BAB 2
LANDASAN TEORI

2.1. Analisis Sentimen

Analisa sentimen atau biasa disebut *opinion mining* merupakan salah satu cabang penelitian *Text Mining*. *Opinion mining* adalah riset komputasional dari opini, *sentimen* dan emosi yang diekspresikan secara tekstual, untuk menganalisa pendapat, *sentimen*, *evluasi*, penilaian sikap dan emosi terhadap *entitas* seperti produk, jasa, organisasi, individu, peristiwa dan *atribut* lainnya [8]. *Sentiment analiysist* atau *analisis sentimen* dalam bahasa Indonesia adalah sebuah teknik atau cara yang digunakan untuk mengidentifikasi bagaimana sebuah *sentimen* diekspresikan menggunakan teks dan bagaimana *sentimen* tersebut bisa dikatagorikan sebagai *sentimen* positif maupun *sentimen* negative, jika diberikan suatu *set* dokumen teks yang berisi opini mengenai suatu objek, maka *opinion mining* bertujuan untuk mengekstrak *atribut* dan komponen dari *objek* yang telah dikomentasi pada setiap dokumen dan untuk menentukan apakah komentar tersebut bermakna positif atau negative [9].

Sentiment Analysis dapat dibedakan berdasarkan sumber datanya, beberapa *level* yang sering digunakan dalam penelitian *Sentiment Analysis* adalah Sentiment *Analysis* pada *level* dokumen dan *Sentiment Analysis* pada *level* kalimat [3]. Berdasarkan *level* sumber datanya *Sentiment Analysis* terbagi menjadi 2 kelompok besar yaitu: *Coarse-grained Sentiment Analysis* dan *fined-grained Sentiment Analysis*. Pada *Sentiment Analysis Coarse-grained*, *Sentiment Analysis* yang dilakukan adalah pada *level* dokumen. Secara garis besar fokus utama dari *Sentiment Analysis* jenis ini adalah menganggap seluruh isi dokumen sebagai sebuah sentiment positif atau sentiment negatif. *Fined-grained Sentiment Analysis* adalah *Sentiment Analysis* pada level kalimat. Fokus utama *fined-greined Sentiment Analysis* adalah menentukan *sentimen* pada setiap kalimat [3].

Analisis level aspek menunjukan performa yang lebih baik dibandingkan *level* dokumen dan *level* kalimat, *analisis* dalam menemukan secara baik mana yang seseorang suka dan tidak. Hal tersebut dikarenakan dalam suatu opini yang

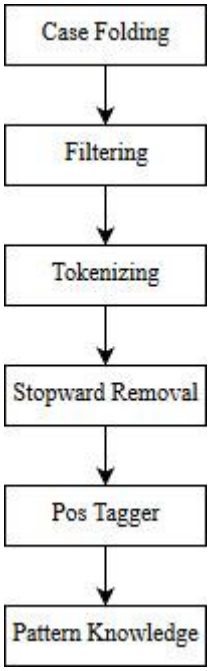
dikemukakan konsumen cenderung membahas tiap aspek produk bukan keseluruhan suatu produk, aspek-aspek yang diulas dalam satu ulasan dapat dinilai berbeda-beda bergantung dengan apa yang dirasakan dari pengalaman dalam menggunakan produk tersebut. Level aspek ini dapat juga disebut *features level, feature-based opinion mining dan summarization [5]*.

Terdapat tiga tahapan umum dalam proses *analisis sentimen*, yaitu pengambilan data dari sumber data atau yang biasa disebut *scraping*, selanjutnya tahap training dan testing dengan menggunakan *algoritma klasifikasi* tertentu. Dalam *analisis sentimen* terdapat banyak metode yang bisa digunakan, dalam penelitian ini metode yang digunakan adalah metode *Learning Vector Quantization* (LVQ).

2.2. Pre-processing

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 [10]. *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.

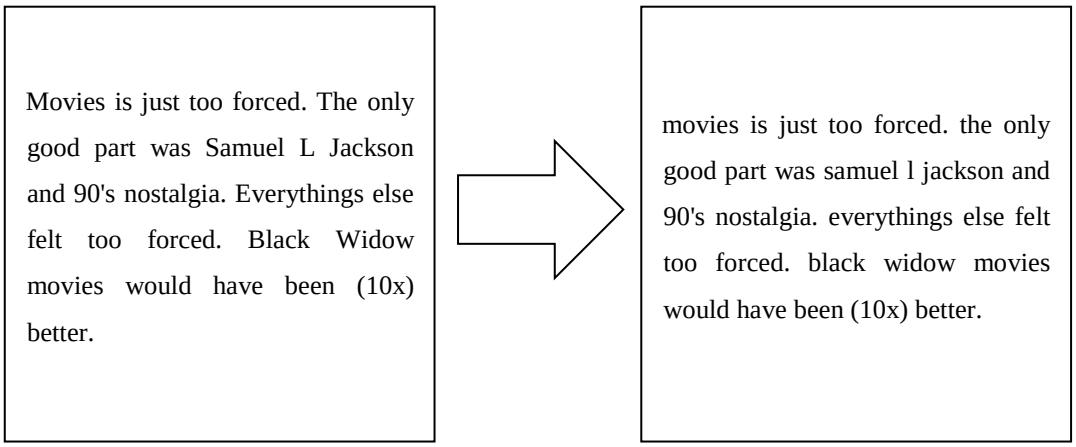
Adapun tahapan preprocessing yang akan dilakukan pada penelitian ini yaitu *Case folding*, *filtering*, *tokenizing*, *stopword removal*, *pos tagger*, dan *pattern knowledge*. Berikut adalah gambaran tahapan *preprocessing* yang dapat dilihat pada Gambar 2.1:



Gambar 2.1 Tahapan Praprocessing

2.2.1. Case Folding

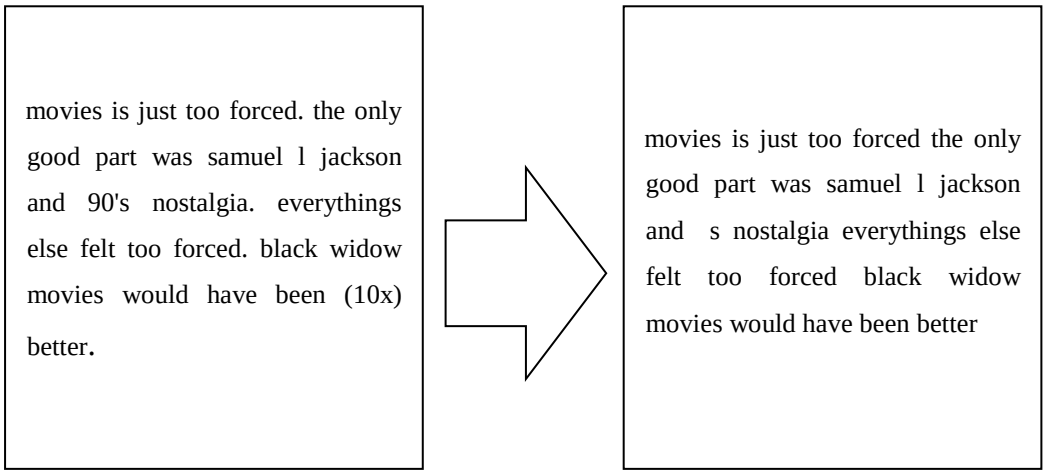
Case folding merupakan proses mengkonversi keseluruhan teks dalam dokumen atau kalimat menjadi suatu bentuk standar (biasanya huruf kecil atau *lowercase*) [11]. Semua huruf dalam dokumen atau kalimat menjadi huruf kecil, hanya huruf ‘a’ sampai dengan ‘z’ yang diterima. Hasil dari *case folding* pada Gambar 2.2.



Gambar 2.2 Proses Case Folding

2.2.2. Filtering

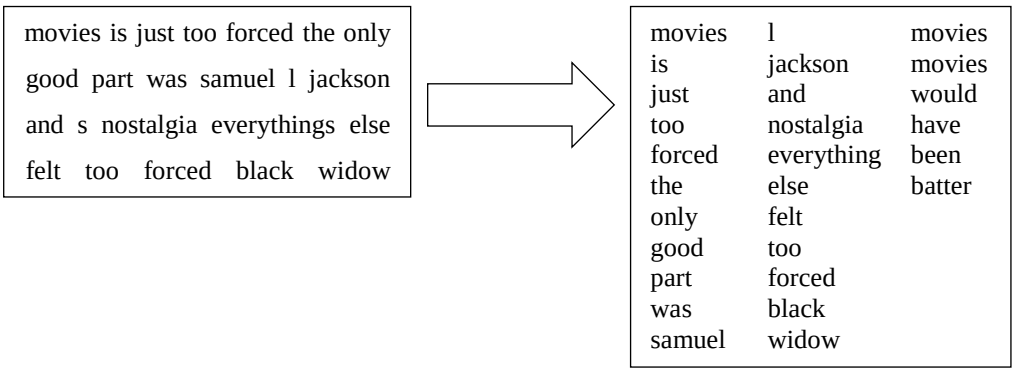
Filtering adalah proses melakukan penyaringan dan menghilangkan simbol-simbol yang ada di dalam dokumen. Proses *filtering* dilakukan untuk mencegah terjadinya salah pemahaman pada komputer [12]. Kata yang terdapat simbol, di depan, di belakang atau pun di antara huruf - hurufnya, akan membuat kata tersebut dianggap oleh komputer memiliki makna yang berbeda dari yang seharusnya. Berikut adalah hasil proses filtering pada Gambar 2.3.



Gambar 2.3 Proses Filtering

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 [13], Berikut adalah hasil proses tokenizing pada Gambar 2.4.

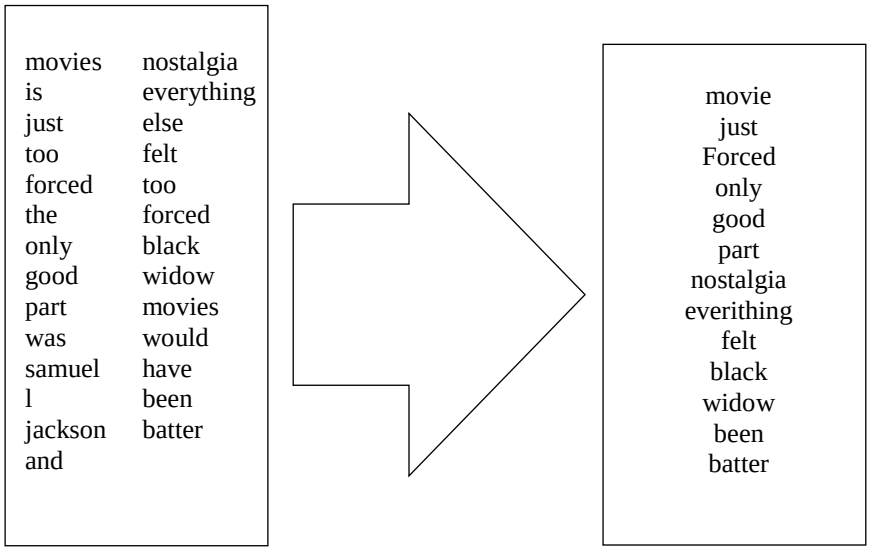


Gambar 2.4 Proses Tokenizing

2.2.4. Stopword Removal

Stopword Removal yaitu proses untuk membuang kata - kata yang tidak relevan pada hasil parsing sebuah dokumen atau kalimat dengan cara membandingkan dengan *stoplist* yang ada. *Stoplist* berisi sekumpulan kata yang tidak relevan namun sering muncul dalam sebuah dokumen. *Stoplist* berisi sekumpulan *stopwords* [14]. *Stopwords* merupakan daftar kata-kata yang tidak mengandung suatu informasi, kata-kata yang berada dalam daftar *stopword* berupa kata ganti orang, kata ganti penghubung dan penunjuk. *Stopwords* yang digunakan yaitu *SEO Stopwords.*

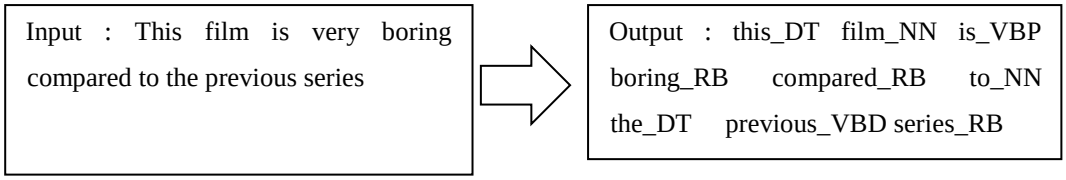
Berikut adalah hasil proses *Stopword Removal* pada Gambar 2.5.



Gambar 2. 5 Proses Removal Stopward

2.2.5. Stanford Part Of Speech Tagger

Part-Of-Speech Tagger yang dikenal dengan lebih singkat sebagai *POS Tagger* adalah sebuah bagian *software* yang membaca *input* teks dengan bahasa tertentu dan menentukan label seperti kata benda, kata sifat, kata kerja dan lainnya [15]. Stanford *POS Tagger* adalah *POS Tagger* yang dikembangkan oleh Stanford dalam penggunaan Bahasa Inggris. Berikut contoh *input* dan *output* yang dihasilkan dari *POS Tagger* pada Stanford.



Gambar 2.6 Proses Pos Tagger

2.2.6. Pattern knowledge

Pattern Knowledge merupakan salah satu metode pendekatan untuk ekstraksi aspek dan opini yang ada pada kata benda, kata sifat atau frase kata benda denggan menggunakan pola pengetahuan [16]. *Pattern rule* digunakan untuk mengekstrak fitur yang merupakan noun atau noun pharase,

Tabel 2.1 Pattern Knowledge Rule

Pattern	Kata Pertama	Kata Kedua	Kata Ketiga
Pattern 1	JJ	NN/NNS	-
Pattern 2	JJ	NN/NNS	NN/NNS
Pattern 3	RR/RBR/RBS	JJ	-
Pattern 4	RR/RBR/RBS	JJ/RR/RBR/RBS	NN/NNS
Pattern 5	RR/RBR/RBS	VBN/VBD	-
Pattern 6	RR/RBR/RBS	RR/RBR/RBS	JJ
Pattern 7	VBN/VBD	NN/NNS	-

Pattern	Kata Pertama	Kata Kedua	Kata Ketiga
Pattern 8	VBN/VBD	RR/RBR/RBS	-
Pattern 9	NN/NNS	JJ	-
Pattern 10	NN	-	-
Pattern 11	VB	-	-
Pattern 12	VB	RP	-
Pattern 13	DT	NN	-
Pattern 14	NN	NN	-
Pattern 15	JJ	VB	NN
Pattern 16	NN	VB	NN
Patteern 17	NN	IN	NN
Pattern 18	NN	NN	NN

2.2.6. N-Gram

N-gram merupakan salah satu proses yang secara luas digunakan dalam *text mining* (pengolahan teks) dan pengolahan bahasa. Secara N-gram merupakan sekumpulan kata yang diberikan dalam sebuah paragraf dan ketika menghitung n-gram biasanya dilakukan dengan menggerakkan satu kata maju ke depan (Meskipun dalam prosesnya terdapat suatu proses dimana kata yang dimajukan sejumlah X kata). Sedangkan jika N=1 maka bisa disebut dengan ***unigram*** yang pada dasarnya hanya terdiri dari satu kata dalam sebuah kalimat. Ketika terdiri dari N=2 maka disebut dengan ***bigram***, ketika N=3 maka disebut ***trigram*** dan ketika terdir dari N>1 bisa disebut dengan four gram, five gram dan seterusnya [17].

2.3. Learning Vector Quantization (LVQ)

Learning Vector Quantization (LVQ) merupakan salah satu bagian algoritma dari Jaringan Saraf Tiruan (JST), dan bisa di sebut dengan jaringan

LVQ, yang dimana setiap unit outpunya merepresentasikan sebuah kelas. Metode LVQ digunakan untuk klasifikasi dimana arsitekturnya sudah ditentukan (kelas sudah ditentukan) [3]. Keunggulan dari metode LVQ adalah mampu untuk memberikan pelatihan terhadap lapisan-lapisan kompetitif sehingga secara otomatis dapat mengklasifikasikan vektor masukan yang diberikan [18].

Pada LVQ, satu kelas dapat diwakili oleh lebih dari satu neuron, oleh karena itu LVQ tidak membutuhkan hidden layer seperti pada MLP. Operasi yang dilakukan antara vektor dengan bobot tidak menggunakan inner-product, melainkan menggunakan kuantisasi perbedaan Euclidean kuadrat [18]. Jika vector input mendekati sama maka lapisan kompetitif akan mengklasifikasikan kedua vektor tersebut kedalam kelas yang sama.

Adapun langkah yang digunakan untuk mendapatkan nilai kuantitas vector pada metode LVQ dijabarkan seperti dibawah ini :

a. Tetapkan bobot (w), maksimum iterasi (maxEpoch), error minimum (eps), learning (α) dan nilai pengurangan learning rate (decAlfa).
b. Masukan :
    1) x : vector – vector pelatihan (x1, x2, …,xn)
    2) T : kelas actual untuk vector – vector pelatihan
c. Tetapkan kondisi awal epoch = 0
d. Kerjakan jika : epoch < maxEpoch atau α > eps
    1) Epoch = epoch+1
    2) Kerjakan untuk i=1 sampai n
       tentukan sehingga || x – wj || adalah minimum (sebut sebagai Cj)
    3) Update nilai dengan ketentuan:
        a) Jika T = Cj, maka
           Wj(baru) = wj(lama) + α(x-wj(lama))
        b) Jika T ≠ Cj, maka
           Wj(baru) = wj(lama) – α(x-wj(lama))
    4) Kurang nilai α dengan
       α = α – (α*decAlfa)

2.4. Confusion Matrix

Confusion matrix juga salah satu metode yang dapat digunakan untuk mengukur kinerja suatu metode klasifikasi. Pada dasarnya *confusion matrix* mengandung informasi yang membandingkan hasil klasifikasi yang dilakukan oleh sistem dengan hasil klasifikasi yang seharusnya [19]. Pada pengukuran kinerja menggunakan *confusion matrix*, terdapat 4 (empat) istilah sebagai representasi hasil proses klasifikasi. Keempat istilah tersebut adalah *True Positive* (TP), *True Negative* (TN), *False Positive* (FP) dan *False Negative* (FN). Nilai *True Negative* (TN) merupakan jumlah data negatif yang terdeteksi dengan benar, sedangkan *False Positive* (FP) merupakan data negatif namun terdeteksi sebagai data positif. Sementara itu, *True Positive* (TP) merupakan data positif yang terdeteksi benar. *False Negative* (FN) merupakan kebalikan dari *True Positive*, sehingga data posifit, namun terdeteksi sebagai data negatif. Pada jenis klasifikasi *binary* yang hanya memiliki 2 keluaran kelas, *confusion matrix* yaitu kelas positif dan kelas negatif. Sehingga dari keempat istilah itu kita bisa kita bisa tau mana saja yang fitur atau tahapan yang akan di gunakan. Confusion matrix adalah suatu metode yang biasanya digunakan untuk melakukan perhitungan akurasi pada konsep data mining atau Sistem Pendukung Keputusan.

Berdasarkan nilai *True Negative* (TN), *False Positive* (FP), *False Negative* (FN), dan *True Positive* (TP) dapat diperoleh nilai akurasi, presisi dan *recall*. Nilai akurasi menggambarkan seberapa akurat sistem dapat mengklasifikasikan hasil akurasi menggambarkan seberapa akurat sistem dapat mengklasifikasikan data secara benar. Dengan kata lain, nilai akurasi merupakan perbandingan antara data yang terklasifikasi benar dengan keseluruhan data. Nilai akurasi dapat diperoleh dengan Persamaan 1. Hasil akurasi menggambarkan seberapa akurat sistem dapat mengklasifikasikan Nilai presisi menggambarkan jumlah data kategori positif yang diklasifikasikan secara benar dibagi dengan total data yang diklasifikasi positif. Presisi dapat diperoleh dengan Persamaan 2. Sementara itu, *recall* menunjukkan berapa persen data kategori positif yang terklasifikasikan dengan benar oleh sistem. Nilai *recall* diperoleh dengan Persamaan 3.

$$\text{Akurasi} = \frac{\text{TP+TN}}{TP+TN+FP+FN} * 100\% \quad (1)$$

$$\textit{Precision} = \frac{\text{TP}}{FP+TP} * 100\% \quad (2)$$

$$\textit{Recall} = \frac{\text{TP}}{FN+TP} * 100\% \quad (3)$$

Dimana:

- TP adalah *True Positive*, yaitu jumlah data positif yang terklasifiksi dengan benar oleh sistem.
- TN adalah *True Negative*, yaitu jumlah data negatif yang terklasifikasi dengan benar oleh sistem.
- FN adalah *False Negative*, yaitu jumlah data negatif namun terklasifiksi salah oleh sistem.
- FP adalah *False Positive*, yaitu jumlah data positif namun terklasifikasi salah oleh sistem.

2.5. 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, studi literature, 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 istem sebagai kebutuhan baru.

2.6. Unified Modeling Language (UML)

Unified Modelling Language (UML) adalah sebuah pemodelan standar dalam industry 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) [20].

2.6.1. Usecase Diagram

Use Case Diagram menggambarkan fungsionalitas yang diharapkan dari sebuah sistem. Sebuah use case merepresentasikan sebuah interaksi antara actor

dengan sistem. *Use Case* Diagram adalah diagram yang menggambarkan kebutuhan sistem dari sudut pandang user, yang memperlihatkan hubungan - hubungan yang terjadi antara Aktor dengan *use case* dalam sistem [20].

2.6.2. Skanario 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 [20].

2.6.3. 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 *ehavior internal* sebuah sistem (dan interaksi antar sub-sistem) secara eksak, tetapi lebih menggambarkan proses dan jalur-jalur aktivitas dari level atas secara umum [20].

2.6.4. Sequance 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 cenario (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 [20].

2.6.5. Class Diagram

Class Diagram adalah sebuah spesifikasi yang jika diinstansi akan menghasilkan sebuah objek dan merupakan inti dari pengembangan dan desain berorientasi objek. *Class Diagram* 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 [20].

2.7. Python

Python adalah bahasa pemrograman interpretative multiguna. Tidak seperti bahasa lain yang susah untuk dibaca dan dipahami, *Python* lebih menekankan pada keterbacaan kode agar lebih mudah untuk memahami sintaks. Hal ini membuat *Python* sangat mudah dipelajari baik untuk pemula maupun unutk yang sudah menguaasai bahasa pemrograman lain. Bahasa ini muncul pada tahun 1991, dicrancang oleh seorang Guido Van Rossum. Sampai saat ini *Python* masih dikembangakan oleh Python Software Foundation. Bahasa *Python* mendukung hamper semua system operasi, bahkan untuk system operasi Linux, hamper semua distronya sudah menyertakan *Python* di dalamnya. Dengan kode yang simple dan mudah diimplementasikan, seorang programmer dapat lebih mengutamakan pengembangan aplikasi yang dibuat, bukan malah sibuk mencari *syntax error* [21].

2.8. Spyder

Spyder adalah open sourch Crorr-plattform integrated development environment (IDE) untuk pemrograman ilmiah dalah bahsa *Python*. Spyder terintegrasi dengan sejumlah paket terkemuka dalam tumpukan *Python* ilmiah, termasuk *NumPy*, *SciPy*, *Matplotlib*, *panda*, *IPython*, *SymPy* dan *Cython* serta perangkat lunak *open source* lainnya. Awalnya dibuat dan dikembangkan oleh Pierre Raybaut pada 2009, sejak 2012 Spyder telah dipertahankan dan terus ditingkatkan oleh tim pengembang Python ilmiah dan komunitas. Spyder dapat dikembangkan dengan plugin pihak pertama dan ketiga, termasuk dukungan untuk alat interaktif untuk inspeksi data dan menanamkan instrumen jaminan kualitas kode introspeksi *Python* dan instrumen introspeksi, seperti *Pyflakes*, *Pylint* dan *Rope* [22].