BAB 2
LANDASAN TEORI

Landasan teori merupakan penjelasan berbagai konsep dasar dan teori-teori yang berkaitan serta mendukung penelitian ini sehingga membuat semua proses dilakukan sistematis.

2.1 Analisis Sentimen

Analisis Sentimen, atau yang dikenal juga dengan *Opinion Mining* adalah ilmu yang menganalisis opini seseorang, sentimen, perilaku, penilaian dan emosi terhadap suatu entitas, yaitu seperti produk, pelayanan, organisasi, dan individu. Secara umum, analisis sentimen dibagi menjadi tiga level bagian utama, yaitu level dokumen, level kalimat, dan level aspek dan entitas. Pada penelitian tugas akhir ini akan melakukan analisis level aspek. Level aspek ini dapat juga disebut *feature level*, *feature-based opinion mining* dan *summarization* [8]. Terdapat beberapa jenis lainnya yang memiliki tugas yang sedikit berbeda seperti *opinion extraction*, *sentiment mining*, *subjectivity analysis*, *emotion analysis*, *review mining* [9]. 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 merupakan perkembangan dari analisis sentimen yang hanya mengacu pada sebuah kalimat. Analisis sentimen berbasis aspek dari opini berbasis teks mengacu pada entitas yang spesifik dan aspek yang dibahasnya. Analisis sentimen berdasarkan aspek bertujuan untuk mendeteksi polaritas teks tertulis berdasarkan dengan aspek tertentu. Umumnya penelitian *aspect based sentiment analysis* (ABSA) terdiri dari berberapa task.

Penelitian yang dilakukan oleh Sasmita, Wicaksono, Louvan, dan Adriani [10] terdiri dari 2 task, yaitu *Aspect Extraction* dan *Aspect Sentiment Orientation Classification.*

1. *Aspect Extraction*

   Pada tahap ini mengekstrak aspek yang sudah ditentukan sebelumnya. Misalnya, dalam kalimat "servis motor disini bagus", aspeknya adalah "servis" entitasnya "motor". Pada kata "motor" tidak menunjukan aspek umum karena evaluasi bukan tentang motor secara kesuluruhan tetapi hanya tentang "servis".

2. *Aspect Sentiment Orientation Classification*

   Pada tahap ini menentukan apakah pendapat dari berbagai aspek kedalam sentimen positif, atau negatif. Pada contoh kalimat "pelayanan yang diberikan memuaskan". Sentimen dari aspek "pelayanan" adalah positif.

Sedangkan penelitian oleh Bryceryn, Konkol, dan Steinberger [11], mereka memiliki 4 subtask, yaitu *Aspect Term Extraction*, *Aspect Term Polarity*, *Aspect Category Detection* dan *Aspect Category Polarity*.

Tujuan pada level aspek ini adalah untuk mengidentifikasi aspek dari suatu entitas, dan sentimen yang diungkapkan oleh penulis komentar tentang aspek tersebut. Contoh dari tujuan pada level aspek misalnya terdapat kalimat "Hasil servis disini sangat bagus tapi harganya mahal". Pada kalimat tersebut kata "servis" dan "harga" diidentifikasikan sebagai aspek yang kemudian ditentukan sentimennya dengan kata "bagus", untuk aspek "servis" dan mengandung sentimen positif sedangkan kata "mahal", untuk aspek "harga" dan mengandung sentimen negatif. Pada tahap mengidentifikasikan aspek ada beberapa pendekatan yang bisa dilakukan 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.* Pada peneltian ini, pendekatan yang digunakan dalam identifikasi aspek adalah *supervised learning.*

Supervised learning bertujuan untuk menemukan pola baru dalam data dengan menghubungkan pola data yang sudah ada (data latih) dengan data yang baru (data uji). Untuk mengembangkan sistem analisis sentimen berdasarkan aspek, dataset latih yang sudah ada, akan di anotasikan sehingga sudah ditentukan aspek dan sentimennya untuk melakukan proses testing. Dataset yang sudah dianotasikan secara manual berasal dari kalimat review dari beberapa bengkel ahass honda di Kota Bandung pada mesin pencarian Google.

2.2 Text Mining

Text mining juga dikenal sebagai data mining teks [12] atau penemuan pengetahuan dari *database* tekstual [13]. Sesuai dengan buku *The Text Mining Handbook* [14], *text mining* dapat didefinisikan sebagai suatu proses menggali informasi dimana seorang user berinteraksi dengan sekumpulan dokumen menggunakan *tools* analisis yang merupakan komponen-komponen dalam data mining. Tujuan dari *text mining* adalah untuk mendapatkan informasi yang berguna dari sekumpulan dokumen. Jadi, sumber data yang digunakan dalam text mining adalah sekumpulan teks yang memiliki format yang tidak terstruktur atau minimal semi terstruktur. Adapun tugas khusus dari *text mining* antara lain yaitu pengkategorisasian teks dan pengelompokkan teks [15]. *Text mining* dapat memberikan solusi dari permasalahan seperti pemrosesan, pengorganisasian / pengelompokkan dan menganalisa *unstructured* data dalam jumlah besar, dalam hal ini data yang akan digunakan adalah data yang diambil dari Google. Dalam memberikan solusi, *text mining* mengadopsi dan mengembangkan banyak teknik dari bidang lain, seperti Data *Mining*, *Information Retrieval*, Statistik dan Matematik, *Machine Learning*, *Linguistic*, *Natural Language Processing* dan *Visualization*. Kegiatan riset untuk text mining antara lain ekstraksi dan penyimpanan teks, preprocessing akan konten teks, pengumpulan data statistik serta *indexing* dan analisis sentimen [15].

2.3 Google

Google adalah sebuah mesin pencari terbesar dan terbaik pada saat ini. Google merupakan sebuah perusahaan besar di amerika yang menyediakan produk dan jasa seputar internet. Pendiri dari google adalah Larry Page dan Sergey Brin. Google Inc. sendiri baru resmi berdiri pada 4 September 1998 sebagai sebuah perusahaan privat, baru kemudian cek dari Andy Bechtolsheim dicairkan dan disetor ke dalam rekening perusahaan. Kantornya yang pertama pun tak kalah sederhananya. Sebuah garasi milik seorang teman yang berlokasi di Menlo Park, California. Baru pada tahun 1999, Google berpindah kanor ke Palo Alto dan mempekerjakan 8 orang karyawan. Google memiliki kantor yang tersebar di berbagai tempat di penjuru dunia. Selain kantor pusat yang berlokasi di California, kantor Google juga berdiri di beberapa kota di Amerika Serikat. Sementara itu, di luar Amerika, ada 5 region yang dibagi berdasarkan letak geografisnya: Asia-Pasifik (termasuk Australia), Eropa, Kanada, Amerika Latin, dan Timur Tengah. Mesin pencari Google, alias Google Search, sudah bukan lagi satu-satunya produk yang ada. Seiring dengan perkembangan Google, jenis produk dan fitur yang ditawarkan pun semakin bervariasi.

Berikut beberapa diantarnya:

a. Gmail.
b. Inbox by Gmail. Dibangun oleh tim Gmail, produk ini bertujuan untuk mengorganisir kotak basuk Gmail pengguna.
c. YouTube.
d. Google Drive. Pengguna dapat menyimpan, membuat, dan membagikan file baik melalui komputer atau perangkat bergerak seperti HP dan tablet. Produk ini juga dapat disinkronisasi dengan produk atau fitur lainnya, seperti Google Docs, Google Forms, Google Sheets, Google Slides, dan Google Calendar untuk kebutuhan kerja atau studi.
e. Google+. Produk ini dapat dikatakan sebagai media sosialnya Google, di mana pengguna dapat membagikan tautan, gambar, video, atau konten lainnya dengan orang-orang yang memiliki ketertarikan yang sama.

f. Google Hangouts. Fitur ini juga dapat digunakan bersamaan dengan ketika pengguna mengakses Gmail.
g. Chrome Browser.
h. Google Maps.
i. Google Photos.
j. Google Play, termasuk Google Play Books, Google Play Movies, Google Play Music, dan Google Play Newsstand.

Pada penelitian ini, data yang akan dipakai untuk melakukan testing sistem didapat dari google dengan menggunakan google API.

2.4 Tools Pendukung

Adapun beberapa *tools* pendukung dalam pembangunan sistem yang akan dilakukan adalah sebagai berikut:

2.4.1 PHP

PHP (*Hypertext Preprocessor*) adalah bahasa komputer yang dibuat untuk pengembangan web dinamis. Teknik menggunakan PHP menurut Nugroho [16] untuk memulai program PHP, pembaca dapat memulainya dengan mengenal sebuah tag pengenal PHP yang digunakan untuk menuliskan kode PHP. Untuk menuliskan dan memperkenalkan kode PHP, pembaca harus memulainya dengan tanda (<?php), setelah tanda tersebut pemabaca dapat melanjutkan dengan kodekode program isi didalamnya. Untuk mengkhiri kode program tersebut, pembaca dapat menutupnya dengan tanda (?>).

2.5 Metode pengembangan Perangkat Lunak Terstruktur

Metode ini diperkenalkan pada tahun 1970, yang merupakan hasil turunan dari pemrograman terstruktur. Metode pengembangan dengan metode terstruktur ini terus diperbaiki sampai akhirnya dapat digunakan dalam dunia nyata. Perancangan ini bertujuan untuk membuat model SOLUSI terhadap PROBLEM yang sudah dimodelkan secara lengkap pada tahap analisis terstruktur. Ada empat kegiatan perancangan yang harus dilakukan, yaitu:

1. Perancangan arsitektural; kita merancang struktur modul P/L dengam mengacu pada model analisis yang sesuai (DFD). Langkahnya adalah: mengidentifikasi jenis aliran (*transform flow* atau *transaction flow*), menemukan batas-batas aliran (*incoming flow* dan *outgoing flow*), kemudian memetakannya menjadi striktur hirarki modul. Selanjutnya, kita alokasikan fungsifungsi yang harus ada pada modul-modul yang tepat.
2. Perancangan data; kita merancang struktur data yang dibutuhkan, serta merancang skema basisdata dengan mengacu pada model analisis yang sesuai (ERD).
3. Perancangan antarmuka; kita merancang antarmuka P/L dengan pengguna, antarmuka dengan sistem lain, dan antarmuka antarmodul.
4. Perancangan prosedural; kita merancang detil dari setiap fungsi pada modul. Notasi yang digunakan bisa berupa flow chart, algoritma, dan lain-lain.

Pastikan bahwa model perancangan yang dibuat sudah mengakomodasi kebutuhan non fungsional. Berikut ini merupakan kelebihan dan kekurangan metode perancangan terstruktur:

A. Kelebihan

Adapun kelebihan dari metode perancangan terstruktur adalah sebagai berikut [17]:

1. Milestone diperlihatkan dengan jelas yang memudahkan dalam manajemen proyek.
2. SSAD (*Structured Analisys and Design*) merupakan pendekatan visual, ini membuat metode ini mudah dimengerti oleh pengguna atau programmer.
3. Penggunaan analisis grafis dan *tool* seperti DFD menjadikan SSAD (*Structured Analisys and Design*) menjadikan bagus untuk digunakan.
4. SSAD (*Structured Analisys and Design*) merupakan metode yang diketahui secara umum pada berbagai industry.
5. SSAD (*Structured Analisys and Design*) sudah diterapkan begitu lama sehingga metode ini sudah matang dan layak untuk digunakan.

6. SSAD (*Structured Analisys and Design*) memungkinkan untuk melakukan validasi antara berbagai kebutuhan
7. SSAD (*Structured Analisys and Design*) relatif simpel dan mudah dimengerti.

B. Kekurangan

Adapun kekurangan dari metode perancangan terstruktur adalah sebagai berikut [17]:

1. SSAD (*Structured Analisys and Design*) berorientasi utama pada proses, sehingga mengabaikan kebutuhan non-fungsional.
2. Sedikit sekali manajemen langsung terkait dengan SSAD (*Structured Analisys and Design*).
3. Prinsip dasar SSAD (*Structured Analisys and Design*) merupakan pengembangan non-iterative (*waterfall*), akan tetapi kebutuhan akan berubah pada setiap proses.
4. Interaksi antara analisis atau pengguna tidak komprehensif, karena sistem telah didefinisikan dari awal, sehingga tidak adaptif terhadap perubahan (kebutuhan-kebutuhan baru).
5. Selain dengan menggunakan desain *logic* dan DFD, tidak cukup *tool* yang digunakan untuk mengkomunikasikan dengan pengguna, sehingga sangat sulit bagi pengguna untuk melakukan evaluasi.
6. Pada SSAD (*Structured Analisys and Design*) sulit sekali untuk memutuskan ketika ingin menghentikan dekomposisi dan mulai membuat sistem.
7. SSAD (*Structured Analisys and Design*) tidak selalu memenuhi kebutuhan pengguna.
8. SSAD (*Structured Analisys and Design*) tidak dapat memenuhi kebutuhan terkait bahasa pemrograman berorientasi obyek, karena metode ini memang didesain untuk mendukung bahasa pemrograman terstruktur, tidak berorientasi pada obyek.

2.6 Metode *Neighbor Weighted K-Nearest Neighbor* (NWKNN).

Metode NWKNN merupakan metode yang hampir mirip dengan metode KNN. Metode NWKNN muncul karena adanya permasalahan data tidak seimbang pada data latih. Yang membedakan antara metode NWKNN dengan metode KNN adalah adanya pemberian bobot pada kelas/jenis yang berasal dari kategori mayoritas maka diberi nilai bobot kecil, sedangkan pada kategori minoritas akan diberi nilai bobot besar [4]. Asumsi yang dikatakan bahwa data latih didistribusikan secara merata dan seimbang di setiap kelas atau kategori tidak selalu tepat. Pada kenyataannya ada kemungkinan bahwa data latih yang tidak tersebar secara merata dan seimbang. Sebuah kelas mayoritas digambarkan begitu banyak pada data latih, sedangkan kelas minoritas sebaliknya. *Neighbor-Weighted K-Nearest Neighbor* (NWKNN) dikembangkan oleh Songbo Tan pada tahun 2005 merupakan modifikasi dari KNN untuk menyeselesaikan masalah tersebut [18]. Langkah algoritma pada metode NWKNN tidak jauh berbeda dengan langkah algoritma KNN, yang membedakan adalah perhitungan bobot dan score untuk menentukan klasifikasi terhadap data uji [19]. Perhitungan bobot kategori dapat dilakukan dengan persamaan (2.1):

$$Weight_i = \frac{1}{\left(\frac{Num(C_i^d)}{Min\left\{Num\left(C_j^d\right) \middle| j=1,\ldots,k^*\right\}}\right)^{1/exp}} \qquad (2.1)$$

Keterangan:
$Num\left(C_i^d\right)$ = Banyaknya data latih d pada kelas i.
$Num\left(C_j^d\right)$ = Banyaknya data latih d pada kelas j,
Dimana j terdapat dalam himpunan k tetangga terdekat.
exp = Eksponen (nilai exp lebih dari 1).
k = Jumlah tetangga paling dekat.

Setiap nilai bobot yang didapatkan akan digunakan untuk menghitung nilai skor data uji terhadap setiap kelas/jenis. Perhitungan score pada metode NWKNN dapat dilakukan dengan persamaan (2.2):

$$\text{Skor}(X, Ci) = \text{Weight}_i * \left(\sum_{djNWKNN(X)} ((Sim(q, d_j) * \delta(d_j, C_i)))\right) \quad (2.2)$$

Keterangan:

Weight_i	= Bobot kelas *i*
djNWKNN(*X*)	= Data Latih dj, pada kumpulan tetangga terdekat dari data uji X
$\delta(d_j, C_i)$	= Akan bernilai 1 jika nilai jarak ∈ C*i* dan akan bernilai 0 jika nilai jarak ∉ C*i*
$Sim(q, d_j)$	= Nilai CosineSimilarity antara data uji dan data latih
C*i*	= Jenis atau kelas *i*

Untuk menghitung jarak kedekatan akan menggunakan *Cosine similarity*. *Cosine similarity* merupakan metode untuk mengukur jarak antara dokumen atau kalimat dengan pendekatan relevansi query. Semakin besar nilai kesamaan vektor query dengan vektor kalimat maka query tersebut dipandang semakin relevan dengan kalimat [20]. Dalam menghitung *cosine similarity*, pertama yaitu melakukan perkalian skalar antara dokumen data uji dengan dokumen data latih kemudian dijumlahkan, setelah itu melakukan perkalian antara panjang vektor dokumen data latih dengan panjang vektor dokumen data uji yang telah dikuadratkan, setelah itu di hitung akar pangkat dua. Selanjutnya hasil perkalian skalar tersebut di bagi dengan hasil perkalian panjang vektor dokumen data latih dan panjnag vektor dokumen data uji. Rumus yang digunakan untuk untuk perhitungan CosineSimilarity dapat dilihat pada persamaan (2.3).

$$CosSim(q, d_j) = \frac{\vec{d}_j \bullet \vec{q}}{\left|\vec{d}_j\right| \bullet |\vec{q}|} = \frac{\sum_{i=1}^{m} (w_{ij} \bullet w_{iq})}{\sqrt{\sum_{i=1}^{m} w_{ij}^{\ 2} \bullet \sum_{i=1}^{m} w_{iq}^{\ 2}}} \quad (2.3)$$

Keterangan:

$\vec{q}$	= Data uji
$\vec{d}_j$	= Data latih
$\vec{d}_j \,.\, \vec{q}$	= Hasil total perkalian vektor antara data latih dengan data uji
$\left\|\vec{d}_j\right\| \,.\, \|\vec{q}\|$	= Hasil Total perkalian vektor antara data latih dengan data uji
w_{ij}	= Bobot nilai i pada data latih j
w_{iq}	= Bobot nilai I pada data uji
m	= Banyaknya jumlah nilai

2.7 Pembobotan

Pembobotan adalah proses merubah term yang merupakan data kualitatif menjadi data kuantitatif sehingga bisa diproses oleh komputer. Pembobotan dilakukan pada dokumen yang sudah dilakukan preprocessing. Beberapa metode pembobotan yang umum digunakan yaitu bembobotan 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. Metode ini akan menghitung nilai Term Frequency (TF) dan Inverse Document Frequency (IDF) pada setiap token (kata) di setiap dokumen dalam korpus. Inverse document frequency (IDF) adalah pembobotan kata yang didasarkan pada banyaknya dokumen yang mengandung kata tertentu. Semakin banyak dokumen yang mengandung suatu kata tertentu, semakin kecil pengaruh kata tersebut pada dokumen. Sebaliknya, semakin sedikit dokumen yang mengandung suatu kata tertentu, semakin besar pengaruh kata tersebut pada dokumen. Rumus untuk menghitung IDF dapat dilihat pada persamaan (2.4) dibawah ini.

$$idf = \log \frac{D}{df} \quad (2.4)$$

Keterangan:
D = total dokumen
df = banyak dokumen yang mengandung kata yang dicari

Pembobotan menggunakan TF-IDF dijelaskan pada Persamaan (2.5) (Feldman,20017) [14].

$$w(t,d) = TF(t,d) * idf \ (2.5)$$

Keterangan:
w(t, d) = Bobot kata t pada dokumen d.
TF(t, d)= Jumlah kemunculan kata t pada dokumen d.
idf = Inverse Document Frequency

2.8 Text Prepocessing

Text preprocessing adalah tahap awal yang digunakan untuk mendapatkan data yang siap untuk diproses. Tahapan dalam text preprocessing yaitu: *case folding*, *convert negation*, *tokenizing*, *filtering*, dan *stemming*.

2.8.1 Case Folding

Case folding adalah proses perubahan teks dari huruf besar menjadi huruf kecil dan menghilangkan seluruh tanda baca pada kalimat.

2.8.2 *Convert Negation*

Pada sebuah kalimat *review* memungkinkan terdapat negasi yang dapat membalikan arti dari suatu kata yang akan membalikan sentimen yang tadinya negatif menjadi positif dan sebaliknya. Jika pada kalimat *review* yang mengandung kata negasi, maka gabungkan dengan kata setelahnya.

2.8.3 Tokenizing

Tokenizing adalah proses pengambilan kata yang menjadi penyusun suatu dokumen. Karakter-karakter pemisah kata akan dihilangkan karena tidak memiliki pengaruh terhadap pemrosesan teks. Karakter-karakter pemisah tersebut bisa berupa spasi, tanda baca, angka, dan karakter selain huruf. Dalam proses ini juga dilakukan pengubahan semua huruf menjadi huruf kecil.

2.8.4 Filtering

Filtering adalah proses pemilihan kata dari hasil *tokenizing* yang akan digunakan untuk merepresentasikan suatu dokumen. *Filtering* dapat dilakukan dengan dua cara yaitu *filtering* berdasarkan *stoplist* atau berdasarkan *wordlist*. *Stoplist* adalah kumpulan kata-kata tidak penting. Setiap kata dalam dokumen akan dibandingkan dengan setiap kata dalam *stoplist*. Kata dalam dokumen yang juga merupakan kata dalam *stoplist*, akan dibuang dari hasil *tokenizing*. *Wordlist* adalah kumpulan kata-kata penting yang berpotensi mampu merepresentasikan suatu dokumen. Sistem akan mengambil kata dari dokumen hasil *tokenizing* yang juga merupakan kata dalam *wordlist*. Dalam pengaplikasiannya, penggunaan *stoplist* lebih efisien daripada *wordlist*. Karena banyaknya kata yang tidak penting jauh lebih sedikit daripada kata yang penting.

2.8.5 Stemming

Proses *stemming* digunakan untuk mengganti bentuk dari suatu kata menjadi kata dasar dan kata tersebut harus sesuai dengan struktur morfologi bahasa Indonesia yang benar. Penggunaan imbuhan berupa awalan atau akhiran pada suatu kata akan dihapus sehingga akan mendapatkan inti dari kata tersebut. Salah satu algoritma *stemming* yaitu *stemming* Arifin-Setiono.

2.9 Tahapan Pengujian

Tahap pengujian sistem merupakan tahapan untuk menemukan kesalahan-kesalahan dan kekurangan-kekurangan pada sistem yang dibangun sehingga bisa diketahui apakah sistem tersebut telah memenuhi kriteria sesuai dengan tujuan atau tidak. Adapun metode pengujian yang digunakan pada sistem ini adalah sebagai berikut.

2.9.1 Pengujian *BlackBox*

Pengujian *black box* merupakan pendekatan komplementer dari teknik white box, karena pengujian black box diharapkan mampu mengungkap kelas kesalahan yang lebih luas dibandingkan teknik white box. Pengujian black box berfokus pada pengujian persyaratan fungsional perangkat lunak, untuk mendapatkan serangkaian kondisi input yang sesuai dengan persyaratan fungsional suatu program [21].

2.9.2 Pengujian Akurasi

Perhitungan akurasi dari masing-masing metode dilakukan dengan rumus pada persamaan 2.6 sebagai berikut.

$$Akurasi\ (\%) = \frac{Keseluruhan\ data\ terklasifikasi\ dengan\ benar}{Keseluruhan\ data\ testing}\ x\ 100\% \quad (2.6)$$

Rumus diatas menjelaskan bahwa “Keseluruhan data terklasifikasi dengan benar” merupakan jumlah keseluruhan prediksi sistem yang terklasifikasi dengan benar. Sedangkan untuk “Keseluruhan data testing” merupakan jumlah keseluruhan data yang diklasifikasi.