BAB I
PENDAHULUAN

1.1. Latar Belakang Masalah

Dalam pembuatan suatu program piranti lunak, dibutuhkan proses *requirementss engineering*, yaitu proses mengumpulkan kebutuhan, menganalisa, mencocokan dengan kondisi dan kebutuhan *user* atau *client*. *Requirements* pada sebuah sistem tidak timbul secara natural, tetapi dikembangkan dan terus di *review* dan dilakukan revisi (A Van Lamsweerde, 2009). Selanjutnya adalah tahap *deployment* yaitu membuat piranti lunak tersebut dapat diakses dan digunakan. Setelah piranti lunak dapat digunakan, *user* atau pengguna dapat memberikan tanggapan berupa meminta tambahan fitur atau melaporkan *bug* yang dialami dan ditemukan kepada *developer*. Seiring dengan perkembangan jaman, dengan difasilitasi komunitas di media sosial, sehingga membantu menghapuskan jarak antara pengguna dan para *developer* (Pagano & Bruegge, 2013).

Tanggapan dari pengguna atau biasa dikenal dengan istilah *user feedback* semakin dibutuhkan karena piranti lunak saat ini lebih berorientasi pada pengguna. *User feedback* kepada piranti lunak yang telah dibuat untuk mengetahui *user experience* yang dapat bermanfaat sebagai masukan ke *developer* untuk pengembangan *feature* yang belum ada atau perbaikan selanjutnya (Pagano & Maalej, 2013).

User feedback dapat diolah dengan maksud mengamati apa saja yang sering dikritik oleh *user* tentang aplikasi tersebut, dan pesan apa yang ingin disampaikan oleh *user* atau fitur apa yang diinginkan untuk ditambahkan selanjutnya. Lingkup yang besar menjadi sebuah tantangan bagi pengolahan

manual karena akan memakan waktu yang lebih banyak, sehingga diharapkan adanya sebuah proses atau alat yang dapat membuat pengolahan menjadi lebih cepat.

Dari masalah di atas, komputer dapat digunakan sebagai alat untuk mengolah data *user feedback* dengan cara mengotomatisasi pemeriksaan dan analisa untuk masing-masing data yang ada. Manfaat menganalisa data dengan otomatisasi selain lebih cepat, juga lebih banyak *parameter* atau faktor - faktor yang dapat dianalisa untuk menghasilkan informasi yang berguna. Informasi yang dihasilkan dari analisa untuk pengembangan selanjutnya biasanya disebut juga dengan istilah *insight.* Contohnya dapat membuat tim *developer* lebih efektif dalam menentukan fitur apa yang akan dikembangkan atau diperbaiki karena mendapatkan hasil analisa dari *user feedback.* Dengan menggunakan komputer dapat mempermudah saat proses verifikasi dan validasi *feedback* yang didapat (Wieringa & Persson, 2010).

Sebelum masuk diolah sebaiknya *dataset* tanggapan pengguna melalui fase *preprocessing* terlebih dahulu. Fase *preprocessing* perlu dilakukan untuk mengurangi ukuran *text input* dengan signifikan (Dalal dan Zaveri, 2011). *Preprocessing* yang dimaksud yaitu memanipulasi ataupun menyeleksi kalimat dalam *dataset* seperti, *stemming* untuk menghapus kata imbuhan, *synonym extraction* untuk mengelompokan kata yang bermakna sama, *punctuation* untuk menghapus tanda baca, dan salah satunya juga *typo correction* untuk mengoreksi penulisan yang salah dan diperbaiki menjadi kata yang dimaksud oleh pengguna.

Dalam *typo correction* terdapat banyak algoritma yang dapat digunakan seperti *N-Gram, Damerau Levenshtein Distance, Levenshtein Distance*. Dalam

penelitian ini akan menggunakan algoritma *Levenshtein Distance*, karena pada penelitian sebelumnya yang dilakukan oleh Andre Fernando pada 2016 lalu menyatakan bahwa algoritma *Levenshtein Distance* lebih cepat daripada algoritma *Damerau Levenshtein Distance* dengan perbedaan nilai *similarity* atau kemiripan yang dihasilkan relatif sama dengan rata-rata sebesar 81%. Sebab itu *Levenshtein Distance* akan digunakan untuk mencari nilai *similarity* terbaik diantara dua *string* atau kosakata, setelah didapatkan nilai *similarity* atau kemiripan yang terbaik maka dapat disimpulkan sebagai kemungkinan kata yang dimaksud.

Sudah pernah dilakukan sebuah penelitian menggunakan algoritma *Levenshtein Distance* untuk pencarian kata isu di kota Bandung melalui *platform Twitter* oleh Dewi Rosmala dan Zulfikar Muhammad Risyad (2018). Hasil pengujian dalam penelitian membuktikan algoritma *Levenshtein Distance* dapat 100% mengubah kata penulisan yang salah pada *Twitter* yang menjadi *input*, menjadi kata kunci pada kategori atau klasifikasi isu yang sudah ditentukan.

Perbedaan penulisan pengetikan setiap orang dalam *user feedback* menjadi masalah yang akan menghambat proses klasifikasi karena komputer tidak akan mengenali kata yang penulisannya salah sebagai kata yang seharusnya dimaksud pengguna. Fase *preprocessing* secara otomatis juga penting dalam proses otomatisasi pengolahan *user feedback* (Wieringa & Persson, 2010). Dalam kasus ini jika ada beberapa kata yang tidak berhasil dikenali dengan efektif atau dengan kata yang seharusnya, bisa jadi terjadi salah penilaian terhadap kata tersebut pada saat proses *learning* atau pembelajaran. Oleh karena itu berdasarkan dari latar belakang yang sudah dijelaskan sebelumnya, penelitian ini berfokus untuk

mengimplementasikan algoritma *Levenshtein Distance* untuk *typo correction* atau koreksi kesalahan penulisan serta mengukur performanya.

1.2. Rumusan Masalah

Berdasarkan latar belakang penelitian ini, rumusan masalah dalam penelitian ini dapat rumuskan seperti berikut.

a. Bagaimana mengimplementasikan algoritma *Levenshtein Distance* untuk *typo correction* Bahasa Indonesia pada *user feedback* aplikasi?
b. Mengukur bagaimana performa algoritma *Levenshtein Distance* untuk *typo correction* Bahasa Indonesia?

1.3. Batasan Masalah

Batasan masalah yang diperhatikan dalam penelitian ini diantaranya adalah:

a. Algoritma Lenvenshtein Distance dibantu dengan metode Normalized Similarity dan setelah melakukan eksperimen atau uji coba seberapa banyak karakter yang dapat ditoleransi untuk menemukan kata dengan makna yang sama ternyata sekitar tiga sampai empat karakter untuk mendapatkan hasil yang baik. Sehingga diputuskan untuk hanya memiliki maksimal tiga perbedaan karakter dari kata yang sebenarnya.
b. *Dataset* yang digunakan adalah *review* dari *platform* penjualan aplikasi berbasis *mobile* yang pada umumnya memiliki tiga kolom yaitu *review, sentiment, rating*. Hanya akan digunakan kolom *review* atau ulasan pengguna tersebut dan *sentiment* yang berupa label apakah *reviewnya* negatif atau positif.

1.4. Tujuan Penelitian

Berdasarkan latar belakang yang sudah dijabarkan, penelitian ini memiliki beberapa tujuan sebagai berikut.

a. Mengimplementasikan algoritma *Levenshtein Distance* untuk *typo correction* Bahasa Indonesia pada *user feedback* aplikasi.

b. Mengukur performa algoritma *Levenshtein Distance* untuk *typo correction* Bahasa Indonesia.

1.5. Manfaat Penelitian

Typo correction bertujuan untuk memperbaiki kesalahan penulisan yang tidak dapat dikenali oleh komputer sesuai kamus kosa-kata atau kata yang seharusnya, agar dapat memperbaiki *data user feedback* aplikasi menjadi penulisan yang benar. Dengan hasil penelitian ini diharapkan dapat membantu para *developer* atau orang – orang yang mengembangkan *Requirements Engineering* kedepannya dapat membersihkan *dataset* yang berbahasa Indonesia dari kesalahan pengetikan atau penulisan.

1.6. Sistematika Penulisan

Sistematika penulisan laporan skripsi ini dapat dijabarkan dalam detail sebagai berikut:

BAB I LATAR BELAKANG

Bab I Pendahuluan berisi latar belakang, rumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian, dan sistematika penulisan.

BAB II LANDASAN TEORI

Bab II Landasan Teori berisi penjelasan atau landasan teori mengenai *Requirements Engineering, User Feedback, Typo Correction,* dan *Levenshtein Distance* sebagai algoritma yang digunakan untuk penelitian ini.

BAB III METODOLOGI PENELITIAN DAN PERANCANGAN SISTEM

Bab III Metodologi Penelitian dan Perancangan Sistem berisi penjelasan mengenai metodologi penelitian yang digunakan. Bab ini juga berisi perancangan *flowchart* dan perancangan antar muka website yang akan digunakan sebagai interface.

BAB IV IMPLEMENTASI DAN ANALISIS

Bab IV Implementasi dan Analisis berisi penjelasan mengenai implementasi yang sudah dibuat, dan analisis dari aplikasi yang sudah dibangun.

BAB V KESIMPULAN DAN SARAN

Bab V Kesimpulan dan Saran berisi kesimpulan dan saran yang dapat digunakan untuk mengembangkan aplikasi di kemudian hari.