BAB II

LANDASAN TEORI

2.1 Chatbot

Sebuah program komputer yang dirancang untuk mensimulasikan sebuah percakapan kepada pengguna (manusia) melalui bentuk teks, suara, dan atau visual dapat disebut sebagai *Chatbot*. Percakapan yang terjadi antara komputer dengan manusia merupakan bentuk respon dari program yang telah dideklarasikan pada *database* program pada komputer. Kemampuan komputer dalam menyimpan banyaknya data tanpa melupakan satu pun informasi yang disimpannya digabungkan dengan kepraktisan bertanya pada sumber informasi langsung dibandingkan dengan mencari informasi sendiri serta kemampuan learning yang dimilikinya menyebabkan chatbot adalah *customer service* yang handal (Hormansyah and Utama, 2018).

Chatbot merupakan salah satu bentuk aplikasi dari *Natural Language Processing*. NLP merupakan salah satu bidang ilmu Kecerdasan Buatan atau dapat disebut sebagai *Artificial Intelligence*. NLP mempelajari komunikasi antara manusia dengan komputer melalui bahasa alami (Kusumadewi, 2003).

Chatbot memiliki 2 komponen utama. 2 komponen utama tersebut yaitu *Chat* dan *Bot*. *Chat* dapat diartikan sebagai pembicaraan sedangkan *Bot* merupakan sebuah program yang mengandung sejumlah data, apabila diberikan masukan dari pengguna maka program tersebut akan memberikan output sebagai jawaban. Untuk menjawab pertanyaan dari pengguna, *chatbot* akan membaca tulisan yang diketikkan oleh pengguna melalui *keyboard* (Adriyani, 2004).

2.2 Pattern-Matching

Pencocokan Pola atau *Pattern Matching* merupakan metode yang digunakan untuk melakukan pencocakan terhadap suatu pola tertentu dengan suatu kumpulan kata atau dapat disebut sebagai *string*. Metode *pattern matching* ini sering digunakan pada Analisis Gambar, Mesin Pencari *Web*, *Editor Teks* serta yang lainnya. Kumpulan dari beberapa karakter yang membentuk suatu kesatuan dapat disebut sebagai *string* (Budiasa, 2009).

2.3 Algoritma Knuth Morris Pratt (KMP)

Algoritma Knuth-Morris-Pratt dikembangkan oleh D. E. Knuth pada tahun 1967, dan J. H. Morris bersama V. R. Pratt pada tahun 1966. Lalu pada tahun 1977, keduanya mempublikasikannya secara bersama. Algoritma Knuth-Morris-Pratt merupakan jenis *Exact String Matching Algorithm*. *Exact String Matching Algorithm* merupakan pencocokan *string* yang dilakukan secara tepat dengan susunan karakter dalam *string* yang memiliki urutan serta jumlah karakter dalam *string* yang sama (Ekaputri, 2006). Algoritma Knuth-Morris-Pratt merupakan pengembangan dari algoritma Brute Force (Rossaria and Susilo, 2015).

Pada algoritma Brute Force, setiap kali ditemukan ketidak cocokan *pattern* dengan teks, maka *pattern* akan digeser satu ke kanan. Berbeda dengan algoritma Knuth-Morris-Pratt. Algoritma Knuth-Morris-Pratt menyimpan informasi dimana informasi tersebut akan digunakan untuk melakukan pergeseran yang lebih jauh, tidak seperti algoritma Brute Force. Sehingga waktu pencarian dapat dikurangi secara signifikan dengan menggunakan algoritma Knuth-Morris-Pratt (Wicaksono, 2007).

Langkah-langkah yang dilakukan oleh algoritma Knuth-Morris-Pratt pada saat melakukan pencocokan *string* yaitu.

1. Algoritma Knuth-Morris-Pratt mulai mencocokkan *pattern* atau pola susunan kata pada awal teks.
2. Algoritma ini akan mencocokkan karakter per karakter *pattern* atau pola susunan kata dengan karakter di teks yang bersesuaian dari kiri ke kanan, sampai salah satu kondisi berikut dipenuhi :
    a. Karakter di *pattern* atau pola susunan kata dan di teks yang dibandingkan tidak cocok (*mismatch*).
    b. Algoritma memberitahukan penemuan di posisi ini apabila semua karakter di *pattern* atau pola susunan kata cocok.
3. Selanjutnya, algoritma menggunakan tabel informasi untuk melakukan pergeseran *pattern* atau pola susunan kata. Lalu algoritma akan mengulangi langkah kedua sampai *pattern* atau pola susunan kata berada di ujung teks.

2.4 Text Pre-Processing

Text pre-processing berfungsi untuk mengubah data yang tidak terstruktur menjadi data yang terstruktur dan data yang terstruktur akan disimpan ke dalam *database*. Tahap *preprocessing* terdiri dari beberapa langkah yaitu *case folding*, *tokenization*, *filtering* dan *stemming* (Rahmawati, Sihwi & Suryani, 2014).

a. Case Folding

Case folding merupakan proses pengubahan semua huruf menjadi huruf kecil yang terdapat dalam teks dokumen (Yavi, 2018).

Tabel 2.1 Contoh Case Folding

Input	Hasil Case Folding
Berapa HARGA Piston yang dijual?	berapa harga piston yang dijual?

b. Tokenizing

Tokenizing merupakan proses pengubahan bentuk kalimat menjadi kata-kata tunggal. *Tokenizing* dapat digunakan untuk menghilangkan tanda baca serta memisahkan kalimat berdasarkan spasi (Saiful, Ransi & Nangi, 2019).

Tabel 2.2 Contoh Tokenizing

Input	Hasil Tokenizing
berapa harga piston yang dijual?	- berapa - harga - piston - yang - dijual

c. Filtering

Filtering merupakan proses penghilangan *stopword* atau kata yang tidak penting misalnya “di”, ”oleh”, “pada”, ”sebuah”, ”karena” dan lain sebagainya (Saiful et al., 2019)(Jumeilah, 2017).

Tabel 2.3 Contoh Filtering

Input	Hasil Filtering
- berapa - harga - piston - yang - dijual	- berapa - harga - piston - dijual

d. Stemming

Stemming adalah proses pengubahan kata yang memiliki imbuhan menjadi kata dasar dengan menghilangkan imbuhan pada kata tersebut (Wahyudi, Susyanto and Nugroho, 2017). Dalam tahap *stemming* diperlukan suatu algoritma yaitu Nazief dan Andriani.

Algoritma Nazief & Adriani sebagai algoritma *stemming* untuk teks berbahasa Indonesia memiliki kemampuan prosentase keakuratan (presisi) yang lebih baik dari algoritma lainnya. Algoritma ini sangat dibutuhkan dan menentukan dalam proses IR (*Information Retrieval*) dalam dokumen Indonesia (Wibowo, 2016).

Algoritma Nazief & Adriani memiliki tahap-tahap sebagai berikut.

1. Cari kata yang akan distem dalam kamus. Jika ditemukan maka diasumsikan kata adalah *root word*. Maka algoritma berhenti.
2. *Inflection Suffixes* (“-lah”, “-kah”, “-ku”, “-mu”, atau “-nya”) dibuang. Jika berupa *particles* (“lah”, “-kah”, “-tah” atau “-pun”) maka langkah ini diulangi lagi untuk menghapus *Possesive Pronouns* (“-ku”, “-mu”, atau “-nya”), jika ada.
3. Hapus *Derivation Suffixes* (“-i”, “-an” atau “kan”). Jika kata ditemukan di kamus, maka algoritma berhenti. Jika tidak maka ke langkah 3a
    a. Jika “-an” telah dihapus dan huruf terakhir dari kata tersebut adalah “-k”, maka “-k” juga ikut dihapus. Jika kata tersebut ditemukan dalam kamus maka algoritma berhenti. Jika tidak ditemukan maka lakukan langkah 3b.

b. Akhiran yang dihapus (“-i”, “-an” atau “kan”) dikembalikan, lanjut ke langkah 4.

4. Hapus *Derivation Prefix*. Jika pada langkah 3 ada sufiks yang dihapus maka pergi ke langkah 4a, jika tidak pergi ke langkah 4b.
   a. Periksa tabel kombinasi awalan-akhiran yang tidak diijinkan. Jika ditemukan maka algoritma berhenti, jika tidak pergi ke langkah 4b.
   b. For i = 1 to 3, tentukan tipe awalan kemudian hapus awalan. Jika *root word* belum juga ditemukan lakukan langkah 5, jika sudah maka algoritma berhenti. Catatan: jika awalan kedua sama dengan awalan pertama algoritma berhenti.
5. Melakukan *Recoding*, yaitu penyusunan kembali kata-kata yang mengalami proses *stemming* yang berlebih.
6. Jika semua langkah telah selesai tetapi tidak juga berhasil maka kata awal diasumsikan sebagai *root word*. Proses selesai.

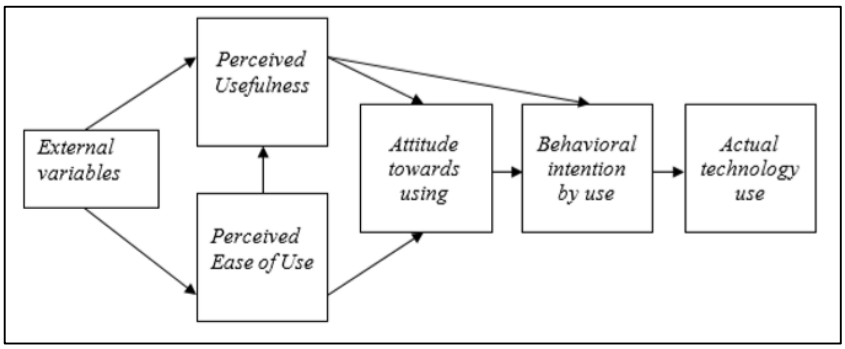
Tabel 2.4 Contoh Stemming

Input	Hasil Stemming
- membeli - menghargai - melampaui - dijual	- beli - harga - lampau - jual

2.5 Technology Acceptance Model (TAM)

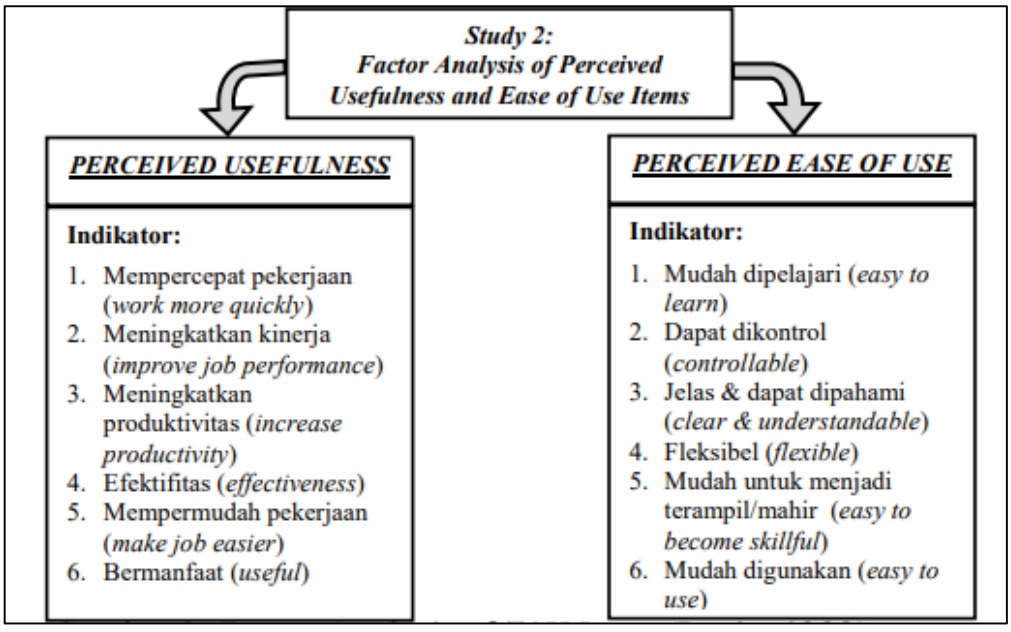
Technology Acceptance Model (TAM) merupakan adaptasi dari *Theory of Reasonede Action Model* (TRA) yang diperkenalkan oleh Fred D. Davis pada tahun 1989. TAM menjelaskan bagaimana penerimaan individu terhadap penggunaan teknologi dan reaksi pengguna ketika dihadapkan dengan sebuah teknologi baru yang digunakan. Menurut Davis (1989), TAM memiliki dua sisi yaitu sisi pertama

atau yang biasa disebut believes yang terdiri atas *percieved usefulness* dan *percieved easy of use* dan sisi yang kedua terdiri dari *attitude, behavior intention to use* dan *usage behavior* (Davis, 1989).



Gambar 2.1 Model TAM (Davis, 1989)

Ketika pengguna menggunakan sistem informasi yang baru, TAM menggunakan 2 faktor yang memengaruhinya yaitu *perceived ease of use* dan *perceived usefulness*. *Perceived ease of use* mengukur sejauh mana orang percaya bahwa menggunakan sistem tertentu akan bebas dari usaha. Sedangkan *Perceived usefulness* mengukur sejauh mana orang percaya bahwa menggunakan sistem tertentu akan meningkatkan kinerja pekerjaannya (Endang Fatmawati, 2015).



Gambar 2.2 Factor Analysis of Perceived Usefulness and Ease of use Items (Davis, 1989)

2.6 Skala Likert

Skala Likert atau *Likert Scale* adalah skala penelitian yang digunakan untuk mengukur sikap, pendapat dan persepsi seseorang atau sekelompok orang tentang fenomena social (Sugiyono, 2014). Skala Likert dapat digunakan dengan cara mengajukan beberapa pertanyaan kepada responden dan responden diminta memberikan pilihan jawaban atau respon dalam skala ukur yang disediakan. Jawaban dari setiap pertanyaan yang menggunakan skala Likert mempunyai gradasi dari sangat positif sampai sangat negatif, yang berupa kata-kata yaitu sangat setuju (SS), setuju (s), tidak memutuskan (N), tidak setuju (TS), dan sangat tidak setuju (STS).

Skala Likert menggunakan skala dengan interval 1 sampai 5 yang dapat dilihat pada tabel 2.5.

Tabel 2.5 Tabel Skala Likert (Sugiyono, 2014)

Keterangan	Nilai (Skor)	Persentase
Sangat setuju	5	81% - 100%
Setuju	4	61% - 80%
Netral	3	41% - 60%
Tidak setuju	2	21% - 40%
Sangat tidak setuju	1	0% - 20%

Berdasarkan Tabel Skala Likert, rumus yang digunakan untuk menghitung persentase skor dapat dilihat pada rumus 2.1.

Persentase skor = (1 x Sangat Tidak Setuju) + (2 x Tidak Setuju) + (3 x Netral) + (4 x Setuju) + (5 x Sangat Setuju) / (5 x Jumlah Responden) x 100% ... (2.1)

2.7 Black Box Testing

Pengujian *Black Box* juga disebut dengan pengujian fungsional, teknik pengujian yang dirancang berdasarkan informasi dari spesifikasi. Pengujian ini tidak memperhatikan mekanisme internal dari sistem, namun hanya terfokus pada hasil yang dihasilkan dari input dan hasil eksekusinya (Nidhra, 2012).

Pengujian *Black Box* digunakan untuk menemukan kesalahan dalam beberapa kategori diantaranya (Pare, 2013).

1. Fungsi-fungsi yang salah atau hilang.
2. Kesalahan *interface*.
3. Kesalahan dalam struktur data atau akses *database external*.
4. Kesalahan peforma.
5. Kesalahan inisialisasi dan terminasi.