BAB III

METODOLOGI DAN PERANCANGAN APLIKASI

3.1 Metode Penelitian

Metode penelitian yang digunakan untuk merancang dan membangun LINE *chatbot* sebagai media informasi suku cadang mobil menggunakan algoritma Knuth-Morris-Pratt yaitu sebagai berikut.

3.1.1 Studi Literatur

Studi literatur dilakukan dengan mempelajari berbagai teori-teori baik dari jurnal, artikel, buku, dan sumber-sumber lainnya mengenai LINE *chatbot*, algoritma Knuth-Morris-Pratt dan literatur-literatur yang lainnya yang berhubungan dengan aplikasi yang akan dirancang dan dibangun.

3.1.2 Analisis Kebutuhan

Analisis kebutuhan dilakukan dengan menganalisa kebutuhan berdasarkan permasalahan yang terjadi dan mengumpulkan data-data yang diperlukan dalam pembuatan sistem dengan melakukan wawancara terhadap pemilik toko Felix Permai Motor.

3.1.3 Perancangan Sistem

Perancangan sistem dilakukan dengan membuat *flowchart* serta struktur tabel untuk pembangunan sistem.

3.1.4 Pembangunan Sistem

Setelah perancangan sistem, pembangunan sistem dilakukan dengan membuat LINE *chatbot* sesuai perancangan yang sudah dibuat dengan menggunakan bahasa pemrograman PHP.

3.1.5 Pengujian Sistem

Pengujian sistem dilakukan dengan menggunakan *Black Box testing*. Pengujian *Black Box* digunakan untuk menemukan fungsi-fungsi yang salah dan mengetahui apakah hasil keluaran dari sistem berjalan sesuai dengan fungsinya.

3.1.6 Evaluasi

Evaluasi dilakukan untuk mengetahui tingkat kepuasan pengguna terhadap sistem. Menurut Roscoe dalam Sugiono (2013), ukuran sampel yang layak dalam penelitian adalah antara 30 sampai dengan 500. Sistem akan dicoba oleh pengguna. Setelah itu, pengguna akan diberikan kuesioner dalam bentuk *google form* untuk mendapat umpan balik terhadap sistem yang telah dibangun. Jenis kuesioner yang digunakan adalah kuesioner tertutup, yaitu kuesioner yang sudah disediakan jawabannya dengan tujuan untuk mempermudah pengguna dalam memberikan jawaban. Kuesioner menggunakan metode *Technology Acceptance Model* dengan skala Likert. Hasil evaluasi sistem diperoleh berdasarkan perhitungan dari skala Likert.

3.2 Perancangan Aplikasi

Proses perancangan aplikasi dimulai dengan membuat *flowchart* serta struktur tabel.

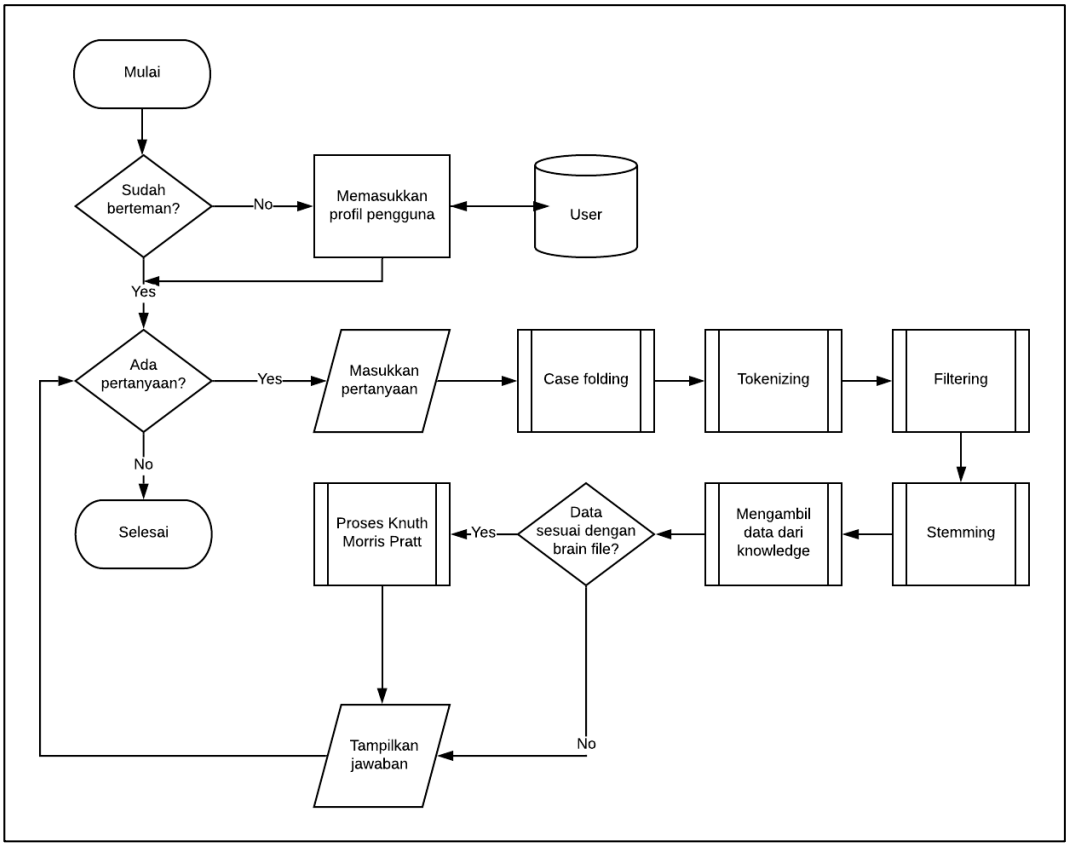
3.2.1 Flowchart Umum

Untuk memulai percakapan dengan *chatbot*, pengguna harus berteman terlebih dahulu dengan *chatbot*. Saat pengguna berteman dengan *chatbot*, data profil pengguna akan disimpan ke tabel user yang ada di dalam *database* dan pengguna dapat memasukkan *input*. Jika ada *input* dari pengguna, *input* tersebut akan menjalani proses *case folding* yang bertujuan untuk mengubah semua huruf yang ada di dalam *input* pengguna menjadi huruf kecil. *Input* yang sudah diproses di tahap *case folding*, akan melewati proses *tokenizing* di mana kata-kata yang ada di dalam kalimat akan dipecah berdasarkan spasi dan memeroleh kata-kata dalam bentuk *array* serta menghilangkan simbol-simbol, tanda baca yang tidak diperlukan.

Setelah mendapatkan hasil dari proses *tokenizing*, kata-kata yang sudah berada dalam bentuk *array* akan melewati proses *filtering*. Proses *filtering* akan memeriksa apakah ada kata-kata yang tidak penting dari setiap *array*. Jika ada, maka kata tersebut akan dibuang sehingga mendapatkan teks yang lebih sederhana. Selanjutnya, hasil dari proses *filtering* akan menjalani proses *stemming* untuk mencari kata dasar dengan menggunakan algoritma Nazief & Adriani.

Setelah mendapatkan hasil dari proses *stemming*, hasil tersebut akan digunakan untuk mengambil data-data dari *database*. Jika tidak terdapat data yang sesuai, *chatbot* akan memberikan jawaban bahwa pertanyaan yang di-*input* tidak berhubungan dengan suku cadang mobil. Namun, jika terdapat data yang sesuai, *chatbot* akan melakukan proses Knuth Morris Pratt. Pada proses ini, hasil *stemming* akan dipecah menjadi elemen-elemen *array* berdasarkan spasi lalu kata-kata dari setiap elemen *array* dan kata kunci yang didapat dari *database* akan dipecah per

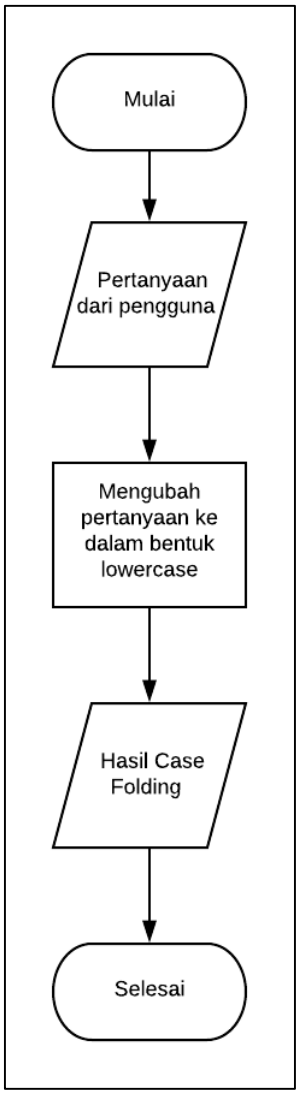
karakter untuk melakukan pencocokan pola dan memberikan skor pada setiap *index* kata kunci. *Index* kata kunci yang memiliki skor tertinggi akan menampilkan jawaban ke pengguna. *Flowchart* umum dapat dilihat pada gambar 3.1.



Gambar 3.1 Flowchart umum

3.2.2 Flowchart Case Folding

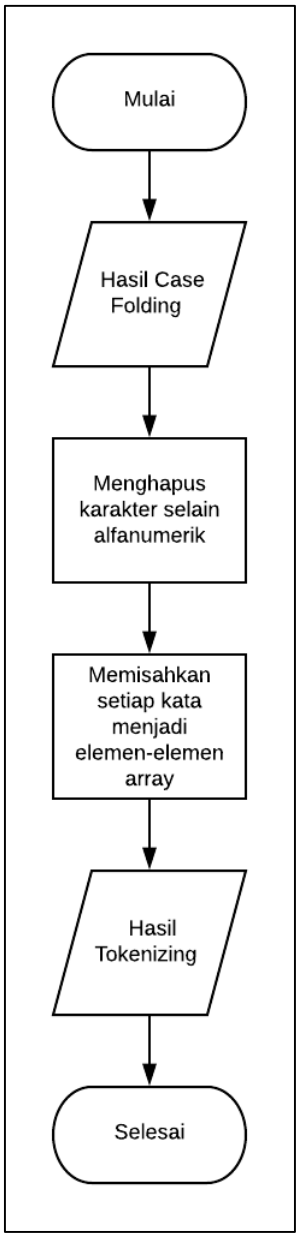
Proses *case folding* merupakan proses yang digunakan untuk mengubah semua huruf dalam suatu teks menjadi huruf kecil. Alur proses seperti pada Gambar 3.2



Gambar 3.2 Flowchart case folding

3.2.3 Flowchart Tokenizing

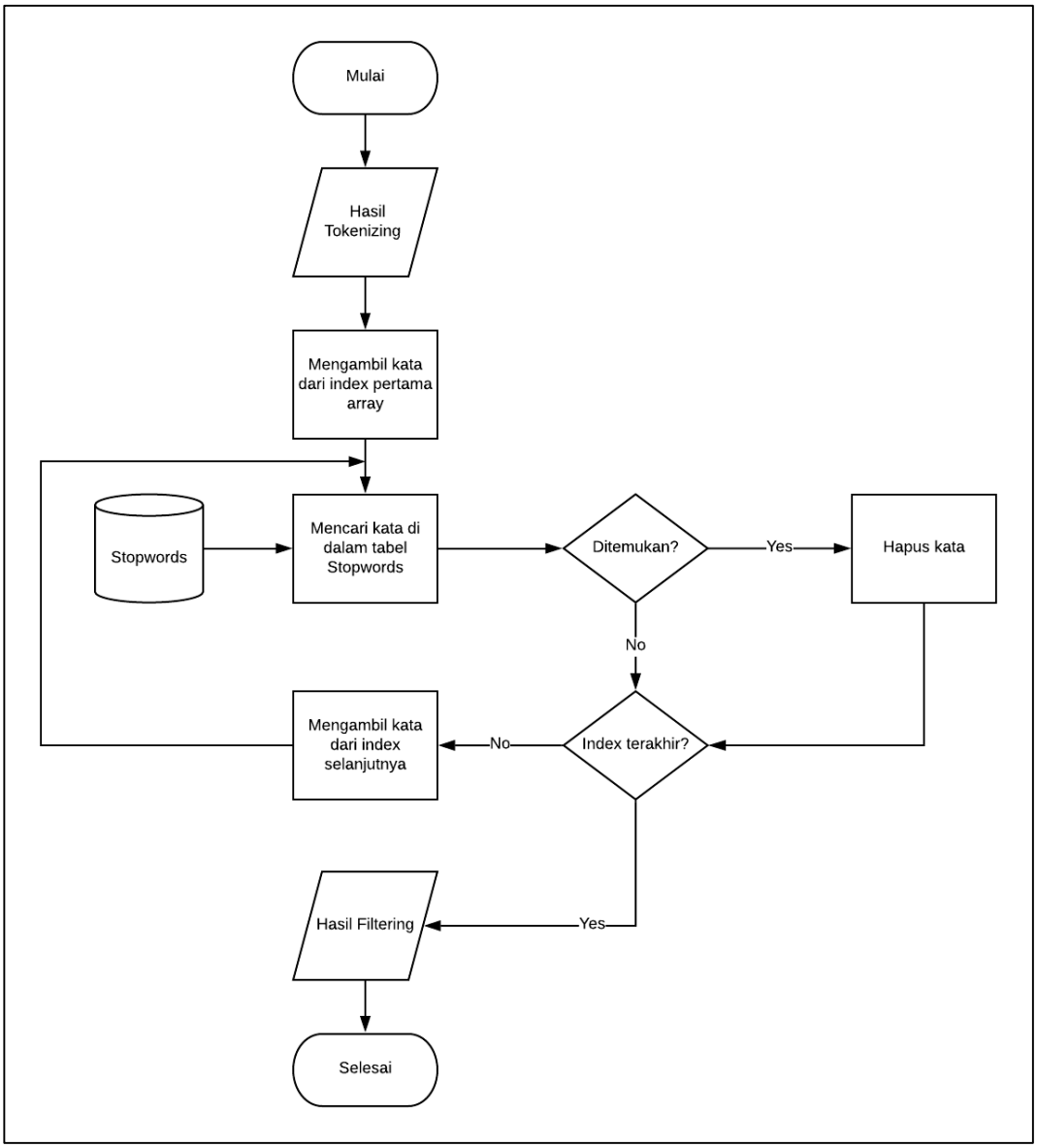
Proses *tokenizing* berfungsi untuk menghilangkan karakter selain alfanumerik dan memecah kalimat menjadi kata-kata ke dalam bentuk *array* berdasarkan spasi. Alur proses seperti pada gambar 3.3



Gambar 3.3 Flowchart tokenizing

3.2.4 Flowchart Filtering

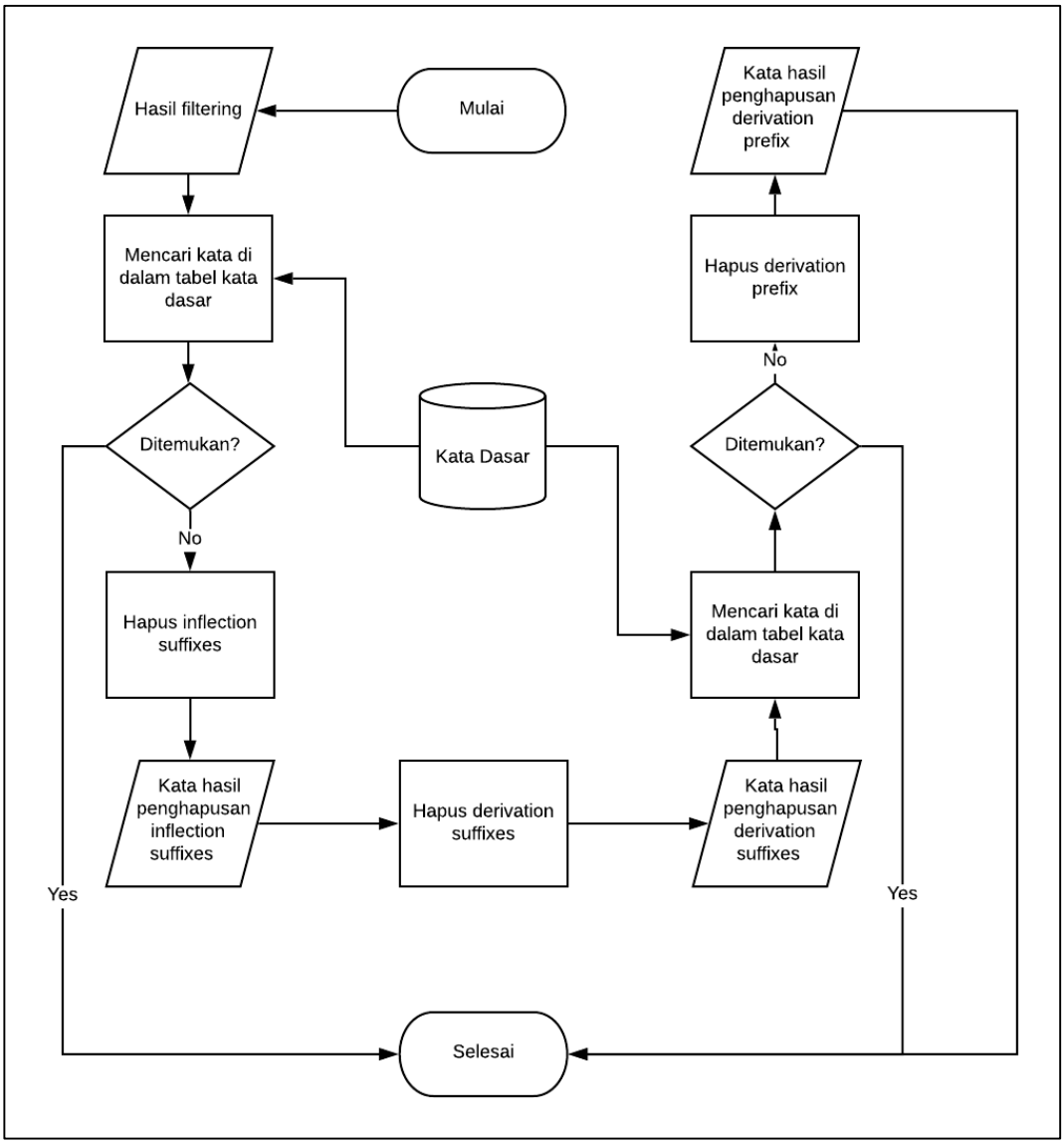
Proses Filtering berfungsi untuk menghilangkan *stopword* supaya mendapatkan kalimat yang lebih sederhana. Alur proses seperti pada gambar 3.4



Gambar 3.4 Flowchart filtering

3.2.5 Flowchart Stemming

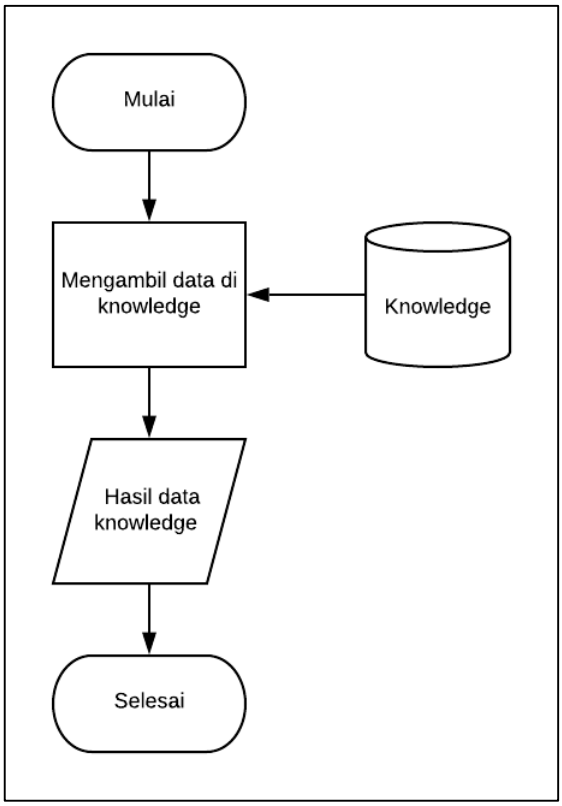
Pada proses *stemming*, hasil *filtering* akan diproses untuk mencari kata dasar. *Stemming* ini menggunakan algoritma Nazief & Adriani. Pertama kali kata dari hasil *filtering* akan diperiksa di dalam tabel kata dasar. Apabila kata ditemukan di dalam tabel kata dasar, maka kata tersebut merupakan kata dasar sehingga algoritma dihentikan. Jika kata tidak ditemukan di dalam kamus, maka diperiksa apakah ada *suffix* (“-lah”, “-kah”, “-ku”, “-mu”, atau “-nya”). Jika ditemukan maka *suffix* akan dihilangkan. Setelah itu, kata akan diperiksa apakah ada *suffix* (“-i”, “-an” atau “-kan”). Jika ditemukan maka *suffix* akan dihilangkan lalu akan diperiksa di dalam tabel kata dasar untuk mengetahui apakah kata tersebut merupakan kata dasar. Jika tidak ditemukan, maka kata akan diperiksa apakah ada prefix (“di-”, “ke-”, “se-”, “te-”, “be-”, “me-”, atau “pe-”). Kata tersebut akan diasumsikan sebagai kata dasar walaupun semua langkah yang dilalui selesai dan tidak ada yang berhasil.



Gambar 3.5 Flowchart Stemming

3.2.6 Flowchart Pengambilan Data di Knowledge

Pada proses pengambilan data pada knowledge, hasil *stemming* menggunakan algoritma Nazief & Adriani akan digunakan untuk mengambil data dari tabel knowledge yang ada di *database*.

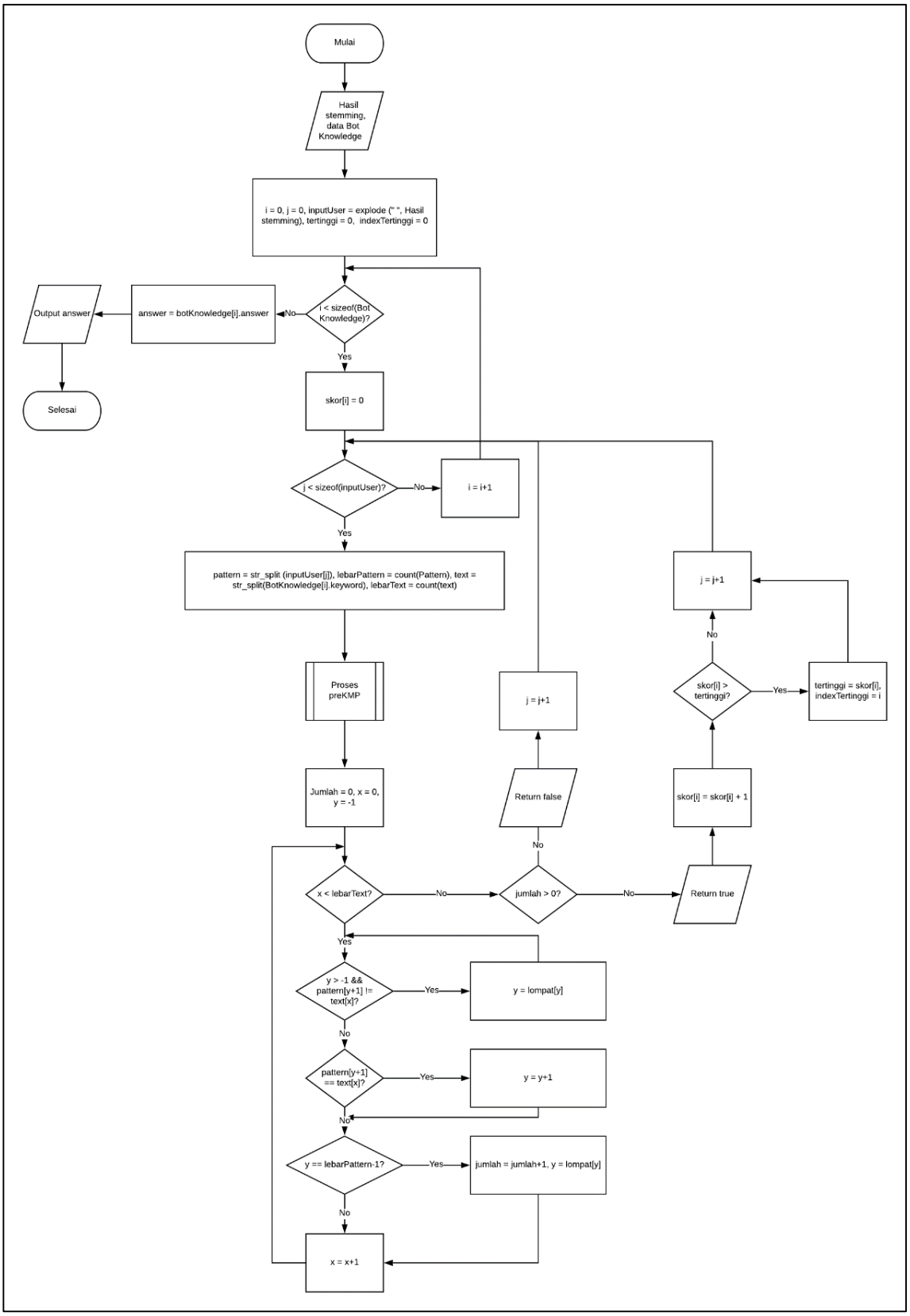


Gambar 3.6 Flowchart pengambilan data knowledge

3.2.7 Flowchart Knuth Morris Pratt

Knuth Morris Pratt merupakan algoritma pencocokan *string* untuk mencari teks berdasarkan urutan dari kiri ke kanan dan pergeserannya ditentukan oleh tabel fungsi pinggiran. Pada proses ini, hasil dari stemming akan dipecah berdasarkan spasi dan disimpan ke dalam *variable* inputUser dalam bentuk *array*. Setelah itu hasil yang diperoleh dari *knowledge* akan diambil satu per satu. Selama masa pengulangan sebanyak *size* dari knowledge yang didapat, dilakukan pencocokan pola antara setiap kata kunci dari *knowledge* yang didapat dengan hasil stemming yang sudah dipecah. Sebelum melakukan pencocokan, *String* pada kata kunci dan *string* pada inputUser akan diubah ke dalam bentuk *array of characters*. *String* kata kunci yang sudah diubah ke dalam bentuk *array of characters* dimasukkan ke *variable text* dan *String* inputUser dimasukkan ke *variable pattern*. *Pattern* akan

masuk ke proses preKMP terlebih dahulu untuk mendapatkan tabel fungsi pinggiran dimana tabel fungsi pinggiran yang didapat akan digunakan sebagai informasi pergeseran saat melakukan pencocokan *string*. Selanjutnya, *pattern* yang sudah mendapatkan tabel fungsi pinggiran akan melakukan pencocokan dengan *text*. *Pattern* akan melakukan pencocokan karakter per karakter sampai akhir *text*. Jika karakter yang ditemukan serupa, maka pergeseran akan bergeser sesuai tabel fungsi pinggiran. Jika tidak serupa, maka pergeseran akan bergeser satu karakter ke kanan. Apabila pattern yang dicari ditemukan di dalam text, maka akan melakukan penambahan skor pada *index knowledge* yang sudah melakukan pencocokan *string* dan variable indexTertinggi akan menyimpan posisi kata kunci yang memiliki skor tertinggi. Setelah semua kata kunci sudah melakukan pencocokan *string*, maka kata kunci yang memiliki skor tertinggi akan menampilkan jawabannya.

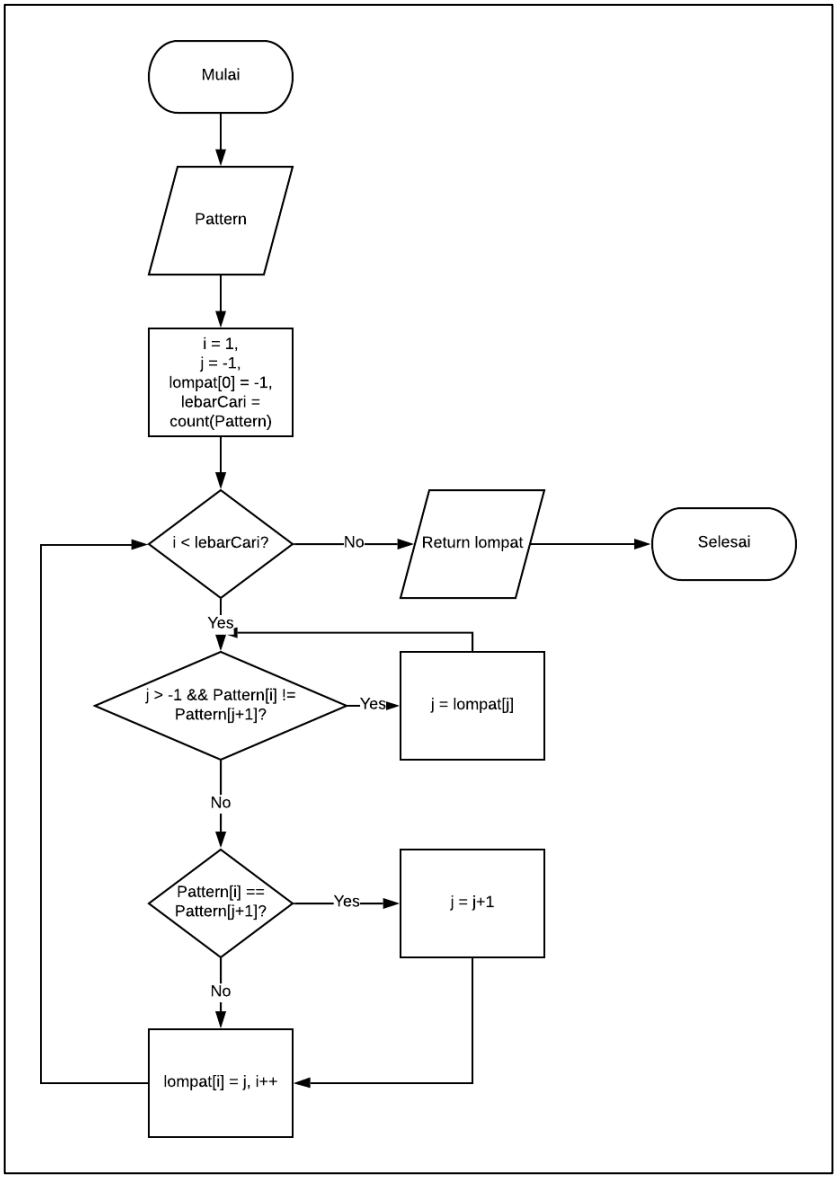


Gambar 3.7 Flowchart proses KMP

3.2.8 Flowchart proses preKMP

Pada proses ini, tabel fungsi pinggiran dibuat untuk menentukan pergeseran pencocokan *string* pada *input* pengguna dengan *string* pada kata kunci dari knowledge yang didapat. Fungsi preKMP membutuhkan satu parameter yaitu pattern atau *input* pengguna untuk membuat tabel fungsi pinggiran.

Pattern yang sudah dipecah menjadi *array of characters* akan menyimpan daftar nilai pinggiran selama iterasi belum melebihi jumlah karakter *pattern*.



Gambar 3.8 Flowchart preKMP

3.2.9 Struktur Tabel

1. Nama Tabel : Users

   Fungsi : menyimpan data *pengguna* yang sudah berteman dengan chatbot.

Tabel 3.1 Tabel Users

Nama Field	Tipe Data	Keterangan
id	int(11)	Primary key
user_id	varchar(100)	Id user yang dikirim dari Line
display_name	Varchar(100)	Nama Line user

2. Nama Tabel : Knowledge

   Fungsi : menyimpan kata kunci pertanyaan berserta jawaban sebagai pengetahuan dasar yang dimiliki chatbot.

Tabel 3.2 Tabel Knowledge

Nama Field	Tipe Data	Keterangan
id	int(11)	Primary key
keyword	varchar(200)	Kata-kata kunci pertanyaan
answer	varchar(200)	Jawaban dari pertanyaan
amount	varchar(200)	Jumlah uang atau stok

3. Nama Tabel : Stopwords

   Fungsi : menyimpan kata-kata *stopwords*.

Tabel 3.3 Tabel Stopwords

Nama Field	Tipe Data	Keterangan
id	int(11)	Primary key
stopword	varchar(100)	Kata stopword

4. Nama Tabel : tb_katadasar

   Fungsi : menyimpan kata-kata dasar.

Tabel 3.4 Tabel Kata Dasar

Nama Field	Tipe Data	Keterangan
id_ktdasar	int(11)	Primary key
katadasar	varchar(20)	Kata dasar
tipe_katadasar	Varchar(20)	Tipe kata dasar