BAB 3
METODOLOGI PENELITIAN

3.1 Metode Penelitian

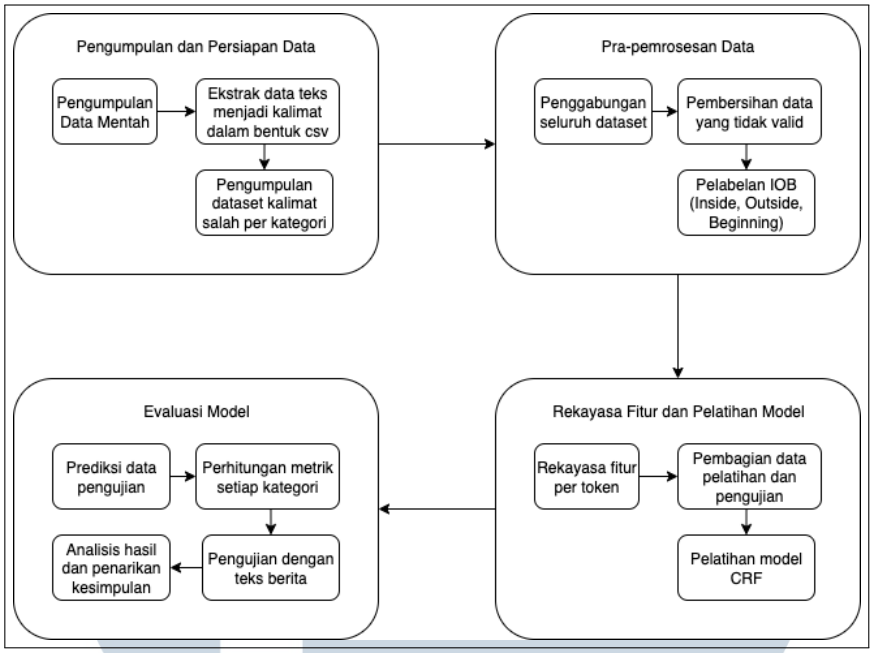
Penelitian ini menggunakan pendekatan kuantitatif dengan menerapkan metode *supervised machine learning* untuk membangun model deteksi kesalahan penggunaan angka dan bilangan. Secara spesifik, algoritma yang digunakan adalah *Conditional Random Fields* (CRF).

CRF dipilih karena kemampuannya yang unggul dalam tugas-tugas sequence labeling (pelabelan sekuensial). Deteksi kesalahan penulisan dalam sebuah kalimat merupakan masalah sekuensial, di mana label untuk sebuah kata (token) tidak hanya bergantung pada kata itu sendiri, tetapi juga pada kata-kata di sekitarnya (konteks). CRF mampu memodelkan ketergantungan ini dengan mempertimbangkan fitur dari token saat ini serta token sebelum dan sesudahnya dalam sebuah urutan. Hal ini membuat CRF sangat cocok untuk mengenali entitas kesalahan yang bisa terdiri dari satu atau beberapa kata, seperti frasa bilangan atau nama geografi.

Tujuan dari penerapan metode ini adalah untuk melatih model CRF agar dapat mengenali dan mengklasifikasikan delapan kategori kesalahan penulisan angka dan bilangan yang spesifik berdasarkan aturan Ejaan Yang Disempurnakan (EYD) dari teks berita berbahasa Indonesia.

3.2 Tahapan Penelitian

Tahapan penelitian ini dirancang secara sistematis mengikuti alur kerja standar dalam pengembangan model *machine learning*, mulai dari pengumpulan data hingga evaluasi akhir. Seluruh proses komputasi dalam penelitian ini, mulai dari pra-pemrosesan data, rekayasa fitur, hingga pelatihan model, dilakukan menggunakan platform Google Colab. Google Colab dipilih karena menyediakan lingkungan berbasis *cloud* yang mendukung eksekusi kode Python secara efisien tanpa memerlukan konfigurasi perangkat keras lokal yang kuat. Alur penelitian ini diilustrasikan pada diagram di Gambar 3.1.



Gambar 3.1. *Pipeline* Penelitian

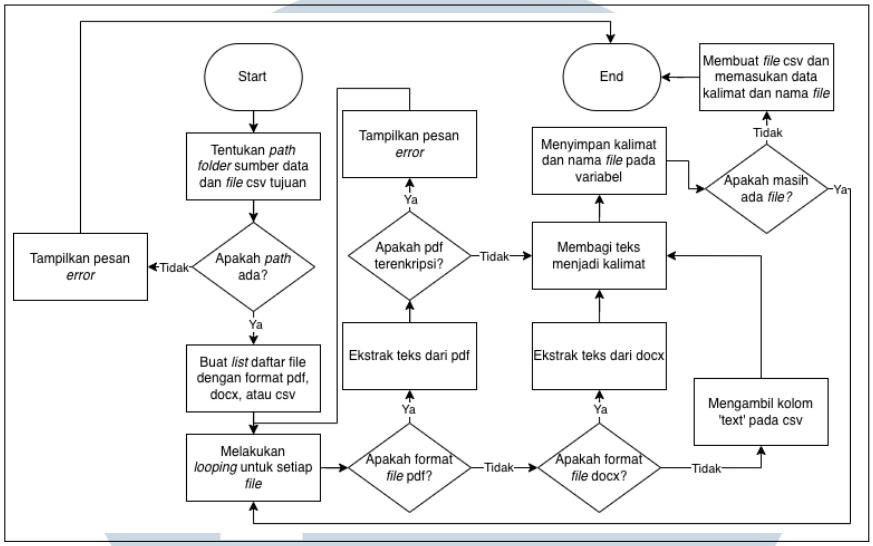
3.3 Pengumpulan dan Persiapan Data

Tahap ini merupakan fondasi dari penelitian, di mana data yang relevan dan berkualitas dikumpulkan untuk melatih dan menguji model. Proses ini terbagi menjadi beberapa langkah:

1. Pengumpulan Data Mentah: Data utama yang digunakan adalah kumpulan artikel berita berbahasa Indonesia dari berbagai sumber daring. Data mentah ini bervariasi dalam format, termasuk dokumen .docx, .pdf, dan .csv. Jumlah artikel yang berhasil dikumpulkan adalah 738.548 artikel.

2. Ekstraksi Kalimat: Dari data mentah tersebut, dilakukan ekstraksi untuk menghasilkan dataset berisi kalimat-kalimat individual. Pada tahap ini, sistem menerima *input* berupa kumpulan data mentah artikel berita berjumlah 738.548 dokumen yang tersimpan dalam berbagai format seperti .docx, .pdf, dan .csv. Proses komputasi dimulai dengan sistem membaca setiap dokumen secara iteratif, lalu memecah teks paragraf menjadi unit-unit kalimat terpisah menggunakan (*sentence tokenizer*). Sebagai *output*, tahap ini menghasilkan dokumen berformat .csv yang berisi daftar lengkap kalimat individual dari seluruh sumber berita yang siap untuk diproses lebih lanjut. Hal ini dilakukan

demi efisiensi pemrosesan selanjutnya. Proses dari ekstraksi kalimat dapat dilihat pada Gambar 3.2



Gambar 3.2. *Flowchart* Ekstraksi Kalimat

3. Pengumpulan Dataset Kalimat Salah: Tahap ini bertujuan untuk mengumpulkan dataset kalimat yang menyalahi aturan angka dan bilangan lengkap dengan kategori kesalahannya serta letak kesalahannya menggunakan pendekatan berbasis aturan (rule-based).

   (a) Dataset Salah: Pada tahap ini dilakukan proses penyaringan dataset, *input* yang digunakan adalah dokumen .csv berisi kalimat hasil ekstraksi sebelumnya. Sistem kemudian melakukan proses *scanning* terhadap setiap kalimat menggunakan pola *Regular Expression* (Regex) yang telah didefinisikan untuk sebelas kategori kesalahan angka dan bilangan. Jika sebuah kalimat cocok dengan pola kesalahan, sistem akan mengambil kalimat tersebut, memberikan label '1' yang artinya ini adalah kalimat salah, serta disimpan kata atau frasa yang menjadi letak kesalahannya. *Output* dari proses ini adalah dataset baru yang terkurasi dalam bentuk csv, berisi kalimat-kalimat yang terindikasi mengandung kesalahan lengkap dengan label kategori dan letak kesalahannya. Terdapat delapan kategori kesalahan (K01-K08) yang diidentifikasi:

      i. K01_AWAL_KALIMAT: Penggunaan angka di awal kalimat.

      ii. K02_1_ANGKA_VS_HURUF: Kesalahan penulisan bilangan yang dapat dinyatakan satu kata.

iii. K02_2_ANGKA_VS_HURUF: Kesalahan penulisan bilangan yang dinyatakan lebih dari satu kata.

iv. K02_3_ANGKA_VS_HURUF: Kesalahan penulisan bilangan yang dinyatakan secara berurutan. Untuk mempermudah model dalam mempelajari kategori ini, dilakukan pemecahan kategori menjadi tiga jenis:

   A. K02_FIRST_3_ANGKA_VS_HURUF: Bilangan pertama dalam kalimat.

   B. K02_3_ANGKA_VS_HURUF: Bilangan yang berada diantara bilangan pertama dan terakhir dalam kalimat.

   C. K02_LAST_3_ANGKA_VS_HURUF: Bilangan terakhir dalam kalimat.

v. K03_BILANGAN_BESAR: Penulisan bilangan besar yang dinyatakan dengan angka saja.

vi. K04_ALAMAT: Kesalahan penulisan angka sebagai bagian dari alamat dengan huruf.

vii. K05_PENOMORAN_BUKU: Kesalahan penulisan nomor bagian karangan atau kitab suci dengan huruf.

viii. K06_BILANGAN_TINGKAT: Penggunaan bilangan tingkat yang tidak sesuai aturan.

ix. K06_2_BILANGAN_TINGKAT: Kesalahan penulisan bilangan tingkat yang menggunakan awalan ke-.

x. K07_AKHIRAN_AN: Penulisan angka dengan akhiran ”-an” dengan kesalahan tanda hubung.

xi. K08_NAMA_GEOGRAFI: Kesalahan penulisan nama geografi yang mengandung bilangan.

(b) Augmentasi Data: Untuk kategori dengan jumlah data hasil scraping yang sedikit, dilakukan augmentasi data untuk menyeimbangkan jumlah data antar kategori yaitu dengan *generate* data dalam bentuk csv menggunakan ChatGPT. Hasil jumlah data per kategori dapat dilihat pada tabel 3.1.

Tabel 3.1. Jumlah Data per Kategori

Kategori	Jumlah Data
K01_AWAL_KALIMAT	32.490
K02_1_ANGKA_VS_HURUF	28.759
K02_2_ANGKA_VS_HURUF	30.028
K02_3_ANGKA_VS_HURUF	30.000
K03_BILANGAN_BESAR	35.329
K04_ALAMAT	33.257
K05_PENOMORAN_BUKU	30.000
K06_BILANGAN_TINGKAT	39.449
K06_2_BILANGAN_TINGKAT	30.000
K07_AKHIRAN_AN	30.818
K08_NAMA_GEOGRAFI	35.759

3.4 Pelabelan IOB

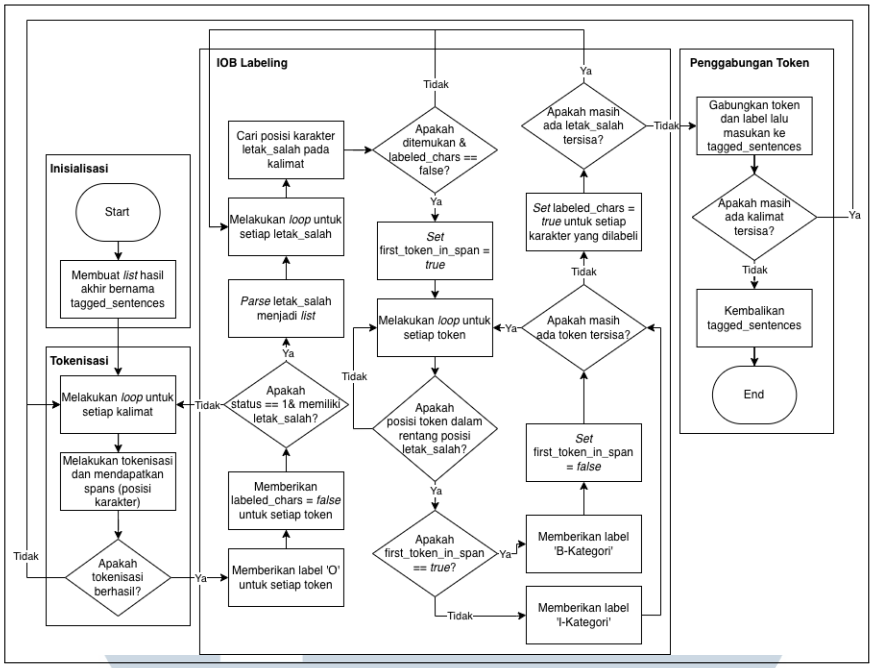
Setelah data terkumpul, tahap pelabelan IOB (*Inside, Outside, Beginning*) dilakukan untuk mengubah data menjadi format yang sesuai untuk model CRF. Skema pelabelan ini digunakan untuk menandai setiap token dalam kalimat:

1. B-KATEGORI: Menandai token pertama dari sebuah frasa kesalahan (contoh: B-K01_AWAL_KALIMAT).

2. I-KATEGORI: Menandai token yang berada di dalam frasa kesalahan, tetapi bukan token pertama (contoh: I-K04_ALAMAT).

3. O: Menandai token yang berada di luar frasa kesalahan (token yang benar).

Tahap pelabelan IOB memproses *input* berupa dataset kalimat yang telah teridentifikasi mengandung kesalahan beserta informasi letak kesalahannya. Dalam proses ini, sistem memecah kalimat menjadi token kata per kata, kemudian memberikan label berdasarkan letak kesalahannya. Token di awal frasa kesalahan diberi label 'B' (Beginning), token di dalam frasa diberi label 'I' (Inside), dan token lainnya diberi label 'O' (Outside) . Hasil akhirnya berupa *output* data sekuensial di mana setiap token kata telah memiliki pasangan label IOB-nya masing-masing yang siap digunakan untuk ekstraksi fitur. Proses dari IOB Labeling yang dilakukan, dapat dilihat pada Gambar 3.3.

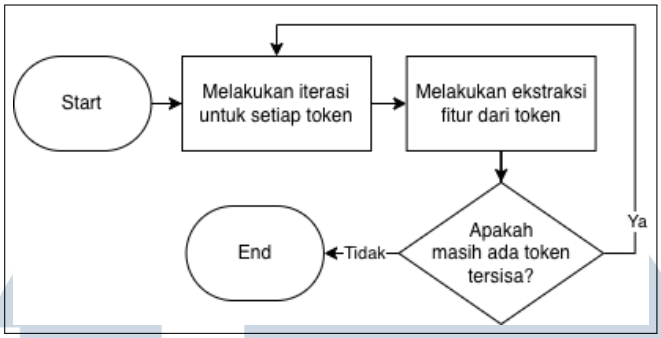


Gambar 3.3. *Flowchart IOB Labelling*

3.5 Rekayasa Fitur & Pembagian data

Pada tahap ini setiap token akan dilakuakn rekayasa fitur. Fitur-fitur ini nantinya akan digunakan digunakan untuk melatih model CRF.

1. Rekayasa Fitur (*Feature Engineering*): Pada tahap rekayasa fitur, *input* yang diterima adalah daftar token dan label hasil dari tahap pelabelan IOB. Sistem kemudian menjalankan proses ekstraksi untuk menganalisis karakteristik linguistik dan kontekstual dari setiap token, seperti memeriksa pola digit, sufiks kata, serta fitur dari token sebelum dan sesudahnya (context window) dengan metode *rule-based*. *Output* yang dihasilkan adalah sekumpulan vektor fitur yang merepresentasikan karakteristik setiap token dalam bentuk struktur data matematis agar pola kesalahannya dapat dipelajari oleh model. *Flowchart* rekayasa fitur dapat dilihat pada Gambar 3.4 dan fitur-fitur yang dibuat dapat dilihat pada Tabel 3.2.



Gambar 3.4. *Flowchart* Rekayasa Fitur

Tabel 3.2. Tabel Fitur

Nama Fitur	Deskripsi Fitur
bias	Fitur bias konstan (selalu 1.0).
word.isdigit()	Bernilai True jika seluruh token adalah digit (angka).
word.suffix_3	Tiga karakter terakhir dari token.
word.is_number_word	True jika token adalah kata bilangan (mis: ’dua’, ’sepuluh’).
pos_in_sentence	Posisi relatif token dalam kalimat (skala 0.0 hingga 1.0).
word.contains_separator	True jika token mengandung ’.’ atau ’,’ dan sisanya digit.
is_address_keyword	True jika token adalah kata kunci alamat (mis: ’jalan’, ’rt’).
is_level_keyword	True jika token adalah kata kunci tingkatan (mis: ’kelas’, ’lantai’).
is_book_keyword	True jika token adalah kata kunci buku (mis: ’jilid’, ’bab’).
word_is_ke	True jika token adalah ’ke’ atau ’ke-’.
is_long_number	True jika token adalah angka ¿ 3 digit.
word_is_an	True jika token adalah ’an’.

Lanjut pada halaman berikutnya

Tabel 3.2 Tabel Fitur (lanjutan)

Nama Fitur	Deskripsi Fitur
is_roman_number	True jika token adalah angka Romawi.
word_is_month	True jika token adalah nama bulan (mis: 'Januari', 'Febuari').
sent_contains_address_keyword	True jika token adalah kata kunci alamat (mis: 'jalan', 'rt').
word.contains_hyphen	True jika token mengandung tanda hubung ('-').
word.startswith_ke_joined	True jika token diawali 'ke' tanpa spasi (mis: 'kedua').
ke_joined_stem_contains_number	Jika (startswith_ke_joined=True), True jika stem-nya kata bilangan.
ke_joined_stem_contains_roman_number	Jika (startswith_ke_joined=True), True jika stem-nya angka Romawi.
word.startswith_ke_hyphen	True jika token diawali 'ke-' (mis: 'ke-12').
ke_hyphen_stem_is_digit	Jika (startswith_ke_hyphen=True), True jika stem-nya digit.
ke_hyphen_stem_is_number_word	Jika (startswith_ke_hyphen=True), True jika stem-nya kata bilangan.
ke_hyphen_stem_contains_roman_number	Jika (startswith_ke_hyphen=True), True jika stem-nya angka Romawi.
could_be_numeric_list	True jika token kata bilangan DAN kalimat terdeteksi punya list.
numberic_list:has_separator_after	True jika ada separator list (mis: ',') setelah token ini.
numberic_list:has_separator_before	True jika ada separator list (mis: ',') sebelum token ini.
numberic_list:has_separator_between	True jika token berada di antara dua separator list.
is_first_number_word_in_list	True jika token adalah kata bilangan pertama dalam list.
Lanjut pada halaman berikutnya	

Tabel 3.2 Tabel Fitur (lanjutan)

Nama Fitur	Deskripsi Fitur
is_last_number_word_in_list	True jika token adalah kata bilangan terakhir dalam list.
is_middle_number_word_in_list	True jika token adalah kata bilangan di tengah list.
-1:word.isdigit()	True jika token[i-1] adalah digit.
-1:is_number_word	True jika token[i-1] adalah kata bilangan.
-1:word_is_ke	True jika token[i-1] adalah 'ke' atau 'ke-'.
-1:is_address_keyword	True jika token[i-1] adalah kata kunci alamat.
-1:is_level_keyword	True jika token[i-1] adalah kata kunci tingkatan.
-1:is_book_keyword	True jika token[i-1] adalah kata kunci buku.
prev_is_ke+word_is_digit	(Jika token[i-1] 'ke'/'ke-'), True jika token[i] digit.
prev_is_ke+word_is_number_word	(Jika token[i-1] 'ke'/'ke-'), True jika token[i] kata bilangan.
prev_is_digit+an	(Jika token[i] 'an'), True jika token[i-1] digit.
prev_is_number_word+an	(Jika token[i] 'an'), True jika token[i-1] kata bilangan.
+1:word.isdigit()	True jika token[i+1] adalah digit.
+1:is_number_word	True jika token[i+1] adalah kata bilangan.
+1:word_is_an	True jika token[i+1] adalah 'an'.
+1:word_is_month	True jika token[i+1] adalah nama bulan.
ke+next_is_digit	(Jika token[i] 'ke'/'ke-'), True jika token[i+1] digit.
Lanjut pada halaman berikutnya	

Tabel 3.2 Tabel Fitur (lanjutan)

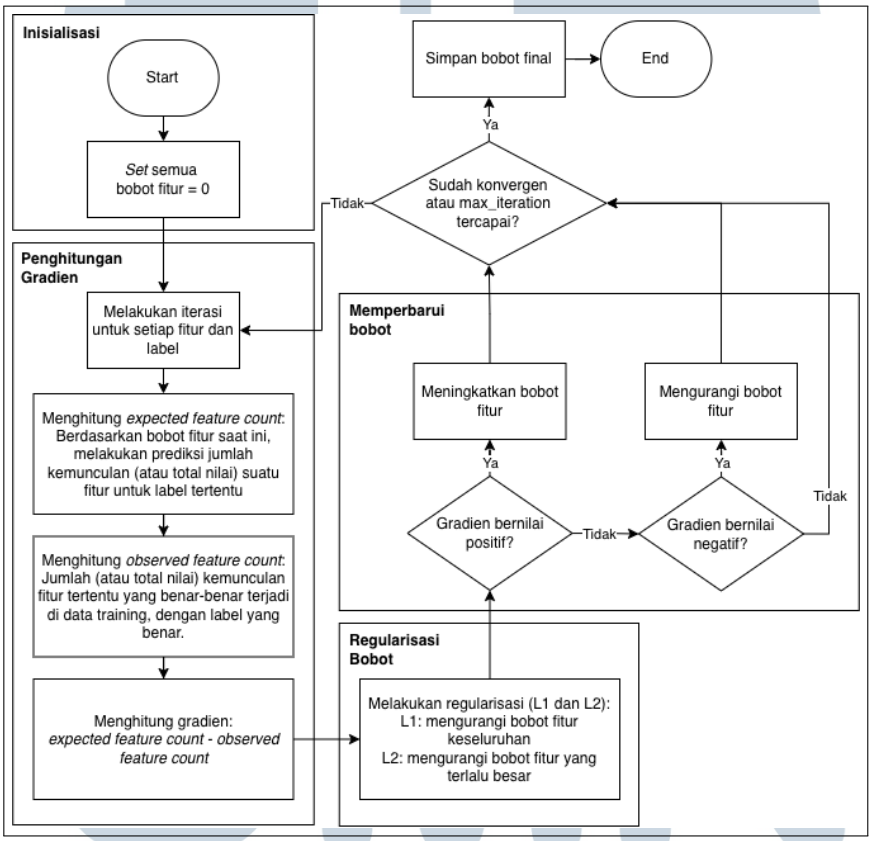
Nama Fitur	Deskripsi Fitur
ke+next_is_number_word	(Jika token[i] 'ke'/'ke-'), True jika token[i+1] kata bilangan.
BOS	True jika token adalah token pertama (Beginning of Sentence).
EOS	True jika token adalah token terakhir (End of Sentence).
k04_error_number_word	True jika kalimat memiliki kata kunci alamat dan token adalah angka dalam bentuk kata.
k04_error_roman_with_nomor	True jika token adalah angka romawi dan token sebelumnya adalah 'nomor' atau 'no.'.
k04_error_digit_without_nomor_specific	True jika token adalah angka arab dan token sebelumnya bukan 'nomor' atau 'no.'.
k04_error_digit_without_nomor_general	True jika token adalah angka arab, token sebelumnya bukan 'nomor' atau 'no.', dan kalimat memiliki kata kunci alamat.

2. Pembagian Data: Dataset yang telah diproses dibagi menjadi dua bagian: 80% sebagai data latih (*training set*) dan 20% sebagai data uji (*testing set*). Pembagian ini dilakukan secara acak untuk memastikan kedua set data merepresentasikan distribusi data keseluruhan.

3.6 Pelatihan Model

Tahap ini adalah inti dari proses pemodelan *machine learning*, model CRF secara iteratif belajar dengan menggunakan data latih yang sudah kaya akan fitur. Model CRF berusaha mencari bobot terbaik, bobot tersebut terdapat dua jenis, yaitu bobot fitur (seberapa kuat suatu fitur dapat memprediksi label tertentu) dan bobot transisi (seberapa besar kemungkinan satu label diikuti oleh label lain). Dalam

tahap pelatihan model, *input* utamanya adalah data latih yang telah dikonversi menjadi vektor fitur beserta labelnya. Proses pelatihan dijalankan di Google Colab, di mana algoritma CRF melakukan optimasi iteratif menggunakan metode L-BFGS untuk menghitung bobot terbaik yang menghubungkan fitur dengan label. Tahapan ini menghasilkan *output* berupa *file* model CRF yang telah terlatih (*pre-trained model*) yang disimpan dalam format joblib untuk digunakan pada tahap inferensi selanjutnya. *Flowchart* dari cara kerja pelatihan model CRF dapat dilihat pada Gambar 3.5.



Gambar 3.5. *Flowchart* pelatihan model CRF

Pelatihan dimulai dengan model yang belum mengetahui apa-apa, yaitu semua bobot bernilai nol. Setelah itu, dalam setiap iterasi model CRF akan mempelajari data latih dan membandingkan dua hal yaitu *expected feature count* dan *observed feature count*. *expected feature count* adalah frekuensi atau nilai (jika fiturnya dalam bentuk angka) suatu fitur terhadap label tertentu dan transisi dari suatu label ke label lain yang diprediksi oleh model berdasarkan bobot saat ini. *observed feature count* adalah frekuensi atau nilai yang sebenarnya yang dihitung dari data latih. Perbedaan antara *expected feature count* dan *observed feature count* adalah gradien

yang digunakan model sebagai sinyal mengenai bagaimana bobot fitur dan transisi harus diperbaiki.

Untuk mencegah *overfitting*, perlu dilakukan regularisasi bobot (L1 dan L2) yang melakukan *penalty* pada bobot agar nilainya tidak terlalu ekstrem. L2 (c2) cenderung mengecilkan bobot-bobot yang terlalu dominan, sementara L1 (c1) mengurangi bobot secara umum dan dapat mendorong bobot fitur yang tidak berguna menjadi nol.

Dengan algoritma optimasi L-BFGS, dilakukan penyesuaian bobot sedikit demi sedikit ke arah yang lebih baik dan memperkecil gradien. Penyesuaian bobot dilakukan dengan mempertimbangkan nilai gradien dan *penalty* dari L1 dan L2. Proses memprediksi, membandingkan, menerapkan *penalty*, dan menyesuaikan bobot ini dilakukan sampai nilai bobot konvergen atau mencapai jumlah iterasi maksimum.

3.7 Evaluasi Model

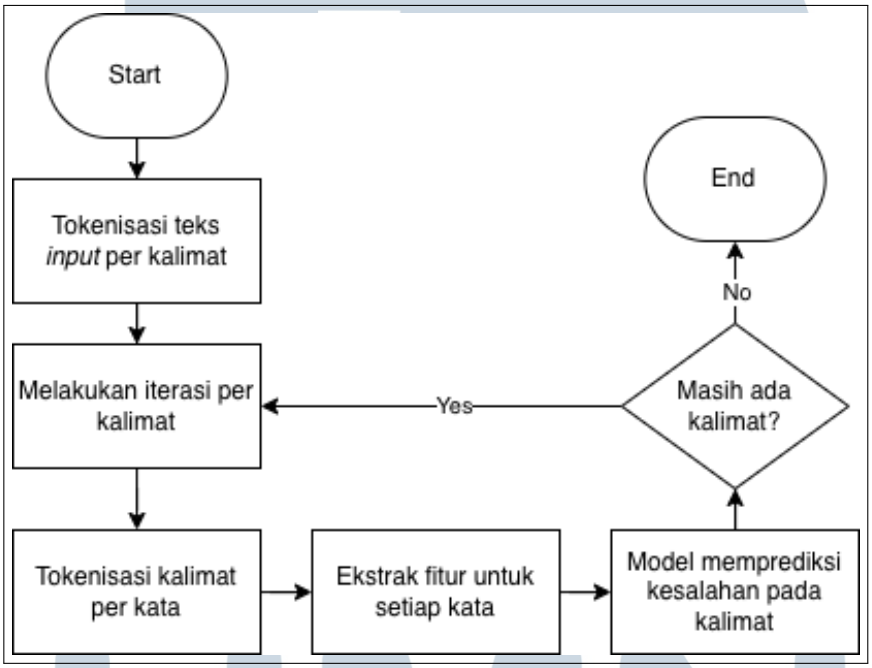
Tahap terakhir adalah mengevaluasi performa model yang telah dilatih menggunakan data uji yang belum pernah dilihat sebelumnya. Evaluasi ini bertujuan untuk mengukur seberapa baik model dapat menggeneralisasi pengetahuannya pada data baru. Metrik yang digunakan adalah:

1. Akurasi Keseluruhan: Persentase token yang labelnya berhasil diprediksi dengan benar.
2. *Precision*: Kemampuan model untuk tidak salah melabeli sebuah token yang benar sebagai token yang salah.
3. *Recall*: Kemampuan model untuk menemukan semua token yang salah di dalam data.
4. *F1-Score*: Rata-rata harmonik dari precision dan recall, memberikan gambaran performa yang seimbang.
5. *Confusion Matrix*: Matriks yang digunakan untuk menganalisis kesalahan prediksi model secara lebih detail, menunjukkan bagaimana model sering keliru antar kategori.

Hasil dari evaluasi ini kemudian dianalisis untuk menarik kesimpulan mengenai efektivitas metode CRF dalam mendeteksi kesalahan penggunaan angka dan bilangan pada teks berita.

3.8 Inferensi Model

Setelah model Conditional Random Fields (CRF) selesai dilatih dan dievaluasi, langkah penting berikutnya adalah menerapkan model tersebut untuk melakukan inferensi, yaitu proses ketika model memberikan prediksi terhadap teks baru yang tidak termasuk dalam data latih dan data uji. *Flowchart* dari proses inferensi dapat dilihat pada Gambar 3.6.



Gambar 3.6. *Flowchart* inferensi model CRF

Proses dimulai dari tahap tokenisasi teks, di mana *input* paragraf dipecah terlebih dahulu menjadi data kalimat, hal ini perlu dilakukan karena model CRF dilatih dengan data kalimat individual bukan paragraf, sehingga hanya mampu memprediksi satu kalimat dalam satu waktu. Setelah terbentuk kumpulan kalimat, sistem memasuki proses iterasi yang memeriksa setiap kalimat secara berurutan. Pada setiap iterasi, kalimat yang sedang diproses kemudian ditokenisasi kembali menjadi kata-kata individual. Setiap token yang dihasilkan akan melalui tahap ekstraksi fitur yang akan digunakan oleh model sebagai acuan untuk melakukan deteksi. Fitur-fitur token tersebut kemudian dikirimkan ke model *machine learning* untuk memprediksi apakah terdapat kesalahan pada kalimat tersebut beserta kategori kesalahannya.

3.9 Implementasi *API*

Agar model yang telah dilatih dapat diakses oleh aplikasi U-Tapis, dilakukan tahap pembuatan *API* menggunakan *back-end framework* Flask. Ketika API ini berjalan, sistem menerima input berupa *request* JSON berisi teks kalimat berita. Proses di dalam *API* mencakup tokenisasi teks input, ekstraksi fitur yang, dan prediksi label menggunakan model CRF yang telah dimuat sama persis dengan tahap inferensi. Hasil prediksi tersebut kemudian disusun ulang dan dikirimkan sebagai *output* respons JSON yang berisi informasi ada atau tidaknya kesalahan dan label untuk setiap tokennya.