BAB II
KAJIAN PUSTAKA DAN DASAR TEORI

Pada bab ini akan dijelaskan kajian pustaka dan dasar teori pembuatan aplikasi *sentiment analysis* review barang dengan menggunakan metode *K-Nearest Neighbor*. Kajian pustaka membahas skripsi yang telah ada dan yang diusulkan. Dasar teori membahas teori yang diperlukan untuk menyusun skripsi yang diusulkan.

Teori dasar yang akan dibahas pada bab ini yaitu konsep dasar *Text Mining* yang terdiri dari *Cleaning, Parsing, Tokenization, Stopword Removal, Stemming.*

2.1 Kajian Pustaka

Kajian pustaka pada skripsi ini membahas skripsi sebelumnya yang berjudul '*Mining Product Features from Online Reviews*'. Skripsi ini membahas tentang analisis sentimen pada *website* epinions.com berbahasa inggris. *Pre-processing* yang digunakan dalam skripsi ini adala POS *tagger* (*Part of Speech)* Skripsi ini secara otomatis mengumpulkan corpus dan mengolah corpus yang hasilnya akan berupa sentimen positif, negatif dan netral. Metode untuk pengklasifikasian pada skripsi ini menggunakan *Sentiment average* dengan cara menjumlahkan nilai probabilitas yang didapatkan dari SentiWordNet. Pengujian pada skripsi ini menggunakan metode *precision* and *recall* [WZJ-10].

Perbedaan skripsi ini dengan skripsi yang dikaji terdapat pada objek yang ingin dicapai berbeda karena pada skripsi ini, penulis ingin mengolah *review* terhadap sebuah barang dan ingin diketahui sentimennya apakah positif atau negatif. Skripsi ini menggunakan metode *tf-idf* untuk pembobotan *term*. Metode yang digunakan pada skripsi ini adalah *K-Nearest Neighbor* dengan *classifier* sentimen yang mampu menentukan sentimen positif dan negatif. Pengujian pada skripsi ini menggunakan pengujian akurasi.

2.2 Text Mining

Text Mining dapat didefinisikan secara luas sebagai proses pengetahuan intensif di mana pengguna berinteraksi dengan koleksi dokumen dari waktu ke waktu dengan menggunakan berbagai macam analisis. Dalam cara yang sejalan dengan data mining, *Text Mining* berusaha untuk mengekstrak informasi yang berguna dari sumber data melalui identifikasi dan eksplorasi *patterns*. *Text Mining* menjadi menarik karena sumber data koleksi dokumen dan pola yang menarik tidak ditemukan dari database formal namun ditemukan dalam data tekstual yang tidak terstruktur pada dokumen dalam koleksi [FRS-07].

Algoritma yang digunakan pada *Text Mining*, tidak hanya melakukan perhitungan hanya pada dokumen, tetapi juga fitur. Terdapat empat macam fitur yang sering digunakan:

- Character

Character merupakan komponan individual yang dapat berupa huruf, angka, karakter spesial dan spasi. *Character* juga merupakan blok pembangun pada level paling tinggi pembentuk semantik fitur, seperti kata, term dan concept. Pada umumnya, representasi *character-based* ini jarang digunakan pada beberapa teknik pemrosesan teks.

- Words.

Kata-kata tertentu dipilih langsung dari sebuah dokumen "asli" berada pada apa yang mungkin digambarkan sebagai tingkat dasar semantik. Untuk alasan ini, fitur *word* kadang kala terdapat di dalam dokumen asli itu sendiri.

- Terms

Merupakan *single word* dan frasa *multi word* yang terpilih secara langsung dari korpus. Representasi *term-based* dari dokumen tersusun dari subset *term* dalam dokumen.

- Concept

Merupakan fitur yang digenerate dari sebuah dokumen secara manual, *rule-based*, atau metodologi lain.

2.3 Preprosesing Dokumen

Pemrosesan teks merupakan proses menggali, mengolah, mengatur informasi dengan cara menganalisa hubungannya, aturan-aturan yang ada di data tekstual semi terstruktur atau tidak terstruktur. Untuk lebih efektif dalam proses pemrosesan dilakukan langkah transformasi data ke dalam suatu format yang memudahkan untuk kebutuhan pemakai. Proses ini disebut preprosesing dokumen. Setelah dalam bentuk yang lebih terstruktur dengan adanya proses diatas data dapat dijadikan sumber data yang dapat di olah lebih lanjut.

2.3.1 Cleaning

Cleaning adalah proses membersihkan dokumen dari karakter-karakter yang tidak diperlukan untuk mengurangi noise seperti emotikon dan simbol-simbol.

2.3.2 Parsing

Parsing adalah proses untuk memecah teks bebas yang besar menjadi bagian-bagian yang disebut kalimat [CDF-08]. Kalimat-kalimat yang dihasilkan kemudian dipecah lagi menjadi kata-kata melalui proses Casefolding.

2.3.3 Casefolding

Proses *Casefolding* memotong setiap kata dalam teks, dan mengubah semua huruf dalam dokumen menjadi huruf kecil. Hanya huruf ‘a’ sampai ‘z’

yang diterima, sedangkan karakter selain huruf dihilangkan. Hasil dari proses *Casefolding* adalah kata-kata yang merupakan penyusun kalimat [AHA-10].

2.3.4 Filtering/Stopword Removal

Sebuah proses penyaringan untuk menghilangkan kata yang 'tidak relevan' pada hasil Casefolding sebuah dokumen teks dengan cara membandingkannya dengan *Stoplist* (*Stopword list*) yang ada [AHA-10]. Contoh dari *Stopword* misalnya, kata sambung, artikel dan preposisi.

2.3.5 Stemming

Stemming merupakan suatu proses untuk menemukan kata dasar dari sebuah kata dengan menghilangkan semua imbuhan (*affixes*) baik yang terdiri dari awalan (*prefixes*), sisipan (*infixes*), akhiran (*suffixes*) dan *confixes* (kombinasi dari awalan dan akhiran) pada kata turunan. Stemming digunakan untuk mengganti bentuk dari suatu kata menjadi kata dasar dari kata tersebut yang sesuai dengan struktur morfologi Bahasa Indonesia yang baik dan benar [DGT-09]. Stemming yang digunakan pada skripsi ini adalah stemming Arifin-Setiono, yang sudah banyak digunakan untuk proses stemming pada teks berbahasa Indonesia.

2.3.5.1 Stemming Arifin-Setiono

Algoritma ini didahului dengan pembacaan tiap kata dari file sampel. Sehingga input dari algoritma ini adalah sebuah kata yang kemudian dilakukan [ASA-01]:

1. Pemeriksaan semua kemungkinan bentuk kata. Setiap kata diasumsikan memiliki 2 Awalan (*prefiks*) dan 3 Akhiran (*sufiks*). Sehingga bentuknya menjadi:

Prefiks 1 + Prefiks 2 + Kata dasar + Sufiks 3 + Sufiks 2 + Sufiks 1

Seandainya kata tersebut tidak memiliki imbuhan sebanyak imbuhan di atas, maka imbuhan yang kosong diberi tanda ‘x’ untuk *prefiks* dan diberi tanda ‘xx’ untuk *sufiks*.

Pemotongan dilakukan secara berurutan sebagai berikut :

AW : AW (Awalan)

AK : AK (Akhiran)

KD : KD (Kata Dasar)

a. AW I, hasilnya disimpan pada p1

b. AW II, hasilnya disimpan pada p2

c. AK I, hasilnya disimpan pada s1

d. AK II, hasilnya disimpan pada s2

e. AK III, hasilnya disimpan pada s3

2. Setiap tahap pemotongan di atas diikuti dengan pemeriksaan di kamus apakah hasil pemotongan itu sudah berada dalam bentuk dasar. Jika pemeriksaan ini berhasil, proses dinyatakan selesai dan tidak perlu melanjutkan proses pemotongan imbuhan lainnya [ASA-01]. Contoh pemenggalan kata “mempermainkannya”

- Langkah 1 :

Cek apakah kata ada dalam kamus

Ya : Success

Tidak : lakukan pemotongan AW I

Kata = permainkannya

- Langkah 2 :

Cek apakah kata ada dalam kamus

Ya : Success

Tidak : lakukan pemotongan AW II

Kata = mainkannya

- Langkah 3 :

Cek apakah kata ada dalam kamus

Ya : Success

Tidak : lakukan pemotongan AK I

Kata = mainkan

- Langkah 4 :

Cek apakah kata ada dalam kamus

Ya : Success

Tidak : lakukan pemotongan AK II

Kata = main

- Langkah 5 :

Cek apakah kata ada dalam kamus

Ya : Success

Tidak : lakukan pemotongan AK III. Dalam hal ini AK III tidak ada, sehingga kata tidak diubah.

Kata = main

- Langkah 6

Cek apakah kata ada dalam kamus

Ya : Success

Tidak : "Kata tidak ditemukan"

3. Jika sampai pada pemotongan AK III belum juga ditemukan di kamus, dilakukan proses kombinasi. KD yang dihasilkan dikombinasikan dengan imbuhan-imbuhannya dalam dua belas konfigurasi berikut:

a. KD

b. KD + AK III

c. KD + AK III + AK II

d. KD + AK III + AK II + AK I

e. AW I + AW II + KD

f. AW I + AW II + KD + AK III

g. AW I + AW II + KD + AK III + AK II

h. AW I + AW II + KD + AK III + AKII + AKI

i. AW II + KD

j. AW II + KD + AK III

k. AW II + KD + AK III + AK II

l. AW II + KD + AK III + AK II + AK I

Kombinasi a, b, c, d, h, dan l sudah diperiksa pada tahap sebelumnya karena kombinasi ini adalah hasil pemotongan bertahap tersebut. Dengan demikian, kombinasi yang masih perlu dilakukan tinggal 6 yakni pada kombinasi-kombinasi yang belum dilakukan (e, f, g, i, j, dan k). Jika hasil pemeriksaan suatu kombinasi adalah ‘ada’, pemeriksaan pada kombinasi lainnya sudah tidak diperlukan lagi.

Pemeriksaan dua belas kombinasi ini diperlukan karena adanya fenomena *overstemming* pada algoritma pemotongan imbuhan. Kelemahan ini berakibat pada pemotongan bagian kata yang sebenarnya adalah milik kata dasar itu sendiri yang kebetulan mirip dengan salah satu jenis imbuhan yang ada. Pemotongan

yang sudah terlanjur tersebut dapat dikembalikan sesuai posisinya dengan dua belas kombinasi itu.

2.4 Analisis Sentimen

Analisis sentimen yang merupakan bagian dari *opinion mining*, adalah proses memahami, mengekstrak dan mengolah data tekstual secara otomatis untuk mendapatkan informasi [BPL-08]. Dilakukan untuk melihat pendapat terhadap sebuah masalah, atau dapat juga digunakan untuk identifikasi kecenderungan hal di pasar [BPL-02]. Analisis sentimen dalam skripsi ini adalah proses klasifikasi dokumen tekstual ke dalam dua kelas, yaitu kelas sentimen positif dan negatif. Besarnya pengaruh dan manfaat dari analisis sentimen, menyebabkan skripsi ataupun aplikasi mengenai analisis sentimen berkembang pesat, bahkan di Amerika kurang lebih 20-30 perusahaan yang memfokuskan pada layanan analisis sentimen [BGL-10]. Pada dasarnya analisis sentimen merupakan klasifikasi, tetapi kenyataannya tidak semudah proses klasifikasi biasa karena terkait penggunaan bahasa. Dimana terdapat ambigu dalam penggunaan kata, tidak adanya intonasi dalam sebuah teks, dan perkembangan dari bahasa itu sendiri [BGL-10].

Manfaat analisis sentimen sangatlah banyak pada masa sekarang. Banyak manfaat analisis sentimen seperti untuk survey mengenai produk, survey organisasi, survey skripsi dan lain-lain. Skripsi analisis sentimen pada skripsi ini dilakukan dengan menggunakan pendekatan dalam machine learning yang dikenal dengan Metode *K-Nearest Neighbor* dan dikhususkan pada dokumen teks berbahasa Indonesia pada sebuah situs *web*.

2.5 *K-Nearest Neighbor*

K-Nearest Neighbor (KNN) adalah suatu metode pengklasifikasian berdasarkan mayoritas dari kategori. Metode ini bertujuan untuk mengklasifikasi objek baru berdasarkan atribut dan *training sample*. Diberikan suatu titik *query*, selanjutnya akan ditemukan sejumlah K objek atau titik *training* yang paling dekat dengan titik *query*. Nilai prediksi dari *query* akan ditentukan berdasarkan klasifikasi tetanggaan [ITR-10].

Pada metode ini selain memiliki kelebihan seperti tangguh terhadap training data yang *noisy* dan efektif apabila training data berjumlah besar, juga mempunyai beberapa kekurangan diantaranya perlu ditentukan nilai K yang paling optimal yang menyatakan jumlah tetangga terdekat dan biaya komputasi yang cukup tinggi karena perhitungan jarak harus dilakukan pada setia *query instance* secara bersama-sama dengan seluruh instan dari *training sample*.

2.6 KNN *Decision Rule*

Pada proses pengambilan keputusan diperlukan suatu nilai *k* yang akan digunakan untuk memilih kategori yang sesuai. Konsep yang digunakan untuk pengambilan keputusan yaitu dengan mengurutkan hasil perhitungan kemiripan *cosSim*(X,d_j) dimulai dari yang besar. Untuk k=1, dipilih nilai *cosSim*(X,d_j) pada urutan paling atas, dan untuk k>1 dipilih sebanyak k urutan teratas [YYX-09].

Sebagai contoh, terdapat sebuah dokumen X yang akan dikategorikan berdasarkan pada sekumpulan dokumen yang ada pada dokumen latih di d∈T. Misalnya k=1, nilai kemiripan antara X dengan dokumen latih d telah ditentukan, maka dipilih d yang memiliki nilai kemiripan yang paling tinggi. Proses ini dijelaskan pada persamaan 2.1.

$$SIM_{max}(X) = \max_{d \in T} SIM(X, d_j) \quad (2.1)$$

Dimana $SIM_{max}(X)$ adalah nilai kemiripan dokumen X yang paling tinggi. $SIM(X,d_j)$ adalah nilai kemiripan antara dokumen X dengan dokumen latih d. Sedangkan $\max_{d \in T}$ $SIM(X,d_j)$ adalah nilai maksimum kemiripan dokumen X dengan dokumen d yang merupakan bagian dari dokumen latih T [AXB-01].

Jika digunakan k>1, maka penentuan kategorinya adalah dengan menjumlahkan semua nilai kemiripan $SIM(X,d_j)$ yang termasuk dalam suatu kategori. Perhitungan dilakukan dengan persamaan 2.2. Dokumen X masuk ke dalam kategori yang memiliki nilai P(X, C_m) paling besar [YYX-09].

$$p(x, c_m) = \sum_{dj \in KNN\ of\ X} \mathrm{SIM}(\mathrm{X}, \mathrm{d_j}).\, \mathrm{y}(\mathrm{d_j}.\, \mathrm{c_m}) \quad (2.2)$$

Keterangan:

P(x,c_m) : probabilitas dokumen X menjadi anggota kategori c_m

sim(x,d_j) : kemiripan antara dokumen X dengan dokumen latih d_j

$y(d_j.c_m)$: fungsi atribut dari sebuah kategori yang memenuhi

$$y(d_j.c_m) = \begin{cases} 1, d_j \in c_m \\ 0, d_j \notin c_m \end{cases} \quad (2.3)$$

Secara umum, langkah-langkah dari metode *K-Nearest Neighbor* (KNN) adalah sebagai berikut [IGA-11]:

1. Menentukan parameter K (jumlah tetangga paling dekat)
2. Menghitung kuadrat jarak euclidean (*query instance*) masing-masing objek terhadap data sampel yang diberikan
3. Mengurutkan objek-objek tersebut ke dalam kelompok yang mempunyai jarak euclidean terkecil
4. Mengumpulkan kategori klasifikasi *nearest neighbor*
5. Dengan menggunakan kategori *nearest neighbor* yang paling mayoritas maka dapat diprediksikan nilai *query instance* yang telah dihitung.

2.7 Pengukuran Kemiripan (*Cosine Similarity*)

Langkah awal yang dilakukan KNN adalah menghitung nilai kemiripan antara dokumen tes dengan semua data latih. Perhitungan kemiripan dapat dilakukan dengan menghitung jarak antar dokumen.

Dengan dilakukannya perhitungan jarak maka pengukuran kemiripan dapat ditentukan. Pengukuran kemiripan yang dapat digunakan adalah *Cosine Similarity, euclidian distance, kernel function.* Dalam skripsi ini metode pengukuran kemiripan yang digunakan adalah *Cosine Similarity*.

Cosine Similarity telah banyak digunakan untuk menghitung *similarity* dokumen dengan *query* yang diberikan direpresentasikan menggunakan sebuah model vektor jarak (*space vector model*). Didalam mendapatkan nilai *Cosine Similarity* sebelumnya dilakukan perhitungan nilai dot product (*scalar product / inner product*), yaitu perhitungan nilai koordinat sebuah *term* pada sebuah dokumen [GRC-06]. Proses pembentukan koordinat dapat diilustrasikan sebagai berikut:

Misal diberikan $\boldsymbol{d_1}$ sampai $\boldsymbol{d_9}$ sebagai dokumen latih dan X adalah dokumen diolah menjadi vektor berdimensi m, sehingga ruang vektor dokumen X diawali oleh sebuah titik dengan koordinat ($\boldsymbol{X_1 X_2, \ldots, X_m}$). Antara koordinat X

dan masing-masing dokumen $\boldsymbol{d_1}$ sampai $\boldsymbol{d_9}$ akan terbentuk sudut-sudut yaitu $\boldsymbol{\theta_1}$, $\boldsymbol{\theta_2}$, $\boldsymbol{\theta_3}$ dan seterusnya.

Pengukuran *Cosine Similarity* yang ditunjukkan oleh persamaan yang akan digunakan untuk mencari kemiripan antara dokumen X dengan dokumen d pada kumpulan dokumen latih. Nilai kemiripan dihitung dengan persamaan 2.4 [GRC-06].

$$\mathbf{SIM}\ (\mathbf{X}, \boldsymbol{d_1}) = \frac{\sum_{j=1}^{m} \boldsymbol{Xj}\ .\boldsymbol{dij}}{\sqrt{(\sum_{j=1}^{m} \boldsymbol{Xj})^2}.\ \sqrt{(\sum_{j=1}^{m} \boldsymbol{dij})^2}} \qquad (2.4)$$

Nilai yang dihasilkan dari pengukuran *Cosine Similarity* akan berada di antara 0 dan 1, $\mathbf{0} \leq \mathbf{SIM}\ (\mathbf{X}, \boldsymbol{d_i}) \leq \mathbf{1}$. Hal ini menunjukkan bahwa nilai *similarity* tidak dipengaruhi oleh panjang dokumen.

2.8 Pembobotan (*Term Weighting*)

Setelah sebuah dokumen menjalani preprosesing, informasi yang didapat adalah sekumpulan token kata-kata penting. Selanjutnya yang dilakukan adalah mengubah data berupa kata-kata (*term*) tersebut kedalam bentuk numerik sehingga dapat dibaca oleh komputer. Langkahnya adalah dengan melakukan pembobotan pada term yang terdapat pada dokumen.

Metode yang paling umum digunakan untuk melakukan pembobotan terhadap *term* adalah pembobotan *TFIDF*. Metode ini banyak diterapkan dalam pencarian teks (*text retrieval)* dan pemrosesan teks (*text preprocessing)* [RBT-04].

Term Frequency (TF) adalah pembobotan kata (*term*) yang didasarkan pada perhitungan jumlah kata yang muncul pada suatu dokumen . Semakin besar kemunculan suatu kata dalam dokumen akan memberikan nilai kesesuaian yang semakin besar. Sehingga semakin tinggi nilai TF suatu kata pada dokumen, maka semakin besar pula pengaruh kepentingan *term* terhadap dokumen tersebut.

Inverse Document Frequency (IDF) adalah pembobotan kata yang didasarkan pada perhitungan jumlah kata yang muncul pada seluruh dokumen. Dengan perhitungan *IDF* ini artinya semakin sering suatu kata muncul pada dokumen lain maka semakin kecil pula pengaruh kepentingan *term*.

TFIDF merupakan perkalian dari hasil perhitungan *TF* dengan hasil perhitungan *IDF*. Untuk melakukan perhitungan bobot *TFIDF* untuk masing-masing *term* dijelaskan dengan persamaan 2.5 dan persamaan 2.6.

$$\boldsymbol{w = TF\ x\ IDF} \tag{2.5}$$

$$\boldsymbol{w(t,d) = TF\ (t,d)\ x\ log\ \frac{D}{DF_t}} \tag{2.6}$$

dimana:

W(t,d) = bobot *term* t pada dokumen d

TF(t,d) = jumlah kemunculan *term* t dalam dokumen d

D = jumlah seluruh dokumen

$\mathbf{DF_t}$ = jumlah dokumen yang memiliki *term* t

Perhitungan diatas menunjukkan seberapa relevan pengaruh sebuah *term* pada dokumen. *Term* yang sering muncul pada sebagian kecil dokumen cenderung memiliki nilai *TFIDF* yang lebih tinggi dibandingkan dengan *term* yang umumnya muncul pada banyak dokumen.

Algoritma TF-IDF dapat dikembangkan sebagai berikut:

1. Membaca seluruh data template wajah dalam dokumen
2. Menghitung kemunculan setiap *template* dalam dokumen (TF)
3. Menghitung banyaknya dokumen yang mengandung *template* wajah (DF)
4. Menghitung TFIDF