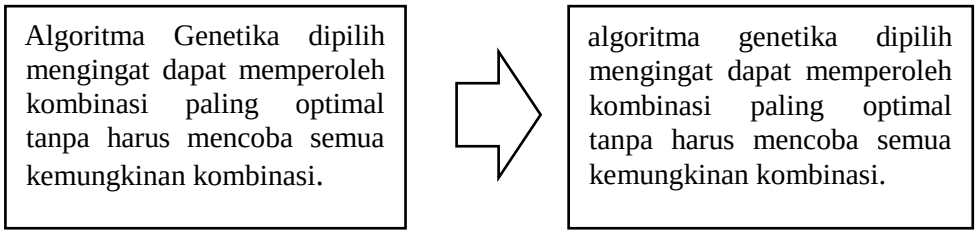
Adapun tahapan *preprocessing* yang akan dilakukan pada penelitian ini yaitu *case folding*, *tokenizing*, *filtering* dan *stopword*. Berikut adalah gambaran tahapan *preprocessing* yang dapat dilihat pada Gambar 2.1 :



Gambar 2. 1 Tahapan *Preprocessing*

2.2.1. Case Folding

Case folding merupakan proses yang dilakukan untuk menyeragamkan karakter pada data (dokumen/teks). Tahapan proses mengubah semua huruf dalam teks dokumen menjadi huruf kecil, Berikut adalah hasil proses *case folding* pada Gambar 2.2 .



Gambar 2. 2 Proses *Case Folding*

2.2.2. Filtering

Filtering adalah proses melakukan penyaringan dan menghilangkan simbol-simbol yang ada di dalam dokumen. Proses *filtering* dilakukan untuk mencegah terjadinya salah pemahaman pada komputer. Kata yang terdapat simbol, di depan, di belakang atau pun di antara huruf-hurufnya, akan

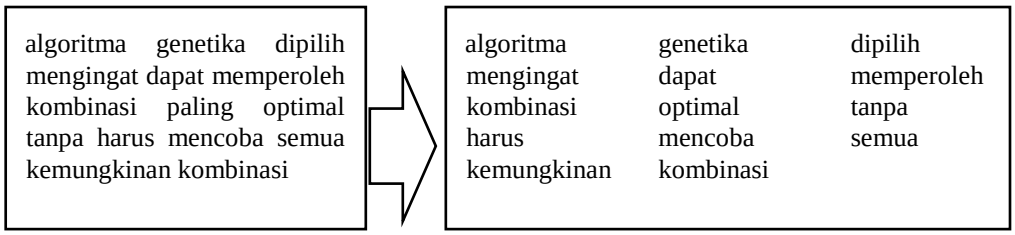
membuat kata tersebut dianggap oleh komputer memiliki makna yang berbeda dari yang seharusnya. Misalnya, pada Gambar 2.3, kata "(carita istilah indung)" akan dianggap berbeda oleh komputer dengan kata "carita istilah indung". Berikut adalah hasil proses *filtering* pada Gambar 2.3.

Gambar 2. 3 Proses *Filtering*

Hasilnya, simbol "(" dan ")" pada kata "(carita istilah indung)" dihilangkan, sehingga menjadi "carita istilah indung".

2.2.3. Tokenizing

Tokenizing adalah tahap pemotongan kalimat inputan menjadi kata perkata pada setiap 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, Berikut adalah hasil proses *tokenizing* pada Gambar 2.4.



Gambar 2. 4 Proses *Tokenizing*

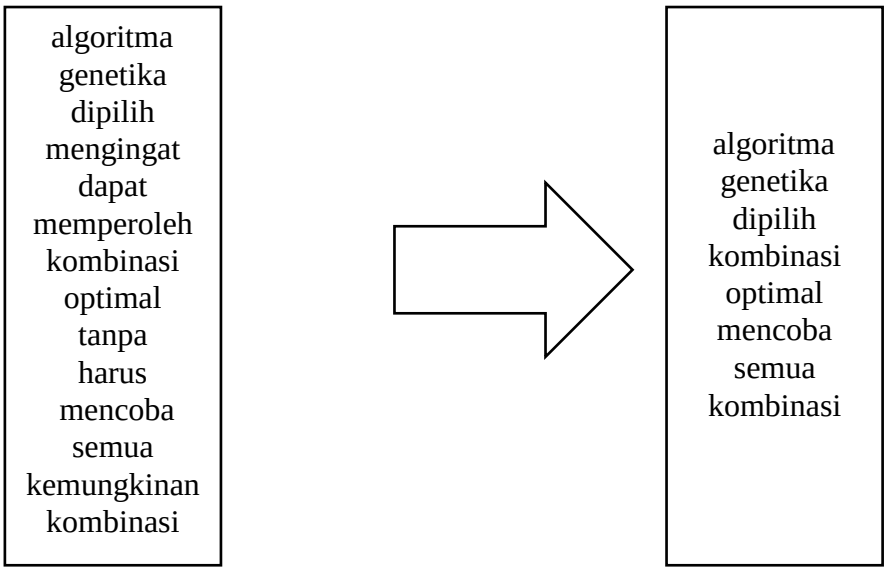
2.2.4 Stopword Removal

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 masuk kedalam *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 s*toplist* maka kata tersebut akan masuk ke proses berikutnya.

Stoplist yang digunakan diambil dari penelitian Fadillah Z. Tala [8]. Jumlah *stopword* yang terdapat pada penelitian tersebut sebanyak 756 kata. Kata – kata yang termasuk kedalam *stoplist* biasanya berupa kata ganti orang, kata penghubung dan lain sebagainya, Berikut adalah hasil proses *Stopword Removal* pada Gambar 2.5.



Gambar 2. 5 Proses *Removal Stopword*

2.3. Algoritma Term Frequency–Inverse Document Frequency

Algoritma Term Frequency – Inverse Document Frequency (TF-IDF) adalah salah satu algoritma yang dapat digunakan untuk menganalisa hubungan antara sebuah frase/kalimat dengan sekumpulan dokumen. Inti utama dari algoritma ini adalah melakukan perhitungan nilai TF dan nilai IDF dari sebuah setiap kata kunci terhadap masing-masing dokumen. Algoritma ini akan menghitung bobot setiap token t di dokumen d dengan rumus:

$$W_{dt} = tf_{dt} * IDF_t \qquad (1)$$

Dimana :

d : dokumen ke-d

t : kata ke-t dari kata kunci

W : bobot dokumen ke-d terhadap kata ke-t *tf* : banyaknya kata yang dicari pada sebuah dokumen

IDF : Inversed Document Frequency
Nilai IDF didapatkan dari :

$$IDF_t = \log(D/df_t) \quad (2)$$

dimana : *D* : total dokumen

df : banyak dokumen yang mengandung kata yang dicari

2.4. K-Nearest Neighbor

Algoritma K-NN adalah suatu metode yang menggunakan algoritma *supervised* dimana hasil klasifikasi data baru berdasar kepada kategori mayoritas tetangga terdekat ke-k [10]. Salah satu kekurangan algoritma K-NN adalah efisiensinya, karena perlu membandingkan dokumen uji dengan semua sampel dalam rangkaian pelatihan. Selain itu, kinerja algoritma ini sangat bergantung pada dua faktor, yaitu fungsi kesamaan yang sesuai dan nilai parameter “k” yang sesuai [13].

Algoritma *K-Nearest Neighbor* digunakan untuk mengklasifikasikan data berdasarkan data latih terdekat. Sebagian besar contoh digunakan untuk proses klasifikasi. Objek diklasifikasikan ke dalam kelas tertentu yang memiliki jumlah maksimal data latih terdekat. Prinsip kerja dari *K-Nearest Neighbor* adalah mencari kemiripan antara dua titik yaitu titik *training* dan titik *testing* [15]. Ada banyak cara untuk mengukur nilai kemiripan antara data *testing* dengan data *training,* diantaranya *consine similarity*

Cosine similarity adalah perhitungan kesamaan antara dua vektor n dimensi dengan mencari kosinus dari sudut diantara keduanya dan sering digunakan untuk membandingkan dokumen dalam text mining. Rumus *Cosine similarity* dapat dilihat pada persamaan 3 :

$$similarity\ (x,y) = \cos\theta = \frac{\sum_{i=1}^{n} x_i y_i}{\sqrt{\sum_{i=1}^{n} x_i^2}\sqrt{\sum_{i=1}^{n} y_i^2}} \qquad (3)$$

Dimana :

$\sum_{i=1}^{n} x_i y_i$ = jumlah bobot kata dokumen x terhadap dokumen y.

$\sqrt{\sum_{i=1}^{n} x_i^2}$ = akar dari jumlah bobot dokumen x.

$\sqrt{\sum_{i=1}^{n} y_i^2}$ = akar dari jumlah bobot dokumen y.

Nilai k yang bagus dapat dipilih dengan optimasi parameter, misalnya dengan menggunakan cross-validation. Kasus khusus di mana klasifikasi diprediksikan berdasarkan data pembelajaran yang paling dekat. Ketepatan algoritma KNN ini 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 [20].

2.6. K – Fold Cross Validation

K-Fold Cross Validation adalah teknik validasi silang dengan membagi data secara acak ke dalam *k* bagian dan masing-masing bagian akan dilakukan proses klasifikasi sebanyak *k* kali. Dalam *k-fold cross-validation*, data awal dipartisi secara acak menjadi saling *k* subset eksklusif atau *fold* D1, D2, Dkk, masing-masing berukuran kurang lebih sama. Pelatihan dan pengujian dilakukan *k* kali. Dalam iterasi *i*, partisi Dii dicadangkan sebagai set tes, dan partisi yang tersisa secara kolektif digunakan untuk melatih model. Artinya di iterasi pertama, himpunan bagian D2, Dkk secara kolektif berfungsi sebagai set pelatihan untuk mendapatkan yang pertama model yang diuji pada D1 iterasi kedua dilatih pada subset D1, D3 , Dkk dan diuji pada D2 dan seterusnya. Tidak seperti metode *holdout* dan random subsampling, Di sini setiap sampel menggunakan jumlah waktu pelatihan yang sama dan satu kali untuk pengujian. Untuk klasifikasi, taksiran akurasi adalah jumlah keseluruhan klasifikasi yang

benar dari iterasi *k*, dibagi dengan jumlah total tupel pada data awal [16]. *Leave-one-out* adalah kasus khusus dari *k-fold cross-validation* dimana *k* di set ke nomor tersebut tupel awal Artinya hanya satu sampel yang ditinggalkan pada satu waktu untuk set tes. Distratifikasi *cross-validation*, lipatannya berlapis-lapis sehingga pembagian kelas tupel di setiap lipatan kira-kira sama dengan yang ada di data awal. Secara umum, *10-fold cross-validation* dan *5-fold cross-validation* direkomendasikan untuk memperkirakan akurasi (bahkan jika daya komputasi memungkinkan penggunaan lebih banyak lipatan) karena biasnya yang relatif rendah dan varian [16].

2.7. Metode Waterfall

Model ini mengusulkan sebuah pendekatan kepada pengembangan software yang sistematik dan sekuensial yang mulai dari tingkat kemajuan sistem pada seluruh analisis, desain, kode, pengujian dan pemeliharaan. Model ini melingkupi aktivitas-aktivitas sebgai berikut : rekayasa dan pemodelan sistem informasi, analisis kebutuhan, desain, koding, mengujian dan pemeliharaan. Tahap-tahap waterfall secara umum dideskripsikan sebagai berikut :

1. **Analisis**

Tahap ini pengembang sistem diperlukan komunikasi yang bertujuan untuk memahami perangkat lunak yang diharapkan oleh pengguna dan batasan perangkat lunak tersebut. Informasi ini biasanya dapat diperoleh melalui wawancara, diskusi atau survei langsung. Informasi dianalisis untuk mendapatkan data yang dibutuhkan oleh pengguna.

2. **Perancangan**

Spesifikasi kebutuhan dari tahap sebelumnya akan dipelajari dalam fase ini dan desain sistem disiapkan. Desain Sistem membantu dalam menentukan perangkat keras*(hardware)* dan sistem persyaratan dan juga membantu dalam mendefinisikan arsitektur sistem secara keseluruhan.

3. **Implementasi**

Pada tahap ini, sistem pertama kali dikembangkan di program kecil yang disebut *unit,* yang terintegrasi dalam tahap selanjutnya. Setiap *unit* dikembangkan dan diuji untuk fungsionalitas yang disebut sebagai *unit testing*.

4. **Pengujian**

Seluruh *unit* yang dikembangkan dalam tahap implementasi diintegrasikan ke dalam sistem setelah pengujian yang dilakukan masing-masing *unit*. Setelah integrasi seluruh sistem diuji untuk mengecek setiap kegagalan maupun kesalahan.

5. **Maintenance**

Tahap akhir dalam model *waterfall*. Perangkat lunak yang sudah jadi, dijalankan serta dilakukan pemeliharaan. Pemeliharaan termasuk dalam memperbaiki kesalahan yang tidak ditemukan pada langkah sebelumnya. Perbaikan implementasi *unit* sistem dan peningkatan jasa sistem sebagai kebutuhan baru.

2.8. Unified Modeling Language (UML)

Unified Modelling Language (UML) adalah sebuah pemodelan standar dalam industri untuk visualisasi, merancang, dan mendokumentasikan sistem piranti lunak. Dengan menggunakan UML dapat membuat model untuk semua jenis aplikasi piranti lunak, aplikasi tersebut dapat berjalan pada piranti keras, sistem operasi, dan jaringan apapun, serta ditulis dalam bahasa pemrograman apapun. UML memuat diagram-diagram pemodelan sistem yang terdiri dari Use Case Diagram (diagram kasus), Class Diagram (diagram kelas), Activity Diagram (diagram aktivitas), Sequence Diagram (diagram urutan) [17].

2.8.1. Usecase Diagram

Use Case Diagram menggambarkan fungsionalitas yang diharapkan dari sebuah sistem. Sebuah *use case* merepresentasikan sebuah interaksi antara aktor dengan sistem. *Use Case Diagram* adalah diagram yang menggambarkan kebutuhan sistem dari sudut pandang *user,* yang memperlihatkan hubunganhubungan yang terjadi antara Aktor dengan *use case* dalam sistem [17].

2.8.2. Skenario Use Case

Skenario use case menjelaskan masing-masing use case yang terdapat pada Use Case Diagram. Penjelasan tersebut berkaitan dengan reaksi atau tanggapan dari sistem terhadap suatu aksi yang diberikan oleh aktor. Setiap use case memiliki skenario normal dan skenario alternatif [17].

2.8.3. Activity Diagram

Activity Diagram menggambarkan alir aktivitas pengguna dengan sistem, bagaimana masing-masing alir berawal, decision yang mungkin terjadi, dan bagaimana mereka berakhir. Activity Diagram tidak menggambarkan behaviour internal sebuah sistem (dan interaksi antar sub-sistem) secara eksak, tetapi lebih menggambarkan proses dan jalur-jalur aktivitas dari level atas secara umum [17].

2.8.4. Sequence Diagram

Sequence Diagram menggambarkan interaksi antar obyek di dalam dan di sekitar sistem (termasuk pengguna, *display*, dan sebagainya) berupa *message* yang digambarkan terhadap waktu. *Sequence Diagram* terdiri dari antara dimensi vertikal (waktu) dan dimensi horizontal (obyek-obyek yang terkait). *Sequence Diagram* biasa digunakan untuk menggambarkan skenario atau rangkaian langkah langkah yang dilakukan sebagai *respond* dari sebuah *event* untuk menghasilkan *output* tertentu. Diawali dari apa yang me-*trigger* aktivitas tersebut, proses dan perubahan apa saja yang terjadi secara internal dan *output* apa yang dihasilkan [17].

2.8.5. Class Diagram Conceptual

Class Conceptual Diagram merupakan tahap pemodelan dari perancangan sistem yang digambarkan dalam bentuk diagram, dengan berisikan *class-class* yang digunakan pada sistem tanpa mendefinisikan terlebih dahulu *method* maupun atribut didalamnya. *Class Conceptual Diagram* juga dijadikan sebagai gambaran dalam pembuatan *Class Diagram*, dan diperoleh dari *use case* [17].

2.8.6. Class Diagram

Class Diagram adalah sebuah spesifikasi yang jika diinstansi akan menghasilkan sebuah objek dan merupakan inti dari pengembangan dan desain berorientasi objek. Class menggambarkan keadaan (atribut atau properti) suatu

sistem, serta menawarkan layanan untuk memanipulasi keadaan tersebut (metoda atau fungsi). Class Diagram digambarkan struktur dan deskripsi Class, package dan objek beserta hubungan satu sama lain seperti containment, pewaris, asosiasi dan lain-lain [17].

2.9. MySQL

MySQL adalah sebuah perangkat lunak sistem manajemen basis data SQL (bahasa Inggris: database management system) atau DBMS yang multithread, multi-user, dengan sekitar 6 juta instalasi di seluruh dunia. MySQL AB membuat MySQL tersedia sebagai perangkat lunak gratis dibawah lisensi GNU General Public License (GPL), tetapi mereka juga menjual dibawah lisensi komersial untuk kasus-kasus dimana penggunaannya tidak cocok dengan penggunaan GPL .

2.10. NetBeans IDE

NetBeans adalah suatu serambi pengembangan perangkat lunak yang ditulis dalam bahasa pemrograman Java. Serambi Pada NetBeans, pengembangan suatu aplikasi dapat dilakukan dimulai dari setelan perangkat lunak modular bernama *modules*. Semula, aplikasi NetBeans IDE ini diperuntukkan bagi pengembangan dalam Java. Namun, aplikasi ini juga mendukung program-program pembuatan bahasa lain secara khusus seperti PHP, C/C++ dan HTML5.

2.11. Java

Java adalah bahasa pemrograman yang multi platform dan multi device. Sekali anda menuliskan sebuah program dengan menggunakan Java, anda dapat menjalankannya hampir di semua komputer dan perangkat lain yang support Java, dengan sedikit perubahan atau tanpa perubahan sama sekali dalam kodenya. Aplikasi dengan berbasis Java ini dikompulasikan ke dalam p-code dan bisa dijalankan dengan Java Virtual Machine. Fungsionalitas dari Java ini dapat berjalan dengan platform sistem operasi yang berbeda karena sifatnya yang umum dan non-spesifik.