1. Problem Statement

The aim of this project was to build a **Named Entity Recognition (NER)** model using **Conditional Random Fields (CRF)** to extract structured information from raw recipe ingredient text.
The extracted entities include:

- Quantities (e.g., "2", "1/2")
- Units (e.g., "cups", "tablespoon")
- Ingredients (e.g., "rice", "turmeric powder")

This structured data can then support recipe management systems, dietary apps, or e-commerce features.

2. Key Methodology

Data Preparation

- Input data was provided in JSON format.
- Each sample had:
    - input: raw ingredient list as a text string.
    - pos: corresponding labels (NER tags) like quantity, unit, ingredient.

Data was read and processed into a **pandas DataFrame**, splitting text and labels into tokens.

Feature Engineering

The CRF model requires manual feature extraction from each token.
Features included:

- Lowercase token
- Token suffixes/prefixes
- Is the token title case
- Is the token a digit
- Previous and next tokens' features (context)

This context-aware design helps the CRF capture patterns in sequences.

Model Training

Used **sklearn-crfsuite**, a Python wrapper over CRFsuite.

- Data was split into training and validation sets.

- Model trained to map token-level features to NER labels.

Evaluation

Measured:

- Overall accuracy
- Label-wise accuracy

Analyzed common errors:

- Misclassification among similar labels (e.g., ingredient vs. unit)

Identified boundary cases and improvement opportunities:

- Data imbalance for certain labels
- Need for richer features (e.g., word embeddings)

3. Overall Model Performance:

- The CRF model achieved an overall accuracy of ≈ 99.5% on the validation dataset.
- This shows the model can correctly predict the majority of tokens, but there is still room for improvement, especially on certain labels.

Label-wise Accuracy & Errors:

- Labels with higher frequency in the data (e.g., 'O' or background tokens) tend to have higher accuracy, as the model sees more examples during training.
- Labels that are less frequent (e.g., 'B-ingredient' or 'I-ingredient') typically have lower accuracy and higher error rates, even if class weights were applied.
- The table below (from your analysis_df) shows:
  - Total tokens per label
  - Number of errors
  - Accuracy per label
  - Applied class weights

Impact of Class Weights:

- Applying weight_dict helped slightly improve accuracy on minority classes.
- However, very rare labels still suffer from misclassifications because the model sees too few examples.

Nature of Misclassifications:

- Most misclassified tokens tend to be:
  - Tokens at the beginning or end of entities, where boundary detection is harder.
  - Tokens that look similar to non-entity tokens (e.g., ingredient words that appear outside actual ingredient lists).

- From error_data, we see that misclassified tokens often have preceding or next tokens that are ambiguous.

Context and Sequence Errors:

- Some errors occur when the model predicts correct labels locally but fails to learn correct sequences of labels (e.g., 'B-ingredient' directly followed by another 'B-ingredient' instead of 'I-ingredient').

4. Boundary Cases:

- In named entity recognition (NER) or sequence labelling with CRF, boundary cases refer to situations where the model struggles to correctly detect the start and end of an entity span.

Typical boundary issues include:

- Predicting 'O' instead of 'B-ingredient' at the start of an entity.
- Failing to continue an 'I-ingredient' label in a multi-word entity, so only the first token is tagged.
- Predicting an extra 'B-ingredient' inside what should be a single entity span.

5. Improvement opportunities

a. Better feature engineering:
- Include features that help detect entity starts and ends, e.g.:
  - POS tags (e.g., nouns often start entities)
  - Capitalization and punctuation (commas or conjunctions often indicate boundaries)
  - Whether the token is the first or last in a sentence or list
- Use more context: previous two tokens, next two tokens.

b. Use richer models:
- BiLSTM-CRF or Transformer-based models capture longer dependencies and context.
- CRF alone is limited to features you define; neural models learn representations automatically.

c. Post-processing rules:
- Fix obvious inconsistencies, e.g.:
  - Remove isolated 'I-ingredient' not preceded by 'B-ingredient'.
  - Merge adjacent 'B-ingredient' tokens if likely part of same entity.

d. Address class imbalance:
- Further adjust class weights.
- Augment data for minority classes ('B-ingredient'/'I-ingredient').

e. Review and refine annotation:
- Check if inconsistent labelling in training data causes boundary confusion.
- Ensure consistent use of 'B-' and 'I-' tags.