BAB I

PENDAHULUAN

1.1 Latar Belakang

Menulis merupakan salah satu keterampilan dan kebiasaan manusia dalam kehidupan sehari-hari bahkan dalam dunia pendidikan. Menulis juga salah satu media komunikasi yang di sampaikan secara tidak langsung, baik itu dalam bentuk berita, jurnal, laporan dan tulisan lainnya. Setiap kata dalam satu kalimat memiliki makna yang penting dan saling berkaitan dengan kalimat lainnya. Tetapi dalam pengetikan ada yang sesuai dengan EYD (Ejaan Yang Dibenarkan) ada juga yang tidak sesuai EYD, dalam bahasa lain bisa disebut bahasa yang baku dan tidak baku. Aktifitas menulis merupakan salah satu manisfestasi kemampuan dan keterampilan berbahasa paling akhir yang dikuasai pengguna bahasa setelah mendengarkan, membaca dan berbicara [1].

Teknologi informasi pada zaman sekarang sangat berkembang pesat salah satu yang sangat berperan pada masa ini adalah komputer, dapat digunakan sebagai media belajar, pekerjaan yang membutuhkan fasilitas komputer, didalamnya memiliki beberapa fitur yang dapat kita manfaatkan sesuai dengan kubutuhan, salah satunya fitur untuk menulis. Baik itu menulis cerita pendek, novel, tugas kuliah, pekerjaan, dan aktifitas lainnya yang berhubugan dengan menulis. Tetapi dalam kasus ini, yang akan lebih di bahas adalah penulisan Skripsi.

Kesalahan pengetikan yang terjadi dapat menyebabkan kata yang baku berubah menjadi tidak baku yang tidak sesuai dengan kalimat yang sebenarnya.

Adapun beberapa faktor yang dapat mempengaruhi terjadinya kesalahan pengetikan, diantaranya salah menekan tombol pada *keyboard*, jarak antara tombol-tombol yang terlalu berdekatan. Contoh kasusnya, misal yang seharusnya seseorang menulis kata "dengan" tetapi orang tersebut menulis "dengn". Proses pengecekan kesalahan pengetikan dengan cara manual akan menghabiskan banyak waktu dan membutuhkan suatu sumber pasti sebagai acuan bahwa kata tersebut memang salah dalam proses penulisannya. *Efisiensi* waktu yang dibutuhkan jika dilakukan dengan manual tentunya tidak akan optimal dan cukup membosankan sehingga kemungkinan adanya *human error* dapat mengakibatkan proses pengecekan kata menjadi tidak optimal [2].

Untuk mengurangi kesalahan penulisan pada editor teks *Miceosoft word* terutama pada kata bahasa Indonesia diperlukan sebuah sistem agar dapat membantu pekerjaan seorang penulis untuk memperbaiki skripsi atau naskah lainnya karena kesalahan pengetikan tersebut. Dalam pengecekan kesalahan pengetikan tersebut, apabila ditemukan kata yang salah maka harus dilakukan pencarian kemungkinan kata yang sesuai. Pada proses pencaraian kemungkinan kata tersebut, diperlukan suatu pendekatan pencarian *string* khusus yaitu dengan pendekatan perkiraan.

Approximate string matching dapat digunakan untuk pencarian *string* berdasarkan *string* yang sama dan *string* yang memiliki kemiripan penulisan dengan *string* yang terdapat pada kamus. Metode ini dapat digunakan untuk pencarian kata tidak baku karena dapat mengidentifikasi *string* yang sama dan yang memiliki kemiripan penulisan. Pada *Approximate String Matching* terdapat tiga macam operasi yang digunakan untuk mentransformasikan suatu *string* menjadi

string lain. Operasi tersebut antara lain yaitu operasi penghapusan, penyisispan, dan penggantian. Operasi-operasi ini digunakan untuk menghitung jumlah perbedaan yang diperlukan untuk mempertimbangkan kecocokan suatu *string* dengan *string* sumber. Jumlah perbedaan tersebut diperoleh dari penjumlahan semua pengubahan yang terjadi dari masing-masing operasi. Pengguanaan perbedaan tersebut diaplikasikan dalam berbagai macam algoritma salah satunya algoritma *Levenshtein distance*[3].

Levenshtein Distance sendiri dikembangkan oleh *Vladimir Levenshtein* pada tahun 1965. Levenshtein Distance adalah salah satu dari beberapa metode untuk menghitung jarak perbedaan antar kata. *Levenshtein Distan*ce merupakan metode dalam menghitung nilai yang didapat dari hasil operasi modifikasi satu kata dengan kata yang lain dengan bantuan matrix. Cara yang digunakan adalah dengan melihat satu persatu karakter dengan karakter lainya, apakah untuk menutupi perbedaan tersebut perlu adanya penambahan huruf, penghapusan huruf, atau penyisipan huruf.

Dengan menggunakan fungsi matrix(m,n) dimana M mewakili kata yang dibandingkan, sedangkan N sebagai pembanding yang masing masing mewakili setiap huruf sehingga dapat lebih mudah melihat operasi apa yang perlu dilakukan untuk kata tersebut. Nilai yang akan didapat adalah seberapa banyak langkah yang diselesaikan untuk mendapatkan kemiripan kata (Rani & Shanthi, 2015). Total angka untuk setiap operasinya mengacu kepada distance, dimana semakin kecil distancesemakin besar kemungkinan kata sesuai dengan target kata [4].

Berdasarkan latar belakang tersebut, permasalahan yang muncul adalah bagaimana caranya agar kita dapat meminimalisir kesalahan pengetikan, dan

bagaimana koreksi kesalahan tersebut dapat otomatis di koreksi oleh sistem sehingga dapat menghasilkan karya ilmiah yang sesuai dengan kaidah bahasa Indonesia yang baik dan benar. Maka dari itu di dapatkan sebuah aplikasi dengan judul "**Perbandingan Akurasi Algoritma *Levenshtein Distance* Dengan Algoritma *Cosine Similarity* Untuk Koreksi Penulisan Dalam Kalimat".**

1.2 Rumusan Masalah

Berdasarkan latar belakang, terdapat beberapa masalah yang dapat di identifakasi diantaranya :

1. Bagaimana implementasi *text mining* untuk koreksi kesalahan pengetikan pada kalimat menggunakan algoritma *Cosine Similarity* ?
2. Bagaimana implementasi *text mining* untuk koreksi kesalahan pengetikan pada kalimat menggunakan algoritma *Dameru Levenshtein Distance* ?
3. Bagaimana perbandingan kinerja pendeteksi kesalahan penulisan dengan menggunakan algoritma *Cosine Similarity* dengan *Levenshtein distance* ?

1.3 Tujuan Penelitian

1. Implementasi *text mining* untuk mendeteksi kesalahan penulisan dengan menggunakan sistem dan algoritma *Cosine Similarity?*
2. Implementasi *text mining* untuk mendeteksi kesalahan penulisan dengan menggunakan sitem dan algoritma *Levenshtein distance ?*
3. Mengetahui perbandingan kinerja algoritma *Cosine Similarity* dengan *Levenshtein distance* untuk memperbaiki kesalahan pengetikan setelah dilakukannya pengecekan identifikasi menggunakan system

1.4 Manfaat Penelitian

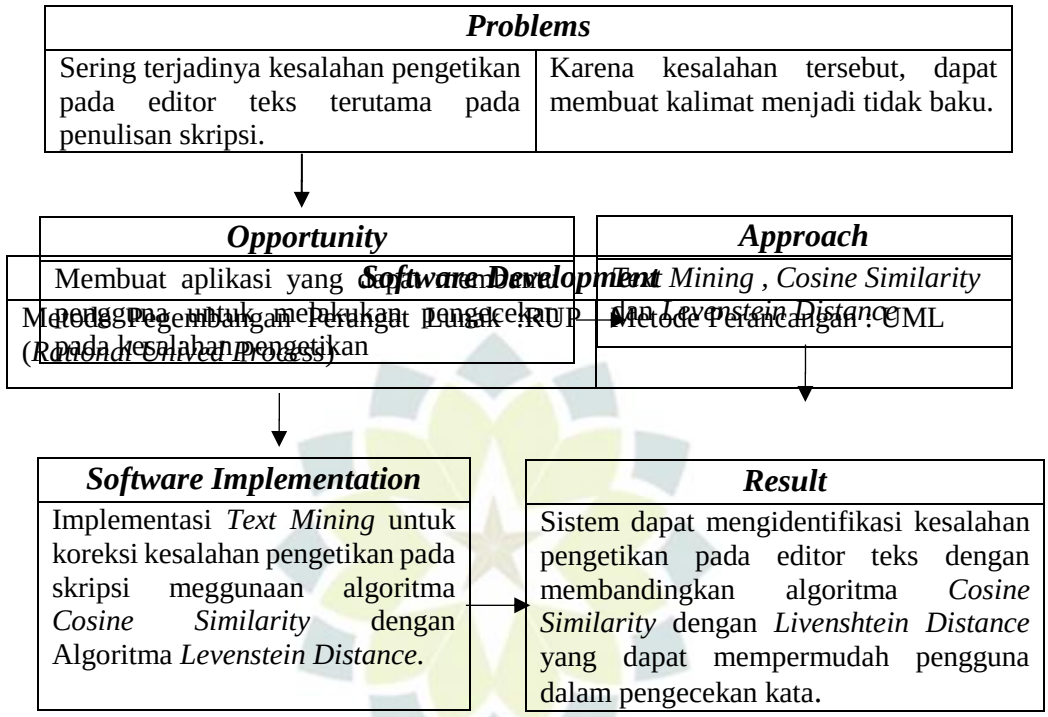
Dibangunnya sistem koreksi kesalahan penulisan kalimat menggunakan algoritma *Cosine Similatity* dengan *Levenshtein distance*, diharapakna dapat membantu semua pengguna yang memiliki kesulitan untuk memperbaiki kesalahan pengetikan / penulisan, supaya tidak terlalu banyak revisi karena sering terjadinya kesalahan dalam penulisan (*typo)* laporan tersebut. Selain iu, dapat mengefesiensikan waktu dalam koreksi.

1.4 Batasan Masalah

1. Pengidentifikasi kesalahan pengetikan teks hanya untuk Bahasa Indonesia
2. Proses identifikasi berupa kesalahan pengetikan, kalimat tidak baku yang akan diberikan tanda merah pada kesalahan teks tersebut.
3. Pengecekan kesalahan pengetikan dengan objek kalimat pada dokumen punilasan
4. *File* yang diidentifikasi berupa dokumen teks dengan cara meng*import* teks kedalam *text editor*
5. Sistem ini akan menggunakan 2 menu, yaitu menu dengan menggunakan algoritma *Levenshtein distance* dan algoritma *Cosine Similarity*
6. Ukuran maksimal dokumen yang akan diinputkan 1 MB.
7. Target pada sitetm berupa kata-kata bahasa Indonesia yang meliputi kata dasar, kata imbuhan me, ber, ter dan me-kan.
8. Target pada sistem menggunakan 30218 kata Bahasa Indonesia tidak termasuk nama Negara, nama hewan, nama buah, nama manusia dan singkatan.
9. Sistem yang akan dibangun berbasis *web* menggunakan PHP.

1.5 Kerangka Pemikiran

Adapun kerangka pemikiran yang digambarkan dapat di lihat pada **Gambar 1.1 :**



Gambar 1.1 **Kerangka Pemikiran**

1.6 Metodologi Penelitian

1.4.1 Metode Pengumpulan Data

Metode yang digunakan dalam pengumpulan data pada penelitian ini terdiri dari 2 tahapan, yaitu:

1. Observasi

   Teknik pengumpulan data dengan mengadakan penelitian dan peninjauan langsung terhdap objek penelitian.

2. Studi Literatur

   Pengumpulan data dengan cara mengumpulkan literatur, jurnal dan bacaan-bacaan yang terkait dengan judul proposal.

Pengumpulan data dengan mengumpulkan literatur, jurnal.

1.4.2 Metode Pengembangan Perangkat Lunak

RUP (*Rational Unived Process*) adalah pendekatan pengembangan perangkat lunak yang dilakukan berulang-ulang (*iterativ*), fokus pada arsitektur (*architecture-centric*), lebih diarahkan berdasarkan penggunaan kasus (*use case driven*). RUP merupakan proses rekayasa perangkat lunak dengan pendefinisian yang baik (*well defined*) dan penstrukturan yang baik (*well structured*). RUP menyediakan pendefinisian struktur yang baik untuk alur hidupp proyek perangkat lunak. RUP adalah sebuah produk proses peranfkat lunak yang dikembangkan oleh *Rational Software* yang diakuisisi oleh IBM di lunan februari 2003.

Gambar 1.2 Proses RUP (*Rational Unived Process*)

RUP memiliki empat buah tahap atau fase yang dapat dilakukan pula secara iternatif. Berikut ini penjelasan untuk setiap fase pada RUP :

1. *Inception* (permulaan)

   Tahapan ini leih kepada memodelkan proses bisnis yang dibutuhkan (*business modeling*) dan endefinisikan kebutuhan akan sistem yang akan

dibuat (*requirements*). Hasil yang diharapkan dari tahap ini adalah memenuhi *Lifecycle Objective Milestone* (batas/tonggak objektif dari siklus) dengan kriteria berikut :

a. Umpan balik dari pendefinisian ruang lingkup, perkiraan biaya dan perkiraan jadwal.
b. Kebutuhan dimengerti dengan pasti (dapat dibuktikan) dan sejalan dengan kasus primer yang dibutuhkan
c. Kredibilitas dari perikiraan biaya, perkiraan jadwal, penentuan skala prioritas, resiko, dan proses pengembangan.
d. Membangun garis dasar dengan membandingkan perencanaan actual dengan perencanaan yang direncanakan.

2. *Elaboration* (perluasan/perencanaan)

   Tahap ini lebih difokuskan pada perencanaan arsitektur sistem. Tahap ini juga dapat mendeteksi apakah arsitektur sistem yang diinginkan dapat dibuat atau tiak. Mendeteksi resiko yang mungkin terjadi dari arsitektur yang dibuat. Tahap ini lebih pada analis dan desain sistem serta implementasi sistem yang fokus pada purwarupa sistem (*prototype*).

3. *Construction* (kontruksi)

   Tahap ini fokus kepada pengembangan komponen dan fitur-fitur sistem. Tahap ini lebih pada implementasi perangkat lunak pada sistem yang fokus pada implementasi perangkat lunak pada kode program. Tahapa ini menghasilkan produksi perangkat lunak dimana menjadi syarat dari *Initial Operational Milestone* atau batas/tonggak kemampuan operasional awal.

4. *Transition* (transisi)

Tahap ini lebih pada *deployment* atau instalasi sistem agar dapat dimengerti oleh *user*. Tahap ini menghasilkan produk perangkat lunak dimana menjadi syarat dari *Initial Oprtional Capability Milestone* atau batas/tonggak kemampuan awal. Aktifitas pada tahap ini termasuk pada pelatihan *user*, pemeliharaan dan pengujian sistem apakah sudah memenuhi harapan *user*.

1.7 Sistematika Penulisan

Sistematika yang dipaparkan pada laporan penelitian ini terbagi dalam beberapa bab yang dibahas, diantaranya adalah :

BAB I : PENDAHULUAN

Bab ini merupakan pengantar yang memberikan gambaran mengenai permasalahan-permasalahan yang kemudian akan dibahas pada bab-bab selanjutnya. Terdapat delapan pokok bahasan dalam bab ini, yaitu latar belakang, rumusan masalah, tujuan penelitian, manfaat penelitian, batasan masalah, kerangka pemikiran, metodologi penelitian dan sistematika penulisan.

BAB II : LANDASAN TEORI

Bab ini akan membahas tentang teori-teori yang digunakan dalam analisa permasalahan yang ada, dan juga teori-teori yang digunakan dalam perancangan dan implementasi.

BAB III : ANALISIS DAN PERANCANGAN

Bab ini membahas mengenai analisis dari permasalahan yang ada saat ini dan analisis kebutuhan yang diperlukan untuk mengatasi permasalahan tersebut.

Pembuatan desain dari sistem dengan mengacu pada analisis yang telah dibahas. Desain sistem yang akan dijelaskan terbagi menjadi tiga bagian, meliputi desain *user interface*, desain data dan desain proses.

BAB IV : IMPLEMENTASI DAN PENGUJIAN

Bab ini berisi tentang pengujian pada sistem. Dijelaskannya proses proses pengujian serta menganalisa kembali sistem dengan rancangan yang dibuat sebelumnya.

BAB V : PENUTUP

Bab ini berisi kesimpulan dan saran untuk pengembangan aplikasi lebih lanjut dalam upaya memperbaiki kelemahan pada aplikasi guna untuk mendapatkan hasil kinerja aplikasi yang lebih baik dan pengembangan program selanjutnya.