BAB II

LANDASAN TEORI

2.1. Requirements Engineering

Requirementss Engineering adalah proses mengumpulkan kebutuhan, menganalisa, mencocokan dengan kondisi dan kebutuhan *user* atau *client*. *Requirements* pada sebuah sistem tidak timbul secara natural, tetapi dikembangkan dan terus di *review* dan dilakukan revisi (A Van Lamsweerde, 2009). *Requirementss Engineering* menjadi bagian yang paling penting dalam tahap desain dan pengembangan piranti lunak sebagaimana pentingnya untuk membangun piranti lunak yang tepat untuk *customer* (Aurum & Wohlin, 2011).

2.2. User Feedback

User feedback atau tanggapan dari pengguna kepada program piranti lunak yang telah dibuat untuk mengetahui *user experience* yang dapat bermanfaat sebagai masukan ke *developer* untuk pengembangan *feature* yang belum ada atau perbaikan selanjutnya (Pagano & Maalej, 2013). Dalam proses analisa kebutuhan *user feedback* dapat di ekstraksi untuk mendapatkan informasi yang relevan, tanggapan itu dapat digunakan dalam proses menentukan prioritas kebutuhan piranti lunak (Morales-Ramirez et al., 2017)

2.3. Typo Correction

Saat memberikan pesan tanggapan atau kritik dan saran, sering kali seorang pengguna melakukan kesalahan pengetikan atau penulisan. Bahasa yang digunakan oleh pengguna tidak formal atau tidak sesuai dengan penulisan yang terdaftar pada kamus besar bahasa Indonesia, sehingga dapat disimpulkan menjadi penulisan yang salah dan tidak dapat dikenali oleh proses komputer. Kata yang tidak formal atau tidak sesuai dengan kamus besar bahasa Indonesia umumnya pergantian huruf, penyisipan huruf, penghilangan huruf dan penukaran urutan beberapa huruf terdekat.

2.4. Levenshtein Distance

Algoritma *Levenshtein Distance* berguna untuk melakukan perhitungan perbedaan jarak antara *string* yang dimasukan dengan *string target* atau yang ada di kamus kosakata. “Jarak atau *distance* adalah jumlah minimum dari operasi *deletion*, *insert* atau *subtitution* yang dibutuhkan untuk merubah *string source* (s) menjadi *string target* (t) “ (Benisius, 2014). Istilah *source* dan *target* sebenarnya tidak terpaku pada kata pertama atau kedua, hanya sebagai istilah untuk membandingkan posisi atau indeks dua kosakata.

$$\mathrm{lev}_{a,b}(i,j)f(x) = \begin{cases} \max(i,j) & \text{if } \min(i,j) = 0, \\ min \begin{cases} \mathrm{lev}_{a,b}(i-1,j)+1 \\ \mathrm{lev}_{a,b}(i,j-1)+1 \\ \mathrm{lev}_{a,b}(i-1,j-1)+1_{(a_i \neq b_j)} \end{cases} & \text{otherwise,} \end{cases} \quad \dots (2.1)$$

Sedikit contoh sesuai dengan rumus diatas misalkan ‘a’ adalah kata ‘cat’ dan ‘b’ adalah kata ‘cap’ , dengan ‘i’ sebagai indeks atau mewakili posisi karakter pada variabel ‘a’ dan ‘j’ posisi karakter variabel ‘b’. Pada indeks pertama, dimana ‘i’ dan ‘j’ sama-sama diposisi indeks ke-1 dari kata ‘cat’ dan ‘cap’ sama-sama berisi karakter ‘c’ di indeks ke-1 sehingga diambil nilai tertinggi karena sama berarti 0, begitu juga dengan indeks ke-2. Saat pada indeks ke-3 barulah berbeda karakter yaitu ‘p’ dan ‘t’, karena tidak sama maka ambil nilai terkecil dari tiga perbandingan yaitu indeks (3-1, 3) lalu (3, 3-1) serta (3-1, 3-1), lalu nilai dari tiga indeks tadi nilainya ditambah 1, karena belum pernah ada yang berbeda sehingga minimalnya sama menjadi 0+1, jadi lah pada indeks (3,3) berisi nilai 1, yang menandakan bahwa terdapat perbedaan 1 karakter.

Berikutnya juga dapat dilihat langsung simulasi dalam bentuk gambar tabel. Langkah-langkah untuk melakukan perhitungan *Levenshtein Distance* pertama misalkan kata kunci yang digunakan yaitu “unverstas”, lalu dilakukan penyaringan dari kamus kosa kata yang memiliki panjang karakter kata kunci-3, sampai kata kunci+3, panjang karakter kata kuncinya adalah 9, maka akan dilakukan penyaringan dari kamus kosa kata yang memiliki panjang karakter 6 - 12. Dalam contoh ini akan dipakai kata “universitas” yang memiliki panjang 11 karakter, dapat dilihat simulasi matriks perhitungannya pada Gambar 2.1.

		U	N	V	E	R	S	T	A	S
	0	1	2	3	4	5	6	7	8	9
U	1	0	1	2	3	4	5	6	7	8
N	2	1	0	1	2	3	4	5	6	7
I	3	2	1	1	2	3	4	5	6	7
V	4	3	2	1	2	3	4	5	6	7
E	5	4	3	2	1	2	3	4	5	6
R	6	5	4	3	2	1	2	3	4	5
S	7	6	5	4	3	2	1	2	3	4
I	8	7	6	5	4	3	2	2	3	4
T	9	8	7	6	5	4	3	2	3	4
A	10	9	8	7	6	5	4	3	2	3
S	11	10	9	8	7	6	5	4	3	2

Gambar 2.1. Simulasi Matriks Perhitungan

Berdasarkan Gambar 2.1 dapat diketahui bahwa dengan algoritma *Levenshtein Distance* antara kata “unverstas” dan “universitas” menghasilkan nilai jarak akhir 2 (dua) karakter. Sehingga dari kata “unverstas” dapat diubah menjadi kata “universitas”, dan disimpulkan bahwa pengguna bermaksud untuk mengetikan kata “universitas”. Hasil *output* dari algoritma Levenshtein Distance adalah bilangan bulat yang merupakan seberapa besar jarak yang didapatkan dari perhitungan tiga operator yaitu *deletion, insertion* dan *substitution.* Hasil *output* yang masih sensitif dengan panjang karakter dari *string* yang dibandingkan, dan agar *output* lebih spesifik dan mengurangi banyak kata dengan *output* sama, maka digunakan formula untuk normalisasi Levenshtein Distance (Schepens dkk,

2011). Berikut formula normalisasi yang digunakan berdasarkan Schepens dkk (2011).

$$score = \frac{length - distance}{length} \quad \ldots (2.2)$$

Length disini adalah nilai panjang karakter yang paling panjang diantara kedua *string* yang dibandingkan, sedangkan *distance* adalah nilai minimal dari *output* Levenshtein Distance pada kedua *string* yang dibandingkan. Formula ini memperbaiki dan menormalisasi hasil Levenshtein Distance menggunakan panjang maksimum dari kedua *string* yang dibandingkan (Schepens dkk, 2011).

2.5. Naive Bayes Classifiers

Naive Bayes *classifiers* adalah teknik klasifikasi yang bersifat probabilistik yang masih didalam lingkup teknologi *machine learning.* Berlandaskan pada penerapan teorema Bayes yaitu ada asumsi yang kuat mengenai ketergantungan antara pasangan fitur, pada teknik pengklasifikasian ini. Kata didalam teks yang akan diklasifikasi menjadi sebuah atribut dalam Naive Bayes *classifiers (Wagh* dkk, 2018). Pada dasarnya teorema Bayes yang digunakan pada Algoritma Naive Bayes berasumsi naif setiap fitur independen untuk setiap kelas (Tang, 2016). Berikut adalah persamaan dasar teorema Bayes berdasarkan Zhang & Gao (2011).

$$P(A|B) = \frac{P(B|A)\,P(A)}{P(B)} \quad \ldots (2.3)$$

Dalam persamaan diatas dapat dilihat P(A) sebagai probabilitas *prior* (kelas) dan P(A|B) probabilitas *posterior* (kondisional). Atau bisa dimisalkan ingin mengklasifikasi teks atau dokumen *B* kepada suatu kelas bernama *A*. Yaitu dengan menghitung probabilitas *B* terhadap kondisi *A*, lalu dikalikan dengan probabilitas terjadinya *A*, kemudian dibagi dengan probabilitas terjadinya *B*. Naive Bayes *classifier* memiliki sebuah kelebihan utama yaitu penanganan data yang terdistorsi, merata-ratakan estimasi probabilitas bersyarat, fitur yang tidak relevan secara seragam didistribusi serta mengabaikan yang bernilai *null* sehingga tidak berpengaruh secara signifikan terhadap hasil klasifikasi (Ismail dkk, 2016).