BAB 2
TINJAUAN PUSTAKA

2.1 Analisis Sentimen

Analisis sentimen atau *opinion mining* merupakan proses memahami, mengekstrak, dan mengolah data tekstual secara otomatis untuk mendapatkan suatu informasi sentimen yang terkandung dalam suatu kalimat opini. Analisis sentimen dan *opinion mining* berfokus kepada opini yang mengekspresikan sentimen positif dan negatif [1].

Secara umum, analisis sentimen sudah dilakukan penelitian pada 3 level:

1. *Document Level*

   Tugas pada level ini adalah untuk mengklasifikasi apakah seluruh pendapat dokumen mengekspresikan sentimen positif atau negatif.

2. *Sentence Level*

   Tugas pada level ini adalah untuk menentukan apakah setiap kalimat menyatakan pendapat positif, negatif, atau netral. Netral biasanya tidak ada pendapat.

3. *Entity* dan *Aspect Level*

   Tugas pada level ini adalah menentukan sentimen (positif atau negatif) berdasarkan target (aspek) dalam suatu kalimat. Seringkali analisis level dokumen atau level kalimat tidak menemukan apa yang sebenernya disukai dan tidak disukai orang.

Level yang akan digunakan dalam penelitian ini adalah *Aspect Level.* Aspek yang digunakan dalam penelitian ini berdasarkan aspek makanan, harga, pelayanan dan tempat.

2.1.1 Analisis Sentimen Berdasarkan Aspek

Analisis sentimen berdasarkan aspek adalah salah satu domain kasus *opinion mining* yang bertujuan untuk mendeteksi polaritas teks tertulis berdasarkan dengan aspek tertentu. Klasifikasi teks opini di level dokumen atau kalimat seringkali tidak cukup karena level tersebut hanya bisa mengindentifikasi sentimen secara umum. Jika diasumsikan sebuah dokumen atau kalimat memiliki sebuah sentimen positif, tidak berarti semua aspek yang berada di dokumen atau kalimat tersebut semuanya positif begitupun untuk dokumen atau kalimat yang

mengandung sentimen negatif. Untuk analisis lebih lengkap perlu ditemukan aspek dan menentukan sentimen positif atau negatif pada setiap aspek [1].

Tujuan analisis sentimen berdasarkan aspek adalah untuk mengidentifikasi aspek dari suatu entitas, dan sentimen yang diungkapkan oleh penulis komentar tentang aspek tersebut [10]. Misalnya, dari kalimat "makanan nya enak tapi pelayanannya kurang bagus" pada kalimat tersebut kata "makanan" dan "pelayanan" diidentifikasikan sebagai aspek yang kemudian ditentukan sentimennya dengan kata "enak", untuk aspek "makanan" dan mengandung sentimen positif sedangkan kata "kurang bagus", untuk aspek "pelayanan" dan mengandung sentimen negatif. Pada tahap mengidentifikasikan aspek ada beberapa pendekatan yaitu, *frequency based*, *relation based*, *supervised learning*, dan *topic modelling* dan untuk penentuan sentimen terhadap aspek mempunyai 2 pendekatan yaitu, *supervised learning* dan *lexicon based* [1].

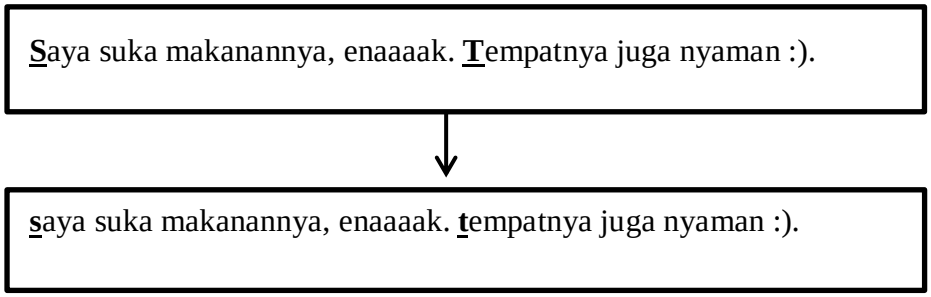
Pada penelitian ini algoritma yang digunakan untuk klasifikasi sentimen adalah *supervised learning* sehingga untuk mengembangkan sistem analisis sentimen berdasarkan aspek, dataset yang sudah di anotasi sangat penting untuk melakukan proses pelatihan dan pengujian. Dataset yang digunakan pada penelitian ini diambil dari penelitian sebelumnya [8].

2.2 Preprocessing

Preprocessing adalah proses pengolahan data asli yang sebelumnya tidak terstruktur menjadi terstruktur [11]. Proses ini bertujuan agar data yang akan digunakan dalam proses klasifikasi sentimen bersih dari *noise*. Oleh karena itu, dibutuhkan proses yang dapat mengubah data yang sebelumnya tidak terstruktur menjadi data yang terstruktur untuk diproses ke tahap selanjutnya. Adapun tahapan dari *Preprocessing* pada penelitian ini antara lain, *case folding*, *filtering*, *tokenization*, *word normalization*, *dan stopword removal*, dan *TF-IDF*.

2.2.1 *Case folding*

Case folding merupakan proses penyeragaman bentuk huruf untuk semua teks pada dokumen baik itu menjadi huruf kecil (*lowercase*) atau huruf kapital (*uppercase*). Pada penelitian ini bentuk huruf memiliki bentuk yang beragam sehingga *case folding* dilakukan untuk menghilangkan *noise*. Contoh pada *case folding* ditunjukan pada Gambar 2.1.

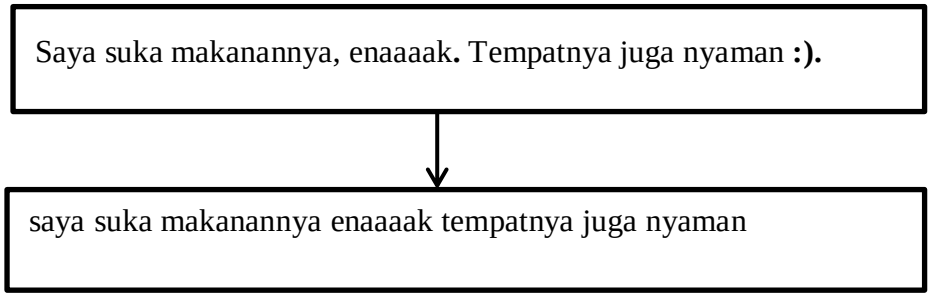


Gambar 2.1 Contoh Proses *Casefolding*

Pada Gambar 2.1 kata "Saya" dan "Tempatnya" diubah menjadi huruf kecil pada proses *case folding*.

2.2.2 *Filtering*

Filtering merupakan proses membersihkan simbol-simbol yang tidak diperlukan dalam dokumen seperti tanda baca, angka, dan *emoticon*. Pada penelitian ini *emoticon* seperti :D, :), :(, ^^, :3 akan dihilangkan. Selain itu tanda baca seperti titik (.), koma (,) dan tanda baca lainnya akan dihilangkan. Jika terdapat angka akan dihilangkan, jika terdapat spasi yang lebih dari satu maka akan diganti menjadi satu spasi. Contoh *Filtering* ditujukan pada Gambar 2.2, yaitu pada kalimat yang sebelumnya telah dilakukan proses *case folding*.

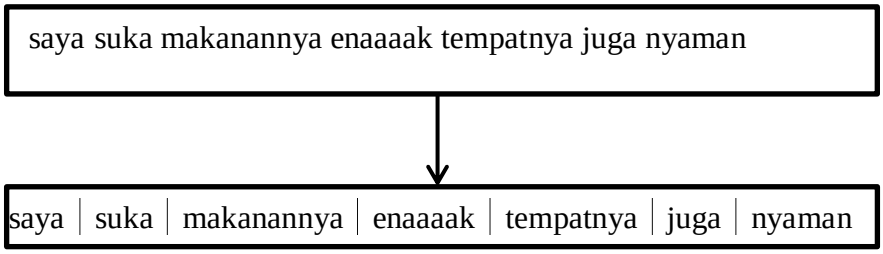


Gambar 2.2 Contoh Proses *Filtering*

Pada Gambar 2.2, semua tanda baca titik (.), koma (,) dan emoticon ":)" dihilangkan.

2.2.3 Tokenization

Tokenization merupakan proses pemisahan setiap kata dalam sebuah kalimat [12]. Proses pemisahan kata dilakukan berdasarkan spasi diantara setiap kata dalam kalimat. Hasil dari proses *tokenization* akan digunakan sebagai masukan dalam tahap transformasi teks ke dalam bentuk angka. Pada penelitian ini proses *tokenization* ditunjukan pada Gambar 2.3.

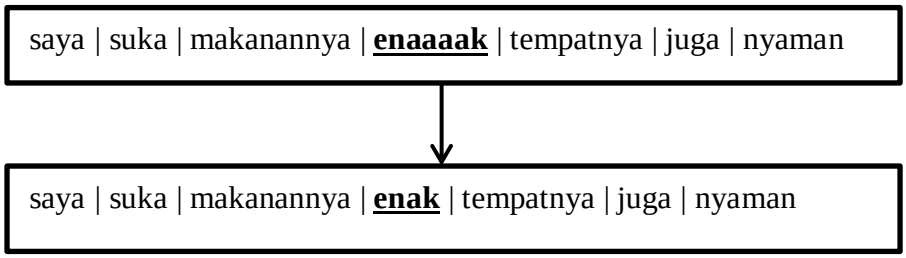


Gambar 2.3 Contoh Proses *Filtering*

Pada Gambar 2.3 kalimat akan dipisahkan berdasarkan spasi menjadi token kata.

2.2.4 *Word Normalization*

Word normalization merupakan proses mengubah kata tidak baku menjadi kata baku [13]. Pada penelitian ini dataset yang digunakan diambil penelitian sebelumnya sering kali ditemukan kata yang tidak baku yang bisa menjadi *noise* pada proses selanjutnya. Pada proses *word normalization* kata yang terdapat pada list kata tidak baku akan diubah menjadi kata baku, jika terdapat kata yang hurufnya berulang seperti, "enaaak" akan dihapus huruf berulang tersebut yang menghasilkan kata "enak" tanpa pengulangan huruf. Contoh proses *word normalization* ditujukan Gambar 2.4.

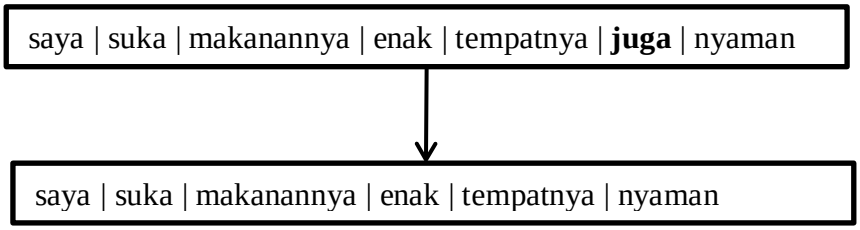


Gambar 2.4 Contoh Proses *Word Normalization*

Pada Gambar 2.4, kata "enaaaak" diubah menjadi kata "enak" karena mempunyai huruf yang berulang.

2.2.5 *Stopword Removal*

Stopword removal merupakan proses penghilangan kata tidak penting pada deskripsi melalui pengecekan kata-kata hasil parsing deskripsi apakah termasuk di dalam daftar kata tidak penting atau tidak memiliki makna seperti kata-kata di, yang, ke, oleh dan lain-lain. Proses penghilangan *stopword* berfungsi untuk mengurangi jumlah kata yang akan diproses yang tidak ada relevansinya dengan teks [13]. Pada penelitian ini kata akan dihilangkan jika terdapat pada list *stopword, list stopword* yang digunakan diambil dari library NLTK [13]. Proses *stopword removal* ditunjukan pada Gambar 2.5.



Gambar 2.5 Contoh *Proses Stopword Removal*

Pada Gambar 2.5 kata "disini" dihilangkan karena terdapat pada list *stopword*.

2.2.6 *TF-IDF*

Metode *TF-IDF* merupakan metode untuk menghitung bobot setiap kata yang paling umum digunakan pada information retrieval. Metode ini juga terkenal efisien, mudah dan memiliki hasil yang akurat [14].

Metode *TF-IDF* adalah cara pemberian bobot hubungan suatu kata (*term*) terhadap dokumen. *TF-IDF* ini adalah sebuah ukuran statistik yang digunakan untuk mengevaluasi seberapa penting seebuah kata di dalam sebuah dokumen atau didalam sekelompok kata. Untuk dokumen tunggal tiap kalimat dianggap sebagai dokumen. Frekuensi kemunculan kata di dalam dokumen yang diberikan menunjukan seberapa penting kata tersebut dalam dokumen tersebut. Frekuensi dokumen yang mengandung kata tersebut menunjukan seberapa umum kata tersebut. Bobot kata semakin besar jika sering muncul dalam suatu dokumen dan semakin kecil jika muncul dalam banyak dokumen [15]. Pada algoritma TF digunakan rumus untuk menghitung bobot(W) masing-masing dokumen terhadap kata kunci dengan menggunakan persamaan 2.1

$$IDF = log \frac{n}{df} \tag{2.1}$$

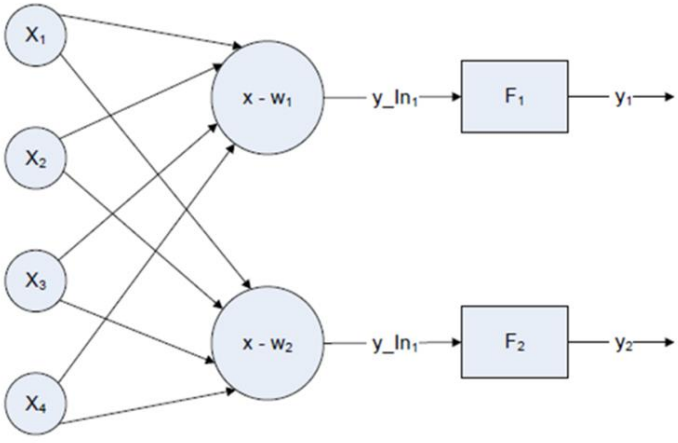
Untuk Perhitungan *TF-IDF* dapat dilihat pada persamaan 2.2

$$W = tf \times IDF_t \tag{2.2}$$

Dimana W merupakan bobot dokumen terhadap kata, tf adalah banyaknya kata yang dicari pada sebuah dokumen, N adalah total dokumen dan df adalah banyaknya dokumen yang mengandung kata yang dicari.

2.3 *Learning Vector Quantization(LVQ)*

Learning Vector Quantization(*LVQ*) merupakan salah satu bagian algoritma dari Jaringan Syaraf Tiruan, dan biasa disebut dengan Jaringan *LVQ*. Jaringan *LVQ* dibuat oleh Teuvo Kohonen untuk versi *supervised learning* dari algoritma *Self-Organizing Maps*(SOM) [10]. *Supervised* atau *active learning* adalah proses belajar yang membutuhkan guru, sesuatu yang memiliki pengetahuan tentang lingkungan [16]. Jaringan *LVQ* termasuk pada lapisan yang kompetitif, terdiri dari dua layer dan bisa bekerja untuk melakukan klasifikasi multi kelas [11], yang berarti jika 2 vektor input mendekati sama, maka lapisan kompetitif akan meletakkan kedua vektor input tersebut ke dalam kelas yang sama. Algoritma *LVQ* dapat disebut sebagai *Single Layer Feedforward* yang berarti jaringan syaraf dengan arsitektur lapis-tunggal umpan-maju [15]. Jaringan *LVQ* disebut sebagai *Single Layer Feedforward* karena "*single-layer*" yang dimaksud mengacu pada lapisan node output secara langsung [17]. Struktur jaringan *LVQ* ditunjukkan pada Gambar 2.6.



Gambar 2.6 Struktur Jaringan *LVQ*

2.3.1 Pelatihan *LVQ*

Langkah-langkah dari algoritme pelatihan *LVQ* terdiri dari [18]:

1. Inisialisasi untuk bobot awal pada algoritme *Learning Vector Quantization*. diawali dengan memilih data secara acak dari data yang ada.
2. Lakukan a sampai c bila iterasi < iterasi maksimum dan *alpha* > *alpha* minimum
   a. Lakukan penambahan nilai iterasi
   b. Lakukan langkah i sampai iii untuk semua vector masukan pada x indeks ke i = 1 sampai n Persamaan 2.2 dengan data latih yang disediakan.
      i. Hitung jarak antara data dengan bobot awal setiap kelas dengan persamaan

$$C_i = \sqrt{\sum_{j=1}^{n}(x_j - w_i)^2} \quad (2.3)$$

Dimana:

C_i = Jarak antara bobot dan data pada kelas-*i*.

n = Jumlah seluruh parameter yang ada.

x_j = data latih ke-*j*.

w_i = Bobot pada kelas ke-*i*.

      ii. Tentukan nilai minimal dari setiap jarak kelas sehigga menjadi keluaran C_i
      iii. Perbaharui bobot w_i dengan rumus berikut.
         1. Jika T= C_i, maka gunakan persamaan

$$w_i(baru) = w_i(lama) + \alpha(x_j - w_i(lama)) \quad (2.4)$$

         2. Jika T= C_i, maka gunakan persamaan

$$w_i(baru) = w_i(lama) - \alpha(x_j - w_i(lama)) \quad (2.5)$$

   c. Pengurangan *learning rate* dengan persamaan

$$\alpha(t+1) = \alpha(t) \times \left(1 - \frac{t}{T}\right) \quad (2.6)$$

α(t) adalah learning rate pada iterasi ke t. α(t+1) adalah *learning rate* pada iterasi ke (t+1). T adalah *max epoch*.

2.3.1 Pengujian *LVQ*

Proses pengujian yang dilakukan menggunakan metode LVQ adalah sebagai berikut.

1. Inisialisasi bobot awal menggunakan bobot terakhir hasil pelatihan sebelumnya.
2. Inisialisasi kondisi awal
3. Lakukan langkah 4 sampai 6 sebanyak jumlah data.
4. Hitung jarak dengan persamaan

$$D_i = \sqrt{\sum_{j=1}^{n}(x_j - w_i)^2} \quad (2.7)$$

5. Tentukan jarak terdekat (C_i) dari perhitungan berdasarkan nilai minimum dari seluruh jarak kelas yang ada (D_i)
6. Melakukan pengecekan dengan ketentuan sebagai berikut.
   a. Jika C_i = T, maka

      benar = benar+1 (2.8)

   b. Jika C_i = T, maka

      benar ≠ 0 (2.9)
7. Hitung akurasi dimana perbandingan antara jumlah data dengan jumlah total data.

$$Akurasi = \frac{jumlah\ data\ benar}{seluruh\ data} \times 100\% \quad (2.10)$$

2.4 Pemodelan Sistem

Pemodelan sistem merupakan metode yang memungkinkan untuk memodelkan sistem pada aplikasi yang akan dibangun. Adapun pemodelan yang digunakan pada penelitian ini.

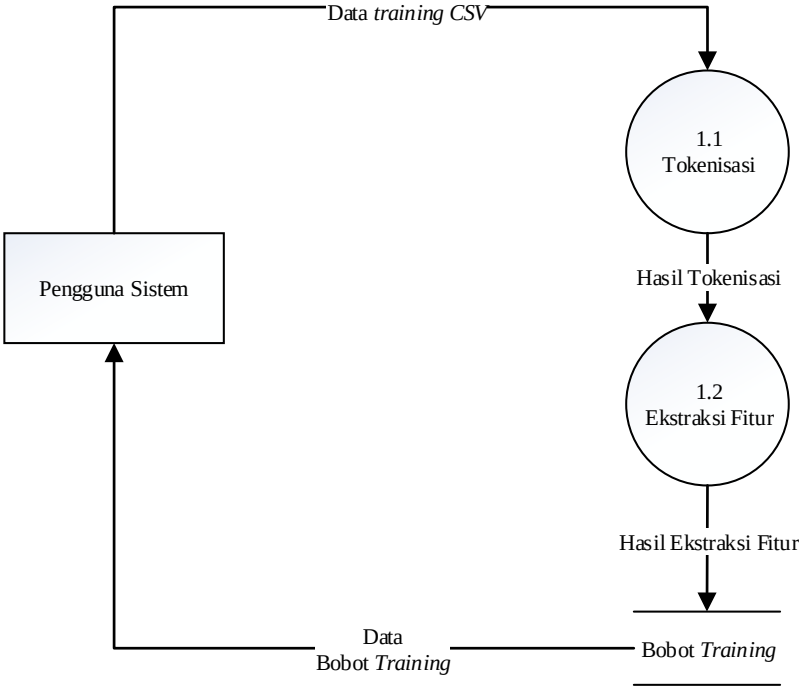
2.4.1 Diagram Konteks

Diagram Konteks adalah sebuah diagram sederhana yang menggambarkan hubungan antara *entity* luar, masukan dan keluaran dari sistem [11]. Diagram konteks dimulai dengan penggambaran terminator, aliran data, aliran kontrol penyimpanan, dan proses tunggal yang menunjukan keseluruhan sistem. Diagram

konteks merupakan level 0 atau level tertinggi dari DFD dimana diagram konteks akan menggambarkan seluruh input, proses dan output pada sistem.

2.4.2 *Data Flow Diagram*

DFD merupakan tampilan input, proses, dan output dari suatu sistem dimana objek data akan mengalir ke dalam sistem melalui input sistem dan ditransformasikan oleh elemen pemrosesan sistem kemudian data yang dihasilkan akan keluar dari sistem [11]. Objek data diwakili oleh panah berlabel dan tranformasi diwaliki oleh lingkaran. DFD digambarkan secara hierarki dimana model aliran data pertama (kadang disebut DFD level 0) mewakili sistem secara keseluruhan. Model aliran data selanjutnya memberikan masing-masing detail yang semakin meningkat. DFD menggambarkan model logika dan mengekspresikan transformasi data dalam suatu sistem. Ini termasuk mekanisme untuk memodelkan aliran data dan mendukung dekomposisi untuk mengilustrasikan detail dari aliran data dan fungsi. DFD tidak dapat menyajikan informasi tentang urutan operasi. Oleh karena itu, ini bukan metode pemodelan proses atau prosedur. Penggunaan DFD dapat dilihat pada Gambar 2.7



Gambar 2.7 Contoh *Data Flow Diagram* (DFD)

Ada empat simbol dasar yang digunakan untuk mewakili DFD.

1. Proses

   Suatu proses menerima input data dan menghasilkan output dengan konten atau formulir yang berbeda. Proses dapat sesederhana mengumpulkan data input dan menyimpan dalam database, atau bisa rumit seperti menghasilkan laporan yang berisi penjualan bulanan semua toko yang banyak.

2. *Data Flow*

   Data flow adalah jalur bagi data untuk berpindah dari satu bagian sistem informasi ke bagian lainnya. Aliran data dapat mewakili elemen data tunggal seperti ID Pelanggan atau dapat mewakili sekumpulan elemen data (atau struktur data).

3. *Data Store*

   Data store atau *data repository* digunakan dalam diagram aliran data untuk mewakili situasi ketika sistem harus menyimpan data karena satu atau lebih proses perlu menggunakan data yang disimpan di lain waktu.

4. *External Entity*

   Entitas eksternal adalah orang, departemen, organisasi luar, atau sistem informasi lain yang menyediakan data ke sistem atau menerima output dari sistem. Entitas eksternal adalah komponen di luar batas sistem informasi.

2.5 Pengujian Sistem

Pengujian sistem adalah proses pemeriksaan atau evaluasi sistem atau komponen sistem secara manual atau otomatis untuk memverifikasi apakah sistem memenuhi kebutuhan-kebutuhan yang dispesifikasikan atau mengidentifikasi perbedaan-perbedaan antara hasil yang diterapkan dengan hasil yang terjadi [12].

Pengujian seharusnya meliputi tiga konsep berikut:

1. Demonstrasi validasi perangkat lunak pada masing-masing tahap disiklus pengembangan sistem.
2. Penentuan validitas sistem akhir dikaitkan dengan kebutuhan pemakai.
3. Pemeriksaan perilaku sistem dengan mengeksekusi sistem pada data sampel pengujian.

Pada dasarnya pengujian diartikan sebagai aktivitas yang dapat atau hanya dilakukan setelah pengkodean (kode program selesai). Namun, pengujian seharusnya dilakukan dalam skala lebih luas. Pengujian dilakukan begitu spesifikasi kebutuhan telah dapat didefinisikan. Kategori pengujian dapat dikategorikan menjadi dua [12], yaitu :

1. Berdasarkan ketersediaan logik sistem, terdiri dari *Black box testing* dan *White box testing*.
2. Berdasarkan arah pengujian, terdiri dari pengujian *top down* dan Pengujian *bottom up*.

Pengujian yang akan dipakai pada penelitian ini adalah pengujian *black box*.

2.5.1 Pengujian *Black Box*

Konsep *black box* digunakan untuk mempresentasikan sistem yang cara kerja di dalamnya tidak tersedia untuk diinspeksi. Di dalam *black box*, item-item yang diuji dianggap "gelap" karena logiknya tidak diketahui, yang diketahui hanya apa yang masuk dan apa yang keluar dari *black box* [10].

1. *Graph-based testing*
2. *Equivalence partitioning*
3. *Comparison testing*
4. *Orthogonal array testing*

Pada pengujian *black box*, akan dicoba beragam masukan dan memeriksa keluaran yang dihasilkan. Sehingga dapat mempelajari apa yang dilakukan kotak, tapi tidak mengetahui sama sekali mengenai cara konversi dilakukan. Teknik pengujian *black box* juga dapat digunakan untuk pengujian berbasis skenario, dimana isi dalam sistem mungkin tidak tersedia untuk diinspeksi tapi masukan dan keluaran yang didefinisikan dengan *use case* dan informasi analisis yang lain [10].

2.6 Bahasa Pemrograman

Bahasa pemrograman atau sering diistilahkan juga dengan bahasa komputer, adalah instruksi standar untuk memerintah komputer. Bahasa pemrograman ini merupakan suatu himpunan dari aturan sintaks dan semantik yang dipakai untuk mendefinisikan program komputer.

2.6.1 *Python*

Python adalah bahasa pemrograman interpretatif multiguna dengan filosofi perancangan yang berfokus pada tingkat keterbacaan kode. *Python* diklaim sebagai bahasa yang menggabungkan kapabilitas, kemampuan, dengan sintaksis kode yang sangat jelas, dan dilengkapi dengan fungsionalitas pustaka standar yang besar serta komprehensif. *Python* sering digunakan oleh banyak aplikasi pada *data science* karena *Python* memiliki pustaka (*library*) untuk *data loading*, *visualization, statistics*, *natural language procesing*, *image processing* dan lainnya [12]. Pada penelitian ini *Python* digunakan sebagai bahasa pemrograman dalam membangun sistem analisis sentimen berdasarkan aspek.

2.6.2 *Spyder*

Spyder adalah *open source cross-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* [12].