BAB III

METODOLOGI PENELITIAN DAN PERANCANGAN SISTEM

3.1. Metodologi Penelitian

Dalam penelitian implementasi algoritma *Levenshtein Distance* untuk *typo correction* Bahasa Indonesia pada *user feedback* aplikasi ini akan menggunakan metode penelitian yang terdiri dari telaah literatur, pengumpulan data, desain sistem, implementasi, pengujian dan evaluasi, serta penulisan laporan.

3.1.1 Telaah Literatur

Pada tahap telaah literatur dilakukan dengan mempelajari dan membaca penelitian-penelitian sebelumnya dengan sumber dari jurnal ilmiah dan karya tulis ilmiah terkait dengan *typo correction* dan algoritma *Levenshtein Distance.*

3.1.2 Pengumpulan Data

Dalam pengumpulan data akan menggunakan *dataset user feedback* gabungan dari ulasan beberapa aplikasi mobile yang didapatkan dari penelitian Pamungkas (2017) yang berjudul *Word Sense Disambiguation for Lexicon-Based Sentiment Analysis* dan kamus KBBI sebagai *string* yang dibandingkannya didapat dari situs Kateglo. Kateglo merupakan situs yang memberikan informasi tentang Bahasa Indonesia, seperti tesaurus, glorasium, dan sebagainya.

3.1.3 Desain Sistem

Desain sistem sendiri diperlukan perancangan kepada sistem pengoreksian penulisan dengan algoritma *Levenshtein Distance* ini akan digunakan *flowchart* diagram.

3.1.4 Implementasi

Tahap implementasi akan dilakukan menggunakan bahasa pemrograman Python dengan tools Jupyter Notebook dan beberapa *library* untuk menunjang

otomatisasi pembelajaran mesin. Serta pembuatan halaman website untuk kebutuhan demonstrasi pembenaran ejaan.

3.1.5 Pengujian dan Evaluasi

Dalam penelitian ini cara pengujian untuk menghitung peforma yang selanjutnya akan dievaluasi dengan hasil koreksi penulisan dari kata yang dimasukan, apakah ditebak dengan benar dengan kata target yaitu kata di dalam KBBI. Evaluasi dilakukan untuk mengukur seberapa banyak kata yang berhasil ditebak dengan benar dan berapa yang salah. Dan dengan melakukan klasifikasi menggunakan Algoritma Naive Bayes terharap data *user feedback*, dilakukan dua kali dengan perbedaan *dataset* tanpa *Typo Correction* dan dengan *Typo Correction*. Sehingga dapat melihat performa *Typo Correction* dalam memperbaiki *dataset user feedback* dan pengaruhnya terhadap hasil klasifikasi menggunakan Naive Bayes.

3.1.6 Penulisan Laporan

Selama pengerjaan penelitian ini penulisan laporan dilakukan secara paralel sembari melakukan penelitian, sehingga tidak ada proses penelitian yang terlewatkan untuk dituliskan dalam laporan.

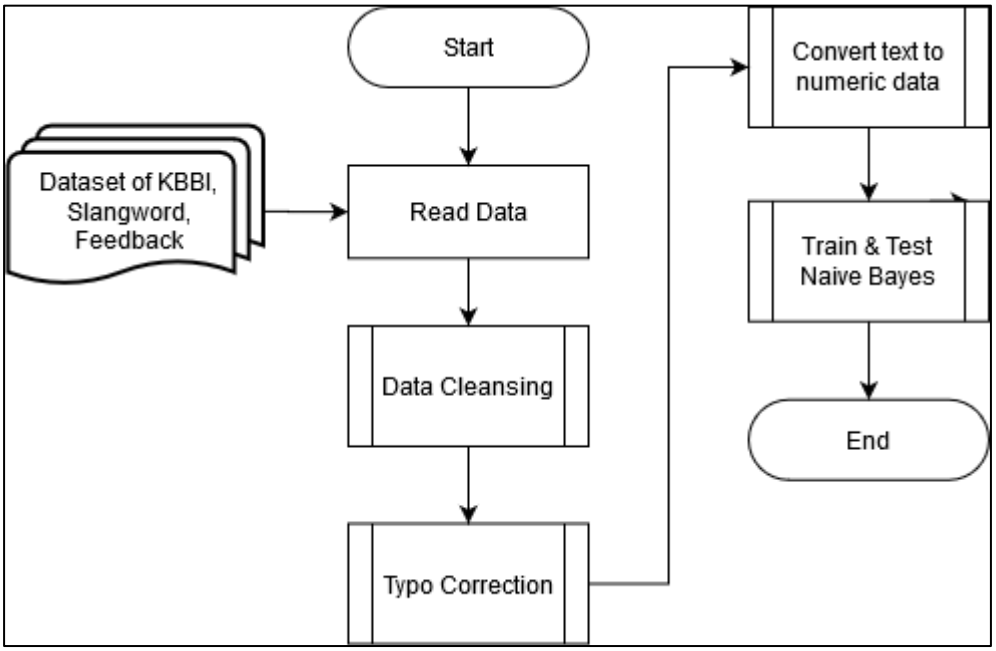
3.2. Perancangan Sistem

Perancangan sistem pengoreksian penulisan dengan algoritma *Levenshtein Distance* ini akan digunakan *flowchart* diagram serta rancangan antarmuka aplikasi berbasis web.

3.2.1. Flowchart

Digunakan diagram *flowchart* untuk memberi gambaran besar secara bertahap sistem implementasi algoritma *levenshtein distance* untuk *typo correction* bahasa Indonesia pada *user feedback* aplikasi. *Flowchart* akan dibagi menjadi tiga, yang pertama adalah *flowchart* proses utama dari read data hingga

pengujiannya berupa klasifikasi. Sisanya adalah *flowchart* sub proses *function typo correction* dan proses klasifikasi.

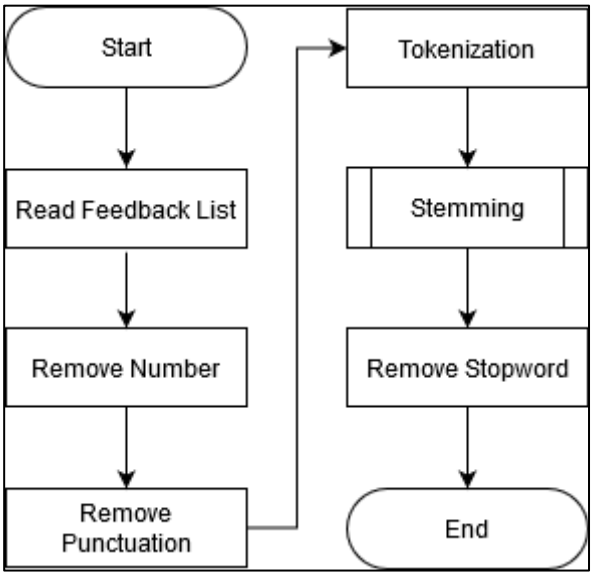


Gambar 3.1. Flowchart Proses Keseluruhan

Dapat dilihat pada Gambar 3.1 adalah gambaran besar dari keseluruhan proses. Dari pertama kali adalah *read data* yang berupa data berformat *CSV* dibuka dan ditampilkan dalam bentuk *DataFrame* dari Pandas *library*. Setelah disajikan dalam bentuk *DataFrame*, kemudian dilakukan *data cleansing* yaitu membersihkan kalimat atau *text*. Manfaat dari membersihkan data pada tahap awal , salah satunya untuk membuat pemrosesan selanjutnya lebih efektif dan dapat menghemat waktu serta tidak menghindari *error* karena terdapat karakter spesial yang dapat berpengaruh ke proses selanjutnya.

Pada tahap pembersihan data ini seperti yang sudah digambarkan dengan *flowchart* pada Gambar 3.2, setelah membaca data lalu pertama kali hilangkan angka karena tidak diperlukan pada proses *text classification* konvensional pada umumnya. Selanjutnya menghilangkan tanda baca, karena hanya dibutuhkan penggalan katanya saja, maka dari itu dihilangkan tanda baca. Untuk proses pemenggalan kata dikenal dengan istilah *Tokenization* yaitu mengubah kalimat menjadi sebuah kata, istilahnya *token.* Kalimat diubah menjadi potongan *token* yang secara teknis berupa *Array of Word* agar mudah diproses dalam perulangan

atau *looping*, karena pada tahap selanjutnya yang disebut *Stemming* akan dilakukan perulangan proses untuk setiap kata.



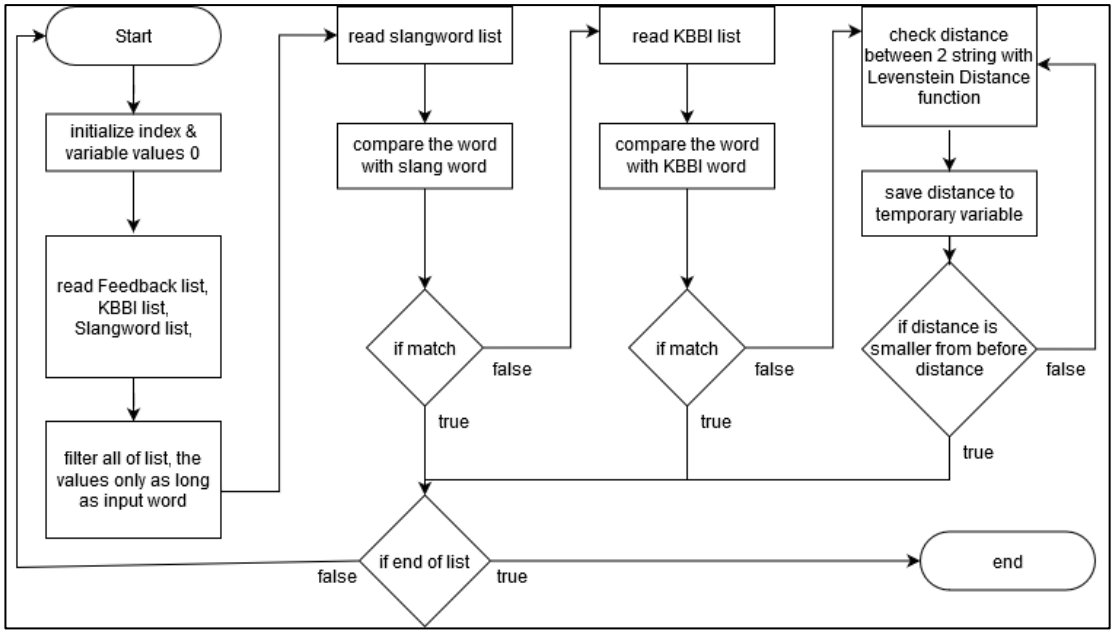
Gambar 3.2. Flowchart proses Data Cleansing

Stemming disini menggunakan *library* Sastrawi, yaitu *library* pengolahan kata untuk bahasa Indonesia. Proses *stemming* menggunakan *library* Sastrawi ini sudah terdapat beberapa pengolahan sendiri didalamnya, sehingga hasil *output* dari fungsi ini sudah lebih mudah untuk di proses selanjutnya. Hasil proses *Stemming* adalah kosa kata baku, karena proses *Stemming* itu adalah menghilangkan imbuhan pada suatu kosa kata.

Lalu dalam kosa kata baku yang sudah dihasilkan, dilakukan penyaringan kembali kata-kata apa saja yang akan berguna untuk proses klasifikasi, dilakukanlah proses *Stopword Remover*. *Stopword* adalah daftar kata yang umum dan banyak digunakan tetapi tidak memilik arti atau makna sebagai kata itu sendiri, tapi dapat memberikan makna jika diberikan ke kosa kata lain, dalam bahasa Indonesia biasa dikenal dengan nama kata bantu.

Kemudian setelah melakukan pembersihan, dataset masuk fase yang paling penting dalam penelitian ini, yaitu Typo Correction. Dapat dilihat pada Gambar 3.3, pertama inisialisasi *variable* indeks dan *variable* yang digunakan menjadi nol, agar menjadi perulangan index disetiap fungsi tersebut dipanggil. Setelahnya yaitu read data yang sudah melalu proses Cleansing dan masuk berupa

satu kosa kata, lalu selanjutnya read data untuk dataset kumpulan kata slang yang berupa DataFrame dan juga dataset kamus besar bahasa Indonesia atau yang disingkat sebagai KBBI. Kata yang masuk kemudian akan dilakukan perulangan untuk membandingkan kata *input* dengan kata pada masing-masing dataset.



Gambar 3.3. Flowchart fungsi Typo Correction

Pertama lakukan pemeriksaan terlebih dahulu dengan daftar kata-kata slang atau bahasa istilah yang tercipta oleh kelompok sosial tertentu pada suatu waktu, dan digunakan dalam situasi informal *dataset slanword* didapatkan dari *website* Github akun milik anggamaulana (2018). Pada saat pemeriksaan jika ditemukan kecocokan, replace dengan kata yang ditemukan didalam *dataset slangword* lalu keluar dari fungsi Typo Correction. Seperti yang dilakukan pada *dataset slangword* , pemeriksaan dilakukan juga kepada dataset KBBI. Setelah melakukan dua kali pemeriksaan di atas jika masih false maka kata input dianggap bukan bagian dari *slangword* dan juga bukan kata yang sesuai dengan kamus besar bahasa Indonesia, lalu disimpulkan bahwa kata tersebut adalah typo dan akan dilakukan pemeriksaan dengan algoritma Levenshtein Distance dan dicari *distance* yang terkecil, setelah ditemukan yang terkecil lalu keluar dari fungsi Typo Correction.

Ketika dataset sudah melalu tahap pembersihan dan fokus pada penelitian ini yaitu adalah Typo Correction, selanjutnya akan dilakukan pengujian dengan cara membandingkan dataset yang tidak dilakukan proses Typo Correction dengan yang melalui proses Typo Correction dengan tujuan untuk mengetahui apakah akan menghasilkan perbedaan yang signifikan, dan dataset manakah yang akan mendapatkan score lebih baik setelah dilakukan klasifikasi. Klasifikasi yang digunakan pada pengujian ini adalah Naive Bayes, karena merupakan metode klasifikasi yang bagus untuk *text processing* dan cukup konvensional tetapi hasil akurasinya cukup tinggi.

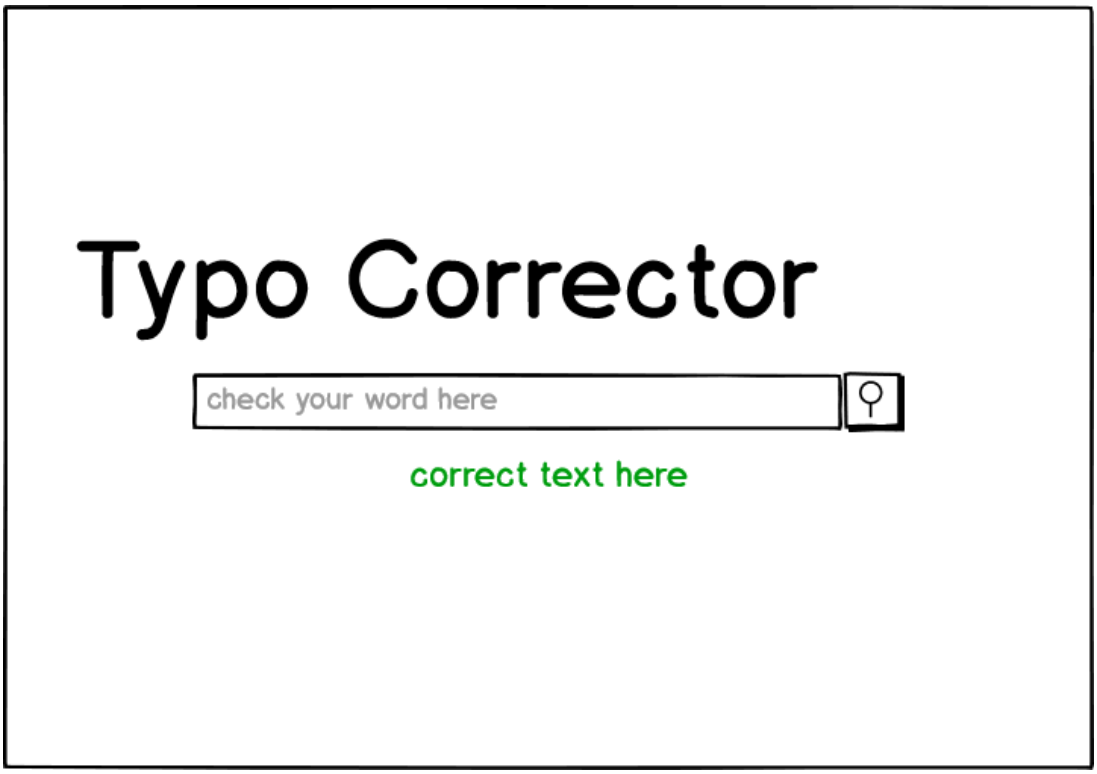
Naive Bayes yang akan digunakan didapatkan dari library Scikit Learn, sehingga hanya akan dijelaskan sedikit tentang cara kerja Naive Bayes. Tapi sebelum mulai proses klasifikasi, dataset terlebih dahulu dilakukan *feature extraction* menggunakan CountVectorizer, yang didapatkan juga dari library Scikit Learn. Maksudnya yaitu mengubah data yang tadinya teks menjadi data numerik agar dapat diproses oleh algoritma Naive Bayes.

Pada dasarnya konsep Naive Bayes adalah menghitung probabilitas suatu kata pada label atau class tertentu, sehingga dari sekian banyak dataset misalkan kata “sehat” lebih banyak muncul, misalkan tujuh kali muncul dengan label “positif”, lalu ada tiga kata “sehat” ditemukan pada data yang berlabel “negatif” maka saat dilakukan testing, kata “sehat” akan ditebak atau dilabelkan “positif”. Probabilitas kata “sehat” pada pada label “positif” adalah 0,7 , sedangkan pada label “negatif” adalah 0,2 sehingga kata “sehat” dianggap positif.

3.2.2. Rancangan Antar Muka

Antar muka dibuat berbasis web yang hanya berfungsi sebagai antarmuka dari fungsi Typo Correction yang dibuat menggunakan bahasa pemrograman Python, yang akan dihubungkan ke antarmuka web melalui API. Framework Flask digunakan sebagai pembuat API, karena berbasis bahasa pemrograman Python sehingga mudah terhubung dengan fungsi Typo Correction. Rancangan antarmuka kurang lebih dapat dilihat seperti pada Gambar 3.4. Antar muka ini

dibuat sebagai alat demonstrasi atau visualisasi dari hasil *Typo Correction* pada *input* yang baru atau mau dilakukan percobaan serta sebagai alat pengujian apakah ada kesalahan atau dikenal dengan istilah *debugging*.



Gambar 3.4. Kerangka tampilan web