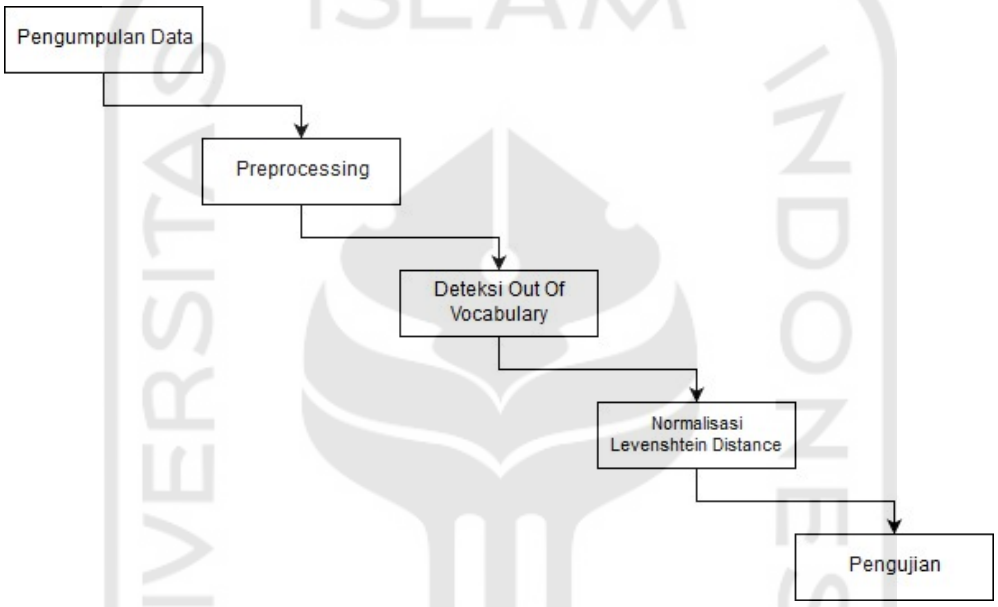
BAB III
METODOLOGI PENELITIAN

3.1 Tahapan Penelitian

Sistem yang akan dibangun dalam penelitian ini memiliki lima tahapan yaitu tahap pengumpulan data, tahap *preprocessing*, tahap deteksi *Out of Vocabulary*, tahap normalisasi, dan pengujian. Tahapan penelitian sistem yang dibuat dapat dilihat pada Gambar 3.1.

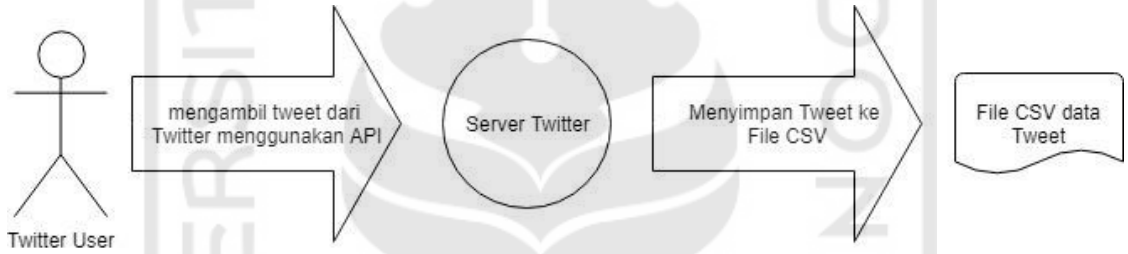


Gambar 3.1 Tahapan penelitian untuk mengembangkan sistem

Gambar tersebut menjelaskan bahwa tahap pertama dimulai dengan pengumpulan data *tweet* dari media sosial Twitter yang digunakan sebagai data. Setelah data *tweet* terkumpul selanjutnya akan melalui tahapan *preprocessing* data, *preprocessing* data yaitu meliputi *case folding*, menghapus URL, menghapus *emoticon*, simbol, dan tanda baca, dan menghapus karakter berulang. Setelah data dibersihkan pada tahap *preprocessing*, maka data akan melalui tahap pendeteksi *Out of Vocabulary* (OOV) untuk dideteksi kata-kata yang tidak standar. Pada pendeteksi *Out of Vocabulary* di penelitian ini, terdiri dari beberapa jenis tahapan yang diambil yaitu meliputi tahap pendeteksi *slangword*, pendeteksi singkatan, dan pendeteksi kata berulang. Tahap selanjutnya adalah tahap normalisasi kata dengan menggunakan algoritma *Levenshtein Distance*. Hasil normalisasi kemudian akan melalui tahap pengujian untuk melihat performa dari sistem yang dibuat.

3.2 Pengumpulan Data

Data yang digunakan dalam uji coba deteksi OOV ini adalah data yang diperoleh dari salah satu media sosial yaitu Twitter, pengumpulan data dengan cara mengambil data *tweet* dari *timeline* pada akun @Lambe_Turah. @Lambe_Turah adalah salah satu akun Twitter yang membahas tentang gosip dan informasi artis dengan gaya bahasa yang bermacam. Twitter dipilih dikarenakan mengandung banyak kalimat tidak beraturan dan mudah didapatkan. Data *tweet* tersebut diperoleh dengan memanfaatkan *Application Programming Interface* (API) yang sudah disediakan oleh Twitter. Dengan memanfaatkan API tersebut maka dibangunlah sebuah sistem *crawling* menggunakan Python. *Tweet* yang akan dilakukan *crawling* adalah *tweet* yang diambil dengan pencarian acak sesuai tanggal 24 April 2019. Gambaran tahap pengambilan data *tweet* dapat dilihat pada Gambar 3.2.



Gambar 3.2 Skema proses *crawling* data *tweet*

Dari gambar dapat dilihat bahwa nantinya peneliti akan membuat sebuah program kecil untuk proses *crawling*. Pada program kecil tersebut maka akan memanfaatkan API dari Twitter. Kemudian, API yang bertugas sebagai penyambung antara Twitter dan program kecil yang peneliti buat tersebut akan melakukan permintaan ke *server*, hasil permintaan tersebut akan menghasilkan *tweet* yang diminta dan kemudian menyimpannya pada *file*.

3.3 *Preprocessing*

Preprocessing pada tahap ini adalah membersihkan dan mempersiapkan teks terlebih dahulu sebelum teks dianalisis oleh komputer. Tahap *preprocessing* dilakukan untuk menghindari data yang kurang sempurna, gangguan data, dan data-data yang tidak konsisten. Adapun tahapan *preprocessing* yang akan dilakukan pada penelitian ini antara lain:

a. Case Folding

Case Folding adalah proses mengubah kata-kata ke dalam bentuk yang sama, misalnya ke huruf kecil atau besar. Dalam tahap ini, semua kata-kata akan diseragamkan menjadi huruf kecil menggunakan metode *python string lower.*

b. Menghapus URL

Situs URL seperti "http://www.situs.com" dan email "namaakun@gmail.com" dihapus pada tahapan ini.

c. Menghapus *Emoticon*, *Symbol* Dan Tanda Baca

Data Twitter memiliki *emoticon*, simbol seperti *hashtag (#), username (@username), retweet* (RT) dan tanda baca yang harus dibuang. Pada tahap ini digunakan *library preprocessing-Twitter* pada Python untuk menghapusnya.

d. Menghapus karakter berulang (*char repetition*)

Data *tweet* yang diperoleh memiliki banyak sekali kata yang memiliki karakter berulang berurutan. Maka dari itu perlu dikurangi karakter pada kata tersebut untuk memudahkan pada tahapan selanjutnya. Maksimal karakter yang dibatasi dalam penelitian ini adalah sebesar 2 huruf.

3.4 Deteksi *Out of Vocabulary* (OOV)

Tahapan deteksi OOV adalah salah satu tahapan yang digunakan untuk menyeleksi setiap kata yang seharusnya dinormalisasi atau diabaikan. Pada deteksi OOV ada beberapa tahapan yang digunakan, yaitu membandingkan kata masukan dengan kamus dasar bahasa Indonesia dan membandingkan kata dengan kamus *slangword* dan singkatan. Kamus dasar bahasa Indonesia berisi daftar kata dasar bahasa Indonesia yang sering digunakan sehari-hari, sedangkan kamus *slangword* dan kamus singkatan berisi kata gaul dan kata yang disingkat pada bahasa Indonesia yang umumnya ditemukan pada penulisan teks media sosial bahasa Indonesia.

a. Mengganti kata gaul (*slangword*)

Bahasa Indonesia pada teks media sosial Twitter banyak mengandung kata yang tidak sesuai kaidah yaitu kata gaul. *Slangword* dalam *tweet* yang terdeteksi pada kamus akan diganti dengan kata yang sesuai kaidah.

b. Mengganti singkatan (*abbreviation*)

Selain kata gaul, penulisan teks pada media sosial juga memungkinkan terjadi kemunculan penyingkatan kata menjadi pendek dari kata yang awalnya memiliki *string*

panjang. Kata kata pada *tweet* yang terdeteksi sesuai kamus singkatan akan diganti dengan kata singkatan yang sebenarnya.

c. Mengganti kata berulang

Dalam data *tweet* yang diperoleh terdapat banyak singkatan yang ditemukan dalam kata yang berulang misalnya kata "sama-sama" menjadi kata "sama2", sehingga tidak memungkinkan untuk memasukan semua ke dalam kamus singkatan dan memisahkan proses perbaikan pada tahap yang berbeda. Tahap ini akan dilakukan pengecekan kata dengan jenis tersebut, kata yang diperiksa adalah kata yang menggandung *string* huruf keseluruhan dan memiliki akhiran karakter "2".

3.5 Perbaikan Kata Dengan *Levenshtein Distance*

Tahap ini adalah tahap untuk menormalisasi kata yang salah eja dengan kata yang standar dengan melibatkan kata dasar bahasa Indonesia dengan algoritma *Levenstein Distance*. Jenis OOV yang akan diperbaiki pada tahapan ini adalah kesalahan ejaan (*miss spelling*). Kata yang terdeteksi sebagai kata yang salah eja akan diperbaiki dengan cara mencocokan kata dengan kamus. Peneliti mengumpulkan kata-kata dari dataset yang diperoleh dari hasil *crawling* melalui Twitter. Setiap kata yang keliru dibandingkan dengan setiap kata yang ada pada kamus kemudian dicari nilai *edit distance* yang terkecil dengan algoritma *Levenshtein Distance*. Langkah langkah perbaikan yang dilakukan *Levenstein Distance* adalah sebagai berikut:

a. Pembentukan tabel dua dimensi untuk kedua *string* yang dibandingkan

Pada tahap ini kata "barng" akan dibandingkan string masukan dengan setiap kata yang ada pada kamus. Contoh kata yang diambil dari kamus untuk dibandingkan yaitu kata "barang". Selanjutnya dibuat tabel dua dimensi seperti Gambar 3.3.

		B	A	R	N	G
	0	1	2	3	4	5
B	1					
A	2					
R	3					
A	4					
N	5					
G	6					

Gambar 3.3 Gambar tabel untuk string yang dibandingkan

b. Membandingkan tabel dua dimensi dengan menghitung *edit distance*

Tahap selanjutnya adalah membandingkan setiap karakter pada tabel yang sudah terbentuk. Karakter pertama pada string pertama dibandingkan dengan karakter pertama pada string kedua. Apabila karakter yang dibandingkan sama, pengisian pada blok [1,1] tinggal menurunkan angka yang ada pada blok [0,0] yaitu angka 0. Jika karakter yang dibandingkan tidak sama, angka yang ada pada blok di kiri, kiri atas, dan atas blok yang akan diisi, ditambah dengan angka 1 (penambahan bersifat sementara tidak permanen sehingga tidak mengubah angka yang di sekitarnya) kemudian dicari jumlah mana yang paling minimum. Jumlah minimum dari penambahan dimasukkan ke blok yang akan diisi. Hasil dari perbandingan karakter dapat dilihat pada Gambar 3.4.

		B	A	R	N	G
	0	1	2	3	4	5
B	1	0	1	2	3	4
A	2	1	0	1	2	3
R	3	2	1	0	1	2
A	4	3	2	1	1	2
N	5	4	3	2	1	2
G	6	5	4	3	2	1

Gambar 3.4 Hasil *edit distance* dari setiap karakter *string*

Pada baris dan kolom terakhir terlihat jumlah *edit distance* dari perbandingan kata “barng” dan “barang” adalah 2. Pada Tabel 3.1 ditunjukkan beberapa kata yang huruf awalan sama dengan kata “barang”.

Tabel 3.1 Saran perbaikan dari perhitungan *edit distance*

No.	**Saran Perbaikan**	***Edit Distance***
1	Bara	2
2	Barang	1
3	Barat	2
4	Beras	2

5	Berang	1
6	Berat	3

Pada Tabel 3.1 dapat dilihat hasil dari algoritma *Levenshtein Distance*, setelah itu akan dilakukan proses penggantian kata yang dideteksi keliru dengan kata yang benar. Kata yang berhasil diprediksi maka akan menampilkan rekomendasi kata dengan mengurutkan nilai *edit distance* dari kecil ke besar. *Pseudocode* dari algoritma yang dibentuk dapat dilihat pada Gambar 3.5. Selain itu, tidak semua yang sudah dicari nilai *edit distance* akan ditampilkan karena semua kata akan memiliki nilai. Sehingga, perlu ditentukan juga nilai ambang batas untuk menyeleksi sebagian kata yang memiliki nilai *edit distance* mendekati dengan kata yang dicari yaitu sebesar 3.

```
m ← length(string1)
m ← length(string1)
if m <> 0 and n <> 0 then
       for i ← 0 to m do
              lev_i0 ← 0
              for j ← 0 to n do
                     lev_0j ← j
                     if I > 0 and j > 0 then
                            if s1_i == s2_j then
                                   lev_ij ← lev_i-1,j-1
                            else
                                   lev_ij ← min(lev_i-1,j, levi,j-1, lev_i-1,j-1)
                            end if
                     end if
              end for
       end for
end if
```

Gambar 3.5 *Pseudocode* algoritma *Levenstein Distance*

3.6 Perancangan Perangkat Lunak

Tahapan ini adalah proses untuk merancang perangkat lunak berdasarkan hasil analisis yang telah dilakukan. Perancangan bertujuan untuk mempermudah untuk membangun perangkat lunak dan membuat perangkat lunak tersusun dengan baik, sehingga dapat lebih efektif dan efisien dalam proses pengembangan perangkat lunak.

3.6.1 Deskripsi Implementasi

Sistem yang dibangun dalam penelitian ini diimplementasikan berdasarkan pada rancangan sistem yang sudah dirancang pada bab sebelumnya. sistem yang dibangun dalam penelitian ini merupakan sistem yang diimplementasikan menggunakan bahasa pemrograman

Python, dan *framework* Django. Adapun perangkat keras dan perangkat lunak pendukung yang digunakan dalam implementasi sistem adalah:

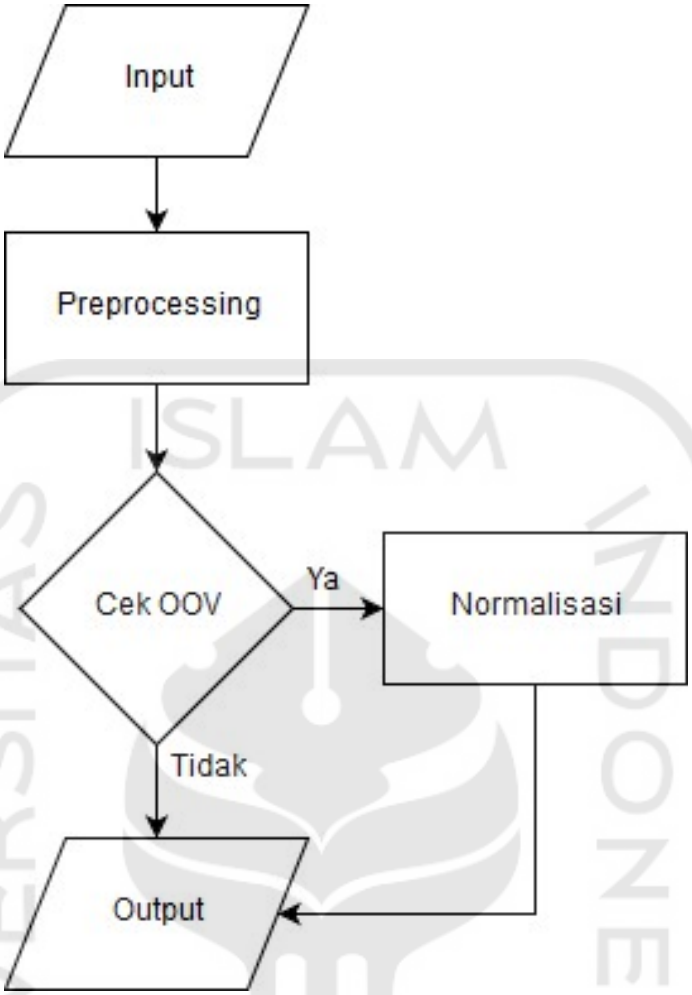
a. Perangkat keras:
    1. Prosesor Intel(R) Core(TM) i5-3337U CPU @ 1.80GHz
    2. Hard Disk Drive 750 GB
    3. Ram 4 GB
b. Perangkat lunak:
    1. Sistem operasi Windows 10 Education 64-bit
    2. Sublime Text 3.1.1
    3. Jupyter 5.7.0
    4. Python 3.6.0
    5. Anaconda
    6. Django 2.1.2

3.6.2 *Flowchart*

Berdasarkan analisis perancangan sistem yang sudah dilakukan, pembuatan *flowchart* sistem yang akan dibangun dalam penelitian terbagi menjadi beberapa bagian yaitu *flowchart* alur normalisasi sistem, *flowchart* deteksi OOV, *flowchart preprocesing*, dan *flowchart* algoritma *Levenshtein Distance*.

a. *Flowchart* alur normalisasi sistem

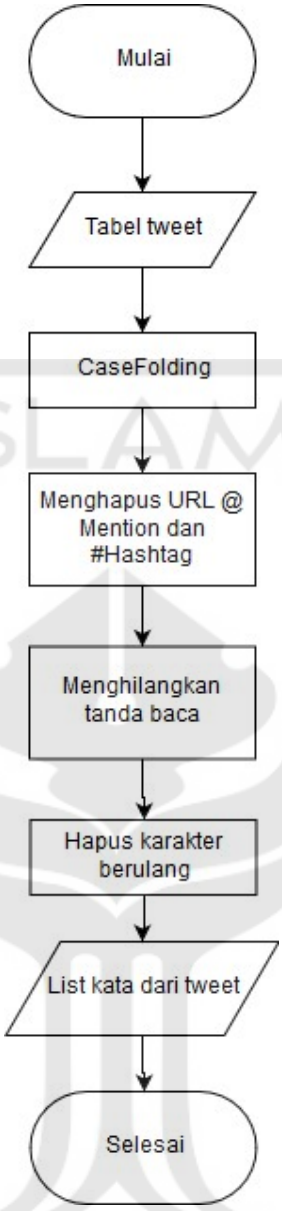
Gambar 3.6 menunjukkan tentang rancangan umum tahapan teks yang dinormalisasi. Pada gambar masukan teks akan melalui tahap *preprocessing*, kemudian melalui tahap deteksi OOV dan apabila memang terdeteksi sebagai OOV, kata akan diperbaiki. Flowchart alur normalisasi sistem menggambarkan seluruh sistem yang akan dibuat.



Gambar 3.6 *Flowchart* alur normalisasi sistem

b. *Flowchart preprocessing*

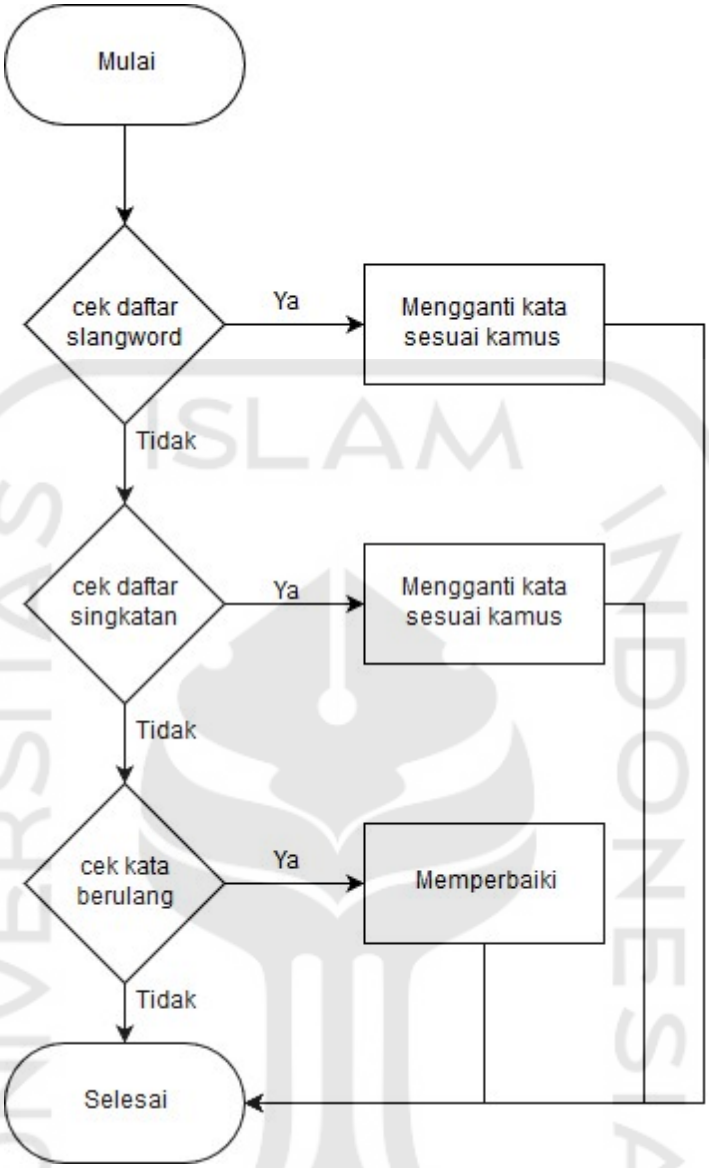
Gambar 3.7 memperlihatkan tentang rancangan tahap *preprocessing* hasil proses *crawling* dari Twitter. Data *tweet* sebelumnya telah disiapkan pada sebuah *file* berformat *txt,* kemudian akan dilakukan proses *clean* dengan beberapa tahapan. Pertama *tweet* akan dilakukan proses penghapusan URL, *mention*, *hashtag*. Selanjutnya *tweet* akan dilakukan proses penyamaan *case* huruf untuk semua *tweet*. Kemudian *tweet* akan dihilangkan beberapa tanda baca yang tidak memungkinkan untuk digunakan. Terakhir adalah adalah menghapus karakter yang berulang apabila ditemukan pada sebuah kata. Hasil dari tahap ini adalah sebuah daftar *list* kata yang siap untuk dideteksi dan dinormalisasi.



Gambar 3.7 *Flowchart preprocesing* pada data *tweet*

c. *Flowchart* deteksi *Out of Vocabulary*

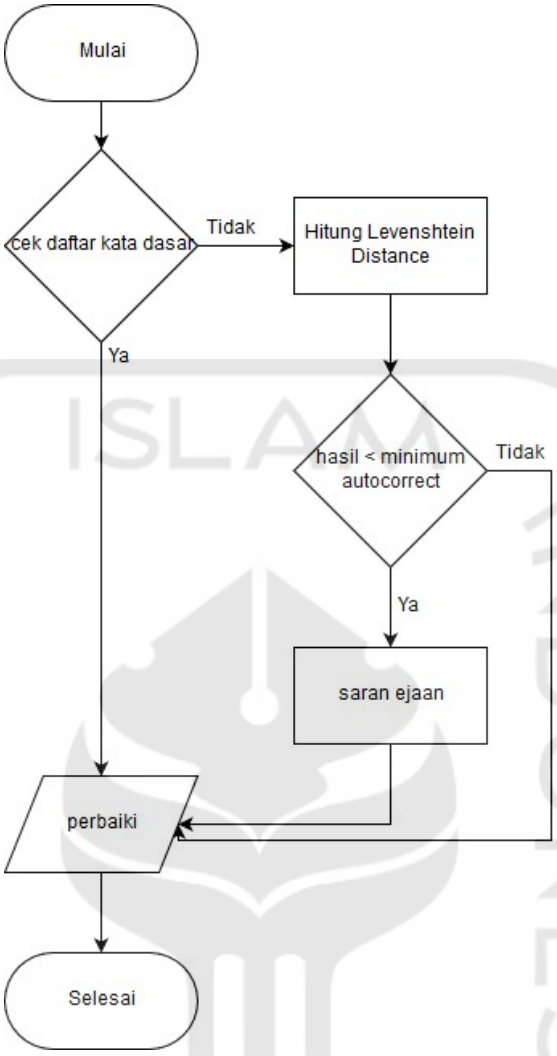
Tahapan mendeteksi OOV digambarkan secara rinci dalam *flowchart* pada Gambar 3.8. Mendeteksi OOV diawali dengan memasukan daftar kalimat *tweet* yang sudah dikumpulkan pada *list,* maka setelah itu akan diperiksa setiap kata dengan kata yang di daftar. Ada 3 tahap pengecekan pada deteksi OOV yaitu deteksi *slangword*, deteksi singkatan, dan deteksi kata berulang. Deteksi *slangword* dan deteksi singkatan menggunakan daftar kamus untuk mendeteksi kata termasuk OOV atau bukan, sedangkan deteksi kata berulang menggunakan algoritma sendiri untuk mengetahui kata tersebut berulang yang disingkat atau tidak.



Gambar 3.8 *Flowchart* deteksi *Out of Vocabulary*

d. *Flowchart* normalisasi *Levenshtein Distance*

Tahap normalisasi *Levenshtein Distance* dijelaskan pada Gambar 3.9. Langkah pertama yang akan dilakukan pada normalisasi *Levenshtein Distance* adalah mengecek *string*/kata yang akan dinormalisasi terdapat di kamus dasar bahasa Indonesia atau tidak. Apabila kata terdapat pada kamus maka kata tidak perlu dinormalisasi, sedangkan apabila kata tidak ditemukan pada kamus, maka kata akan diperbaiki dengan menghitung jarak kata masukan dengan setiap kata di kamus dasar. Kemudian akan diambil kandidat kata yang sudah dihitung dengan algoritma *Levenshtein Distance* dengan kedekatan kata maksimal sebesar tiga.



Gambar 3.9 *Flowchart* algoritma *Levenshtein Distance*

3.6.3 Rancangan *Interface*

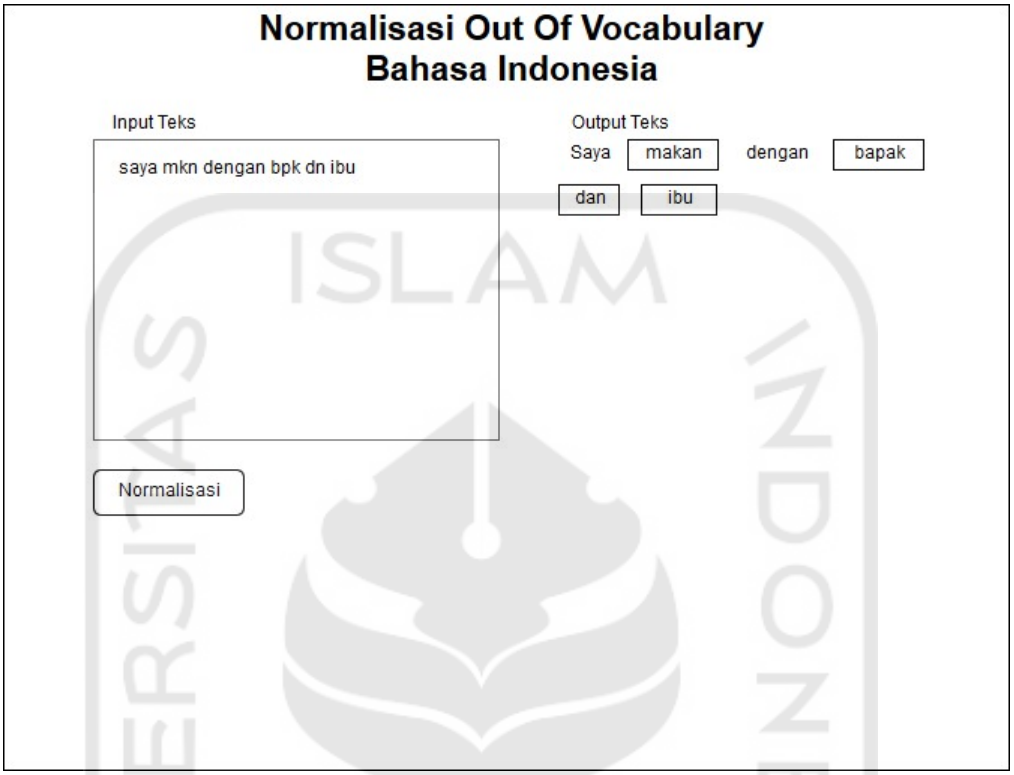
Perancangan *Interface* digunakan untuk menggambarkan tampilan antarmuka yang akan dibuat. Antarmuka sistem merupakan perantara antara sistem dengan pengguna sistem. antarmuka merupakan hal yang penting dikarenakan pengguna dapat berkomunikasi dengan sistem melalui antarmuka.

Pada penelitian ini, rancangan yang dibangun hanya terdapat satu bagian atau satu halaman saja dikarenakan tujuan dari penelitian hanya menampilkan hasil normalisasi.

A. Rancangan *Interface* Normalisasi Teks

Rancangan *interface* normalisasi teks yang akan dibangun diperlihatkan oleh Gambar 3.10 dengan *interface* ini berisi *form input*, *output* dan tombol untuk melakukan normalisasi.

Form input berupa *text field* untuk kalimat yang akan dinormalisasi sedangkan form *output* berupa kalimat hasil prediksi dari normalisasi dari masukan *form input*.



Gambar 3.10 Tampilan antarmuka sistem normalisasi teks

Antarmuka akan menampilkan hasil kata yang terdeteksi OOV dengan kata yang standar sebenarnya, hasil keluaran dapat dilihat di rancangan antarmuka sebelah kanan pada Gambar 3.10. selain itu kata yang sesuai kamus atau terdapat pada kamus standar akan langsung ditampilkan, sedangkan kata yang terdapat pada kamus *slang* dan singkatan akan langsung diperbaiki sesuai kamus perbaikannya. Kata yang terdeteksi bahwa kata tersebut mengandung salah ejaan kata, maka akan menampilkan rekomendasi dari hasil perhitungan *Levenshtein Distance* yang dapat dipilih sesuai kata yang terdekat pada kamus kata standar.

3.7 Pengujian Normalisasi *Levenshtein Distance*

Tahapan pengujian adalah tahapan pengujian fungsi yang telah dibuat dengan algoritma *Levenstein Distance* terhadap keluaran yang dihasilkan. Hal ini bertujuan untuk mengetahui apakah hasil keluaran yang dihasilkan sudah dengan yang diharapkan. Cara melakukan pengujian terhadap keluaran yang dihasilkan adalah dengan membandingkan hasil prediksi normalisasi yang menggunakan algoritma *Levenshtein Distance* dengan hasil teks yang benar

dan dihitung jumlah normalisasi yang benar untuk menghitung akurasi atau persentasenya. Perhitungan persentasenya menggunakan persamaan (3.1).

$$Persentase = \frac{jumlah\ kata\ benar}{jumlah\ kata\ uji} \times 100\% \qquad (3.1)$$

Persamaan tersebut dapat dilihat bahwa jumlah kata benar merupakan banyaknya kata hasil normalisasi yang sesuai dengan yang seharusnya dan jumlah kata uji merupakan banyaknya kata yang diujikan untuk dinormalkan.

Akurasi yang diperoleh dengan menggunakan algoritma *Levenshtein Distance* adalah dengan menguji coba masukan terhadap fungsi yang telah dibangun. Hasil manualnya atau tanpa menggunakan metode diperoleh dengan membuat daftar kata tidak standar dan perbaikan dari kata tidak standar tersebut.