BAB III
METODE PENELITIAN

3.1 Subjek dan Objek Penelitian

Subjek yang digunakan dalam penelitian ini adalah suatu dokumen yang berisi kata dasar dalam Bahasa Indonesia yang digunakan sebagai acuan dalam melakukan koreksi. Sedangkan objek yang digunakan dalam penelitian adalah suatu teks/kalimat yang menggunakan bahasa Indonesia. Kalimat ini akan dideteksi apakah ada kesalahan ketik atau tidak. Jika terdapat kesalahan ketik, maka kesalahan ketik tersebut akan dikoreksi.

3.2 Alat dan Bahan

Alat penelitian yang digunakan adalah *hardware* dan *software* yang didukung dengan *library* yang dibutuhkan dalam penelitian. Dan bahan penelitian yang digunakan adalah kalimat berbahasa Indonesia.

3.2.1 Alat Penelitian

Perangkat keras (*hardware*) dan spesifikasi yang digunakan dalam melakukan penelitian ini adalah sebagai berikut :

1. Device : HP-14 BS0XX
2. *Processor* : Intel(R) Core(TM) i3-6006U CPU @ 2.00GHz 1.99 GHz
3. RAM : 4,00 GB
4. Layar : 14 Inchi

Penelitian ini membutuhkan beberapa *software* dan beberapa *library* yang harus diinstall diantaranya adalah sebagai berikut :

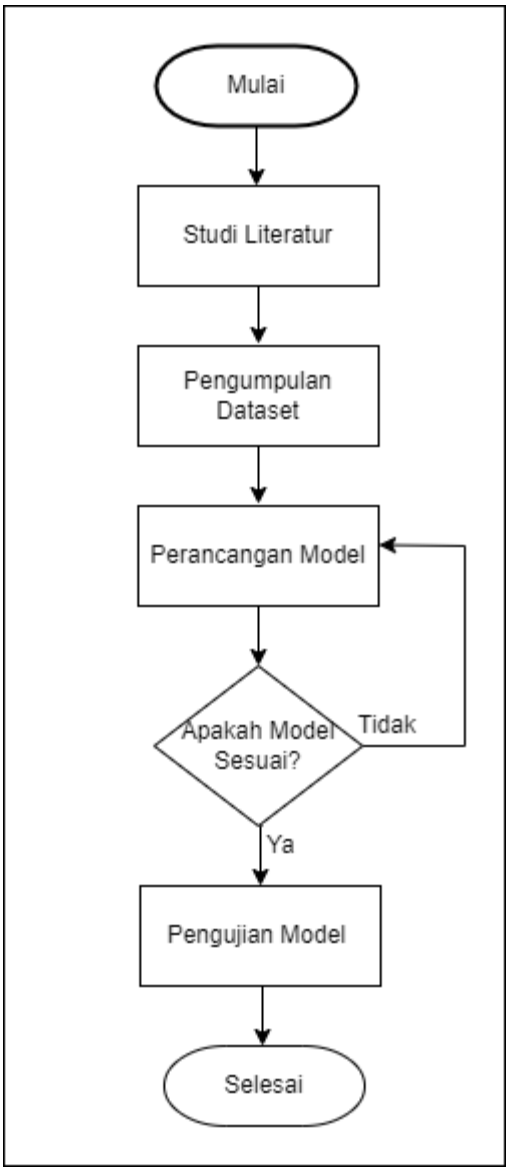
1. Sistem operasi : Windows 10
2. *Python* 3.7
   a. *Anaconda*
   b. *Jupyter notebook*
3. *Excel*
4. *Notepad++*

3.2.2 Bahan Penelitian

Bahan yang digunakan dalam penelitian ini adalah kumpulan kata dasar yang ada dalam KBBI yang dikumpulkan ke dalam dokumen dengan format .csv.

3.3 Tahapan Penelitian

Penelitian ini dibuat melalui beberapa tahapan proses, seperti ditunjukkan pada Gambar 3.1.



Gambar 3.1 Tahapan Penelitian

3.3.1 Studi Literatur

Langkah awal dalam memulai perancangan model ini adalah dengan melakukan studi literatur. Studi literatur dapat dilakukan dengan membaca dan mencatat referensi dari berbagai sumber yang ada, dimana dalam perancangan ini penulis memperoleh referensi dari berbagai sumber yang

terkait dengan penelitian yang dilakukan, diantaranya yaitu : jurnal, skripsi, ataupun *website*. Dari sumber-sumber tersebut didapatkan studi literatur mengenai teori-teori mengenai hal-hal yang terkait dengan penelitian, selain itu didapat pula cara kerja dan perhitungan dari metode yang digunakan.

3.3.2 Pengumpulan Data

3.3.2.1 KBBI (Kamus Besar Bahasa Indonesia)

Pengumpulan data dilakukan dengan mengumpulkan kata dasar yang terdapat dalam Kamus Besar Bahasa Indonesia (KBBI). Kata-kata tersebut dikumpulkan di dalam suatu dokumen dengan format .csv untuk nantinya dijadikan sebagai dataset dalam mencocokkan kata saat proses koreksi kata. Dataset yang digunakan dalam perancangan model ini berjumlah sebanyak 36.5 kata.

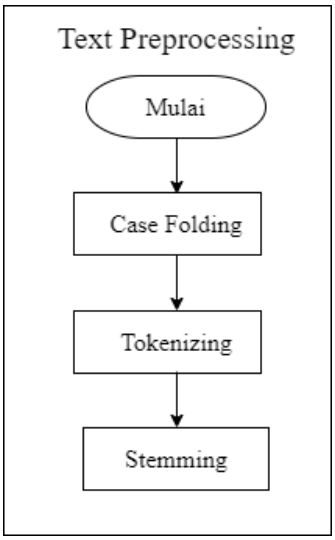
3.3.2.2 Kumpulan Novel

Dataset yang kedua berupa kumpulan novel yang digunakan sebagai model dari *FastText* dan RNN. Dataset ini dikumpulkan dalam suatu dokumen dengan format .txt dengan ukuran file sebesar 11.309 kb. Korpus novel yang digunakan untuk dataset RNN akan dipotong menjadi 8 potongan kata. Terdapat dua dataset pada model RNN yaitu dataset asal dan dataset tujuan.

3.3.3 Perancangan Model

Pada penelitian ini akan dibuat empat model, setiap model mempunyai tahapan perancangan yang berbeda-beda sesuai dengan cara kerja setiap algoritmanya. Namun, pada setiap model akan dilakukan tahap *preprocessing*. Tahap *preprocessing* adalah tahap menyederhanakan teks agar mudah diproses oleh komputer. *Preprocessing* dilakukan pada kalimat

inputan yang akan dicek apakah ada kesalahan ketik atau tidak. Pada penelitian ini tahap *preprocessing* dibagi menjadi beberapa tahapan, yaitu :



Gambar 3.2 Tahapan *Preprocessing*

Gambar 3.2 adalah gambar dari tahapan *preprocessing*. Berikut adalah penjelasan setiap tahapannya :

a. *Case Folding*

Case Folding adalah mengubah kalimat inputan menjadi huruf kecil atau *lowercase*, selain itu pada proses ini seluruh karakter selain huruf akan dihilangkan. Berikut pada tabel 3.1 adalah contoh dari tahapan *case folding* :

Tabel 3. 1 Contoh dari *Case Folding*

Sebelum *Case Folding*	**Setelah *Case Folding***
kemarin Ibu memasak rendang	kemarin ibu memasak rendang
adik sedang tidur, sedangkan aku sedang belajar	adik sedang tidur sedangkan aku sedang belajar
Dimanakah ibukota Inggris?	Dimanakah ibukota inggis

b. *Tokenizing*

Tokenizing adalah tahapan memotong kalimat *input*an setiap satuan kata nya atau sederhananya adalah proses pemecahan kalimat menjadi kata. Tabel 3.2 dibawah ini adalah contoh dari tahap *tokenizing* :

Tabel 3. 2 Contoh dari *Tokenizing*

Sebelum Tokenizing	**Setelah Tokenizing**
hobi saya membaca buku	[‘hobi’, ‘saya’, ‘membaca’, ‘buku’]
semalam saya tidur terlambat	[‘semalam’, ‘saya’, ‘tidur’, ‘terlambat’]

c. *Stemming*

Stemming adalah proses mengubah setiap kata menjadi kata dasar yang akan dicocokkan dengan kata dasar yang ada di kamus. Tahap *stemming* ini hanya dilakukan pada model *Levenshtein distance* dan *Soundex* saja, untuk model RNN dan *FastText* tidak dibutuhkan tahap *stemming* karena tahap *stemming* akan menghilangkan sebagian makna pada kalimat *input*an. Tabel 3.3 berikut ini adalah contoh dari tahapan *stemming* :

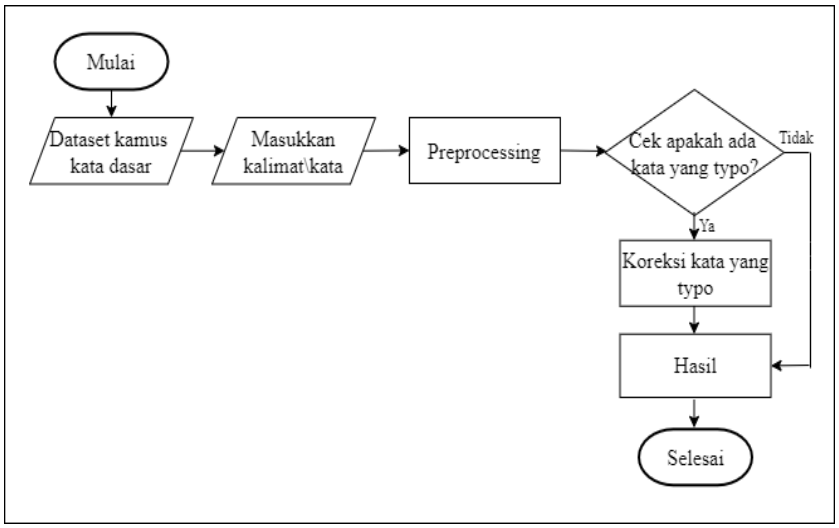
Tabel 3. 3 Contoh dari *Stemming*

Sebelum *Stemming*	**Setelah *Stemming***
Memasak	Masak
Mengoreksi	Koreksi
Dimakan	Makan

Berikut adalah tahapan pemodelan dan cara kerja dari algoritma yang digunakan dalam penelitian ini :

3.3.3.1 Levenshtein Distance

Dalam mengoreksi sebuah kalimat atau kata, algoritma *Levenshtein Distance* akan melakukan pencocokan antara kata yang salah dengan kata dasar atau datasetnya. Dari pencocokan itulah akan diketahui jarak kemiripan paling dekat antara kata yang salah dengan suatu kata di dalam datasetnya. Kata dalam dataset yang menunjukkan jarak kemiripan paling dekat dengan kata yang salah akan dijadikan kata koreksi. Berikut adalah tahapan alur dari Levenshtein Distance :



Gambar 3.3 Tahapan perancangan model Levenshtein Distance

Gambar 3.3 adalah diagram alur untuk mengoreksi kata yang salah ketik menggunakan *Levenshtein Distance*. Langkah pertama yang harus dilakukan adalah *import* dataset kamus kata dasar. Setelah itu, *input* kan suatu kalimat dimana kalimat tersebut akan dikoreksi apakah ada kata yang tidak cocok dengan yang ada di kamus, sebelum ke tahap selanjutnya kata tersebut harus melalui proses *preprocessing* yaitu *case folding*, *tokenizing*, dan *stemming*. Lalu cek apakah terdapat yang *typo* atau tidak, jika ada

maka akan dikoreksi dengan menggunakan aturan *Levenshtein Distance*, jika tidak ada kata yang *typo* maka hasil koreksi nya adalah kalimat inputan awal.

Untuk menghitung jarak *Levenshtein Distance* dapat menggunakan tiga operasi, yaitu : operasi penyisipan karakter (*insertion*), operasi penghapusan karakter (*deletion*), dan operasi penukaran karakter (*substitution*). Berikut adalah contoh perhitungan algoritma *Levenshtein Distanc*e menggunakan operasinya antara kata "nasi" dan "nasa" :

Nas**i** ⟶ Nas**a** (Penukaran "i" ke "a")

Dari perhitungan diatas dapat disimpulkan bahwa jarak *Levenshtein* antara "Nasi" dan "Nasa" adalah 1, karena diperlukan 1 pengeditan untuk mengubah kata "Nasi" dan "Nasa". Selain menggunakan cara operasi, algoritma *Levenshtein Distance* juga dapat dihitung dengan menggunakan rumus pada Persamaan (3.6)

$$lev_{a,b}(i,j) = \begin{cases} \max(i,j) & \text{if min(i,j)=0,} \\ min\begin{cases} lev_{a,b}(i-1,j)+1 \\ lev_{a,b}(i,j-1)+1 \\ lev_{a,b}(i-1,j-1)+1_{a_i \neq b_j} \end{cases} & \text{otherwise} \end{cases} \quad (3.6)$$

a = *String* 1

b = *String* 2

a_i = Posisi karakter *string* 1

b_j = Posisi karakter *string* 2

Untuk menghitung *Levenshtein Distance* dengan menggunakan rumus diatas, sebelumnya kita harus menginisialisasi a dan b terlebih dahulu.

a = nasa, i = 1

b = nasi, j = 1

Tabel 3 1. Tabel Perhitungan *Levenshtein Distance*

		N	**A**	**S**	**I**
	0	**1**	**2**	**3**	**4**
N	**1**	0	1	2	3
A	**2**	1	0	1	2
S	**3**	2	1	0	1
A	**4**	3	2	1	1

- $lev_{a,b}(1,1) = \begin{cases} \max(i,j) \\ min\begin{cases} lev_{a,b}(0,1)+1 \\ lev_{a,b}(1,0)+1 \\ lev_{a,b}(0,0) \end{cases} \end{cases} = min\begin{cases} 1+1=2 \\ 1+1=2 \\ 0 \end{cases}$

  $lev_{a,b}(1,1) = 0$

- $lev_{a,b}(1,2) = \begin{cases} \max(i,j) \\ min\begin{cases} lev_{a,b}(0,2)+1 \\ lev_{a,b}(1,1)+1 \\ lev_{a,b}(0,1)+1_{a_i \neq b_j} \end{cases} \end{cases} = min\begin{cases} 2+1=3 \\ 0+1=1 \\ 1+1=2 \end{cases}$

  $lev_{a,b}(1,2) = 1$

  ⋮

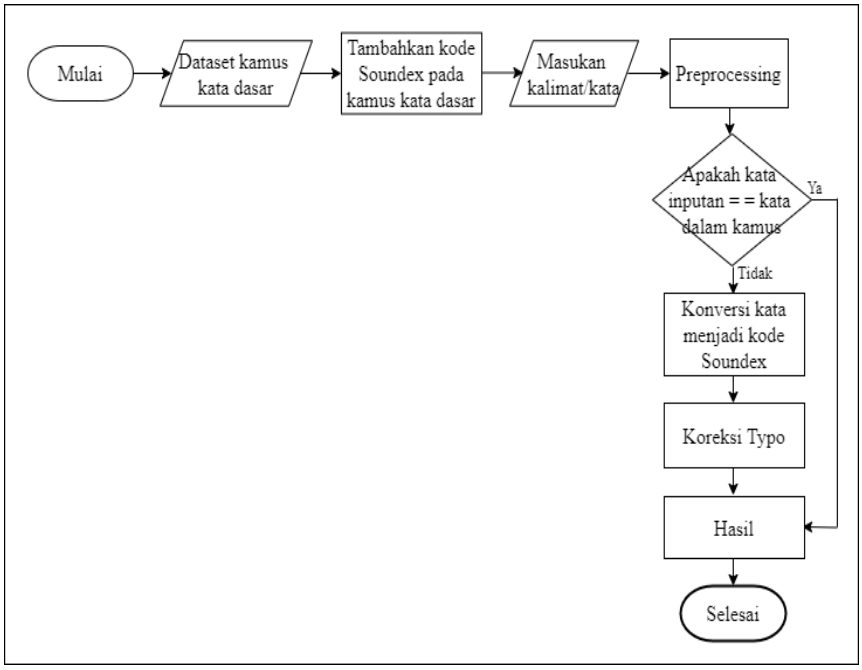
- $lev_{a,b}(4,4) = \begin{cases} \max(i,j) \\ min\begin{cases} lev_{a,b}(3,4)+1 \\ lev_{a,b}(4,3)+1 \\ lev_{a,b}(3,3)+1_{a_i \neq b_j} \end{cases} \end{cases} = min\begin{cases} 1+1=2 \\ 1+1=2 \\ 0+1=1 \end{cases}$

  $lev_{a,b}(4,4) = 1$

Jarak *levenshtein* ditunjukkan pada pojok kanan bawah tabel 3.1 dari tabel tersebut diketahui bahwa jarak antara kata “Nasa” dan “Nasi” adalah 1.

3.3.3.2 Algoritma *Soundex*

Algoritma *Soundex* digunakan untuk mendeteksi apakah ada kesalahan dalam pengetikan atau tidak. Berikut adalah diagram alur untuk mendeteksi adanya kesalahan ketik menggunakan algoritma *Soundex* :



Gambar 3.4 Tahapan perancangan model Soundex

Gambar 3.4 menggambarkan proses yang dilakukan dalam mengoreksi pada kesalahan penulisan kata menggunakan *Soundex*. Dalam mendeteksi kesalahan ketik atau *typo* hal pertama adalah *import* dataset. Lalu tambahkan kode *Soundex* untuk setiap kata pada kamus kata dasar. Setelah itu, *input*kan suatu kalimat, sebelum ke tahap selanjutnya kalimat tersebut harus melalui proses *preprocessing* yaitu *case folding*, *tokenizing*, dan *stemming*. Lalu cek apakah terdapat yang *typo* atau tidak, jika ada maka akan dikoreksi dengan menggunakan aturan *Soundex*, jika

tidak ada kata yang *typo* maka hasil koreksi nya adalah kalimat inputan awal.

Contoh cara kerja algoritma *Soundex* ditunjukkan pada tabel 3.4 :

Tabel 3. 4. Contoh Cara Kerja Algoritma *Soundex*

Kata Asli	**Setelah Pemberian Kode**	**Hasil**
Ambil	A5603	A563
Amsil	A5203	A523

Berikut adalah pembahasan dari tabel diatas :

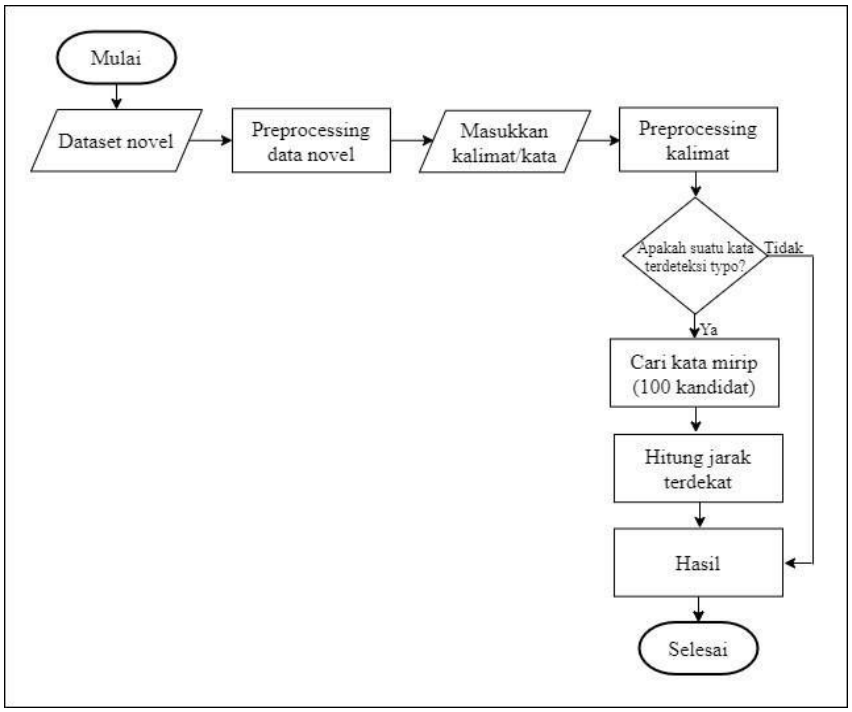
1. Ambil huruf awal pada suatu kata.
   Ambil ⟶ A
   Amdil ⟶ A
2. Rubah huruf-huruf selanjutnya sesuai aturan kode fonetis menjadi kode yang berupa angka 0-6.
   Ambil ⟶ A5603
   Amdil ⟶ A5603
3. Rubah angkat berurut menjadi satu angka.
   Ambil ⟶ A5603 (tidak ada angka berurut)
   Amdil ⟶ A5603 (tidak ada angka berurut)
4. Hapus semua angka 0.
   Ambil ⟶ A563
   Amdil ⟶ A563

Dari pembahasan diatas menyatakan kata ‘Ambil’ dan ‘Amdil’ merupakan kata yang mirip dibuktikan dengan hasil dari kode *Soundex* memiliki nilai yang sama sama.

3.3.3.3 FastText

FastText adalah *library* yang digunakan untuk *word embedding*, yakni teknik untuk mengubah setiap kata menjadi

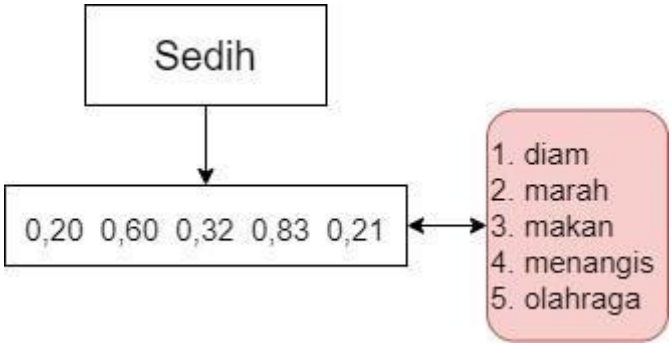
vektor atau *array* yang berupa angka. Berikut adalah diagram alir dari tahapan koreksi kata menggunakan *FastText* :



Gambar 3.5 *Tahapan perancangan model FastText*

Pada gambar 3.5 menunjukan tahapan alur dari perancangan model *FastText*. Dalam melakukan tahapan penelitian dengan metode *FastText*, langkah awal yang harus dilakukan adalah meng *import* dataset (novel). Dataset yang diinputkan harus melalui proses *preprocessing* terlebih dahulu, yaitu : *case folding* dan *tokenizing*. Setelah itu, inputkan suatu kalimat yang akan dikoreksi, kalimat yang diinputkan harus dilakukan *preprocessing* terlebih dahulu untuk mengubah kalimat menjadi huruf kecil dan memecah kalimat menjadi satuan kata. Lalu cek apakah terdapat yang *typo* atau tidak, jika tidak ada kata yang *typo* maka hasil koreksi nya adalah kalimat inputan awal. Namun jika ada kata yang *typo*, maka akan dikoreksi

dengan mencari kata yang mirip sebanyak 100 kandidat. Kandidat-kandidat tersebut akan dihitung jarak terdekatnya dengan kata-kata yang ada di dalam kamus, kata dengan jarak terdekat pada kamus adalah kata hasil koreksi. Berikut adalah proses cara kerja *FastText*:

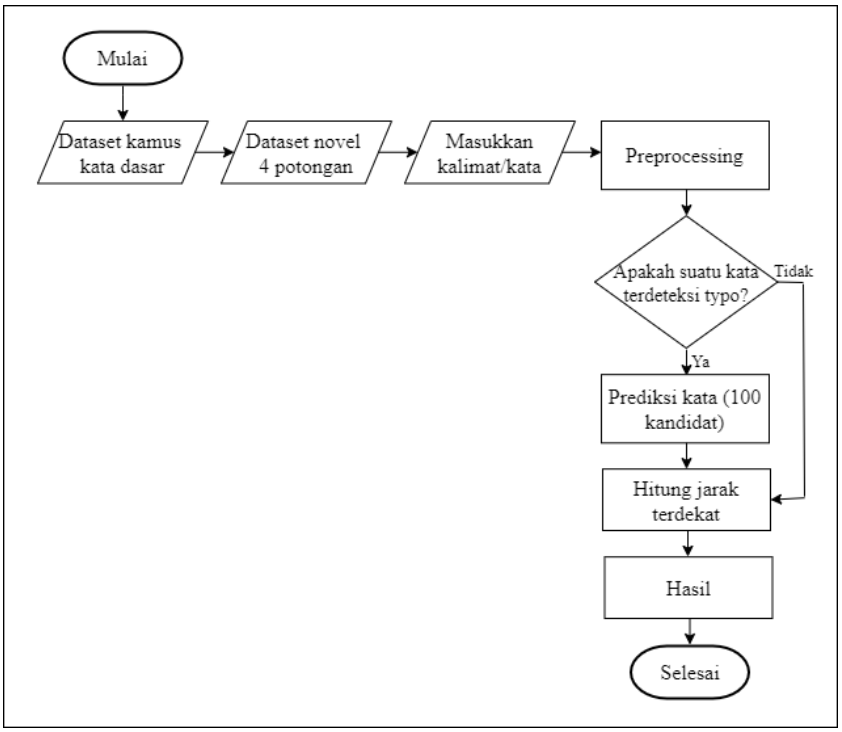


Gambar 3,6 *Cara Kerja* FastText

Gambar 3.6 menunjukkan cara kerja *FastText* dalam mencari kemiripan dua buah kata berdasarkan vektornya. Dijelaskan bahwa di dalam kamus mempunyai ukuran 5 kata, yaitu : diam, marah, makan, menangis, dan olahraga. Jika kita memasukkan sebuah kata yaitu ‘sedih’, maka akan mengeluarkan lima vektor yang besarnya berbedabeda. Dari kelima vektor tersebut, diketahui vektor keempat mempunyai nilai tertinggi dibanding vektor lainnya. Jika dilihat di kamusnya, kata keempat pada kamus adalah kata ‘menangis’. Maka dari itu kata yang memiliki kedekatan nilai tertinggi dengan kata ‘sedih’ adalah kata ‘menangis’.

3.3.3.4 Recurrent Neural Network (RNN)

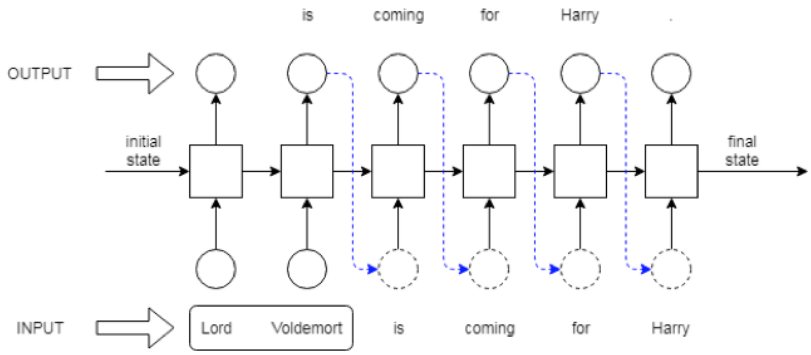
Algotitma *RNN* digunakan untuk memprediksi kata yang sesuai untuk menggantikan kata yang salah. Berikut adalah diagram alir dari algoritma *RNN* :



Gambar 3.7 Tahapan Perancangan Model *Recurrent Neural Network*

Gambar 3.6 merupakan tahapan alur perancangan model dari RNN yang mana dimulai dengan mengimport dataset (kamus dan novel 4 potongan), selanjutnya novel 4 potongan akan di*training*. Lalu masukkan kalimat/kata yang akan dikoreksi, sebelumnya kalimat tersebut harus dilakukan *preprocessing* terlebih dahulu yaitu untuk mengubah kalimat menjadi huruf kecil dan memecah kata menjadi satuan kata. Selanjutnya kalimat akan dicek apakah terdapat yang *typo* atau tidak. Jika tidak ada kata yang *typo* maka hasil koreksi nya adalah kalimat inputan awal. Namun jika ada maka model akan memprediksi 100 kandidat kata yang seharusnya benar, dari kandidat tersebut dihitung jarak terdekatnya dengan kata yang ada di dalam kamus. Kata dengan

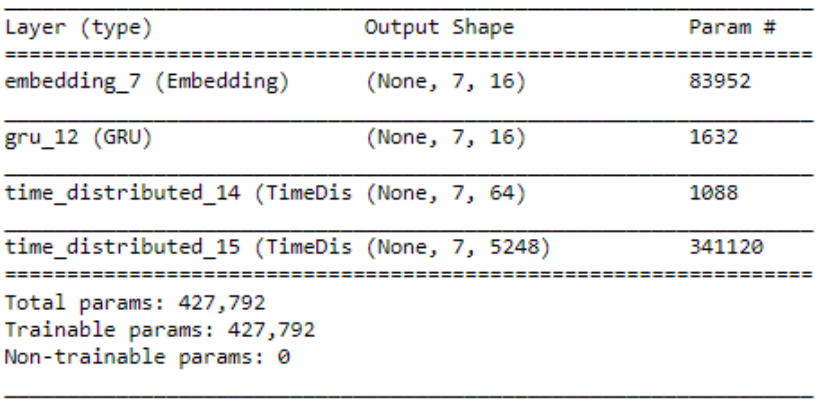
jarak terdekat adalah kata yang dijadikan hasil koreksi. Berikut adalah proses mencari rekomendasi kata untuk kata yang salah :



Gambar 3.8 Proses Mencari Rekomendasi Kata [33]

Pada gambar 3.7 diatas menggambarkan proses inferensi urutan masukan dan kata keluaran. Pada diagram menunjukkan bahwa ada sebanyak enam kata yang diinputkan, enam kata tersebut kami beri label sebagai 1, 2, 3, 4, 5, dan 6. Maka model akan mengeluarkan output berupa enam kata lain yaitu kata dengan label 2, 3, 4, 5, 6, dan kata selanjutnya. Pada kata selanjutnya model akan memilih kata terakhir pada urutan dengan probabilitas tertinggi sebagai prediksi kata selanjutnya.

Arsitektur yang digunakan dalam membangun model RNN adalah arsitektur GRU dimana terdapat satu *layer* dengan beberapa kali percobaan arsitektur. Berikut adalah arsitektur yang digunakan dalam membangun model RNN, dapat dilihat pada gambar 3.8 :



Gambar 3.9 Arsitektur model RNN

Gambar 3.8 adalah arsitektur yang digunakan pada model RNN. Model RNN menggunakan jenis GRU (Gated Recurrent Unit). Arsitektur GRU pemrosesannya lebih cepat dibanding dengan arsitektur LSTM. Arsitektur ini menggunakan 1 *layer* dengan 30 *epoch*, GRU 16 dan *dense* 64.

3.3.4 Pengujian Model

Pengujian model algoritma pada penelitian ini dilakukan secara otomatis. Pengujian model dilakukan sebanyak dua kali pengujian, yaitu pengujian dengan menggunakan dokumen yang berisi 100 kalimat dan dokumen yang berisi 1000 kalimat. Objek yang digunakan sebagai pengujiaan adalah dokumen teks bacaan. Setiap kalimat inputan akan diberikan satu *error* pada salah satu kata dalam kalimat tersebut secara random. Jenis *error* yang digunakan adalah *substitution* (penukaran), dimana suatu huruf akan diganti dengan huruf lainnya agar suatu kata menjadi error (*typo*). Berikut tabel 3.2 adalah contoh dataset yang digunakan untuk pengujian model :

Tabel 3. 2 Dataset Pengujian

Teknik Persiapan Dataset Pengujian	
Lantai rumah itu kotor	Lantai ruma**g** itu
Ibu pergi ke pasar	Ibu pergi ke kan**s**or
…	…
Cuaca hari ini mendung	Cuaca hari ini mendun**f**