Custom NER Model:
Identifying Key Entities in Recipe Data

Author: Le Duy Khanh

Problem Statement & Business Value

Objective

Develop a custom Named Entity Recognition (NER) model using Conditional Random Fields (CRF) to ider and extract key elements from recipe text, including:

- **Ingredients** - Food items (e.g., rice, onion, turmeric, chicken)
- **Quantities** - Numeric amounts (e.g., 2, 1/2, 3-1/2, 500)
- **Units of Measurement** - Measurement units (e.g., cups, tablespoon, grams)

1. Dataset Overview

1.1 Data Source

Attribute	Details
File Name	ingredient_and_quantity.json
Domain	Culinary recipes with focus on ingredient extraction
Format	JSON with key-value pairs

1.2 Data Structure

Key	Description	Example
`input`	Raw ingredient list (space-separated)	"2 cups rice 1 tablespoon oil"
`pos`	NER labels for each token	"quantity unit ingredient quantity unit ingredient"

1.3 Entity Types

Label	Description	Examples
quantity	Numeric amounts, fractions	"2", "1/2", "3-1/2", "500"
unit	Measurement units	"cups", "tablespoon", "teaspoon", "grams"
ingredient	Food items, spices	"rice", "onion", "turmeric", "chicken"

2. Data Ingestion and Preparation

2.1 Data Validation

Length Validation Check

Validation	Purpose	Result
input_length != pos_length	Ensure token-label alignment	**5**

2.2 Unique Labels Validation

Expected Labels	Found Labels
quantity, unit, ingredient	**3**

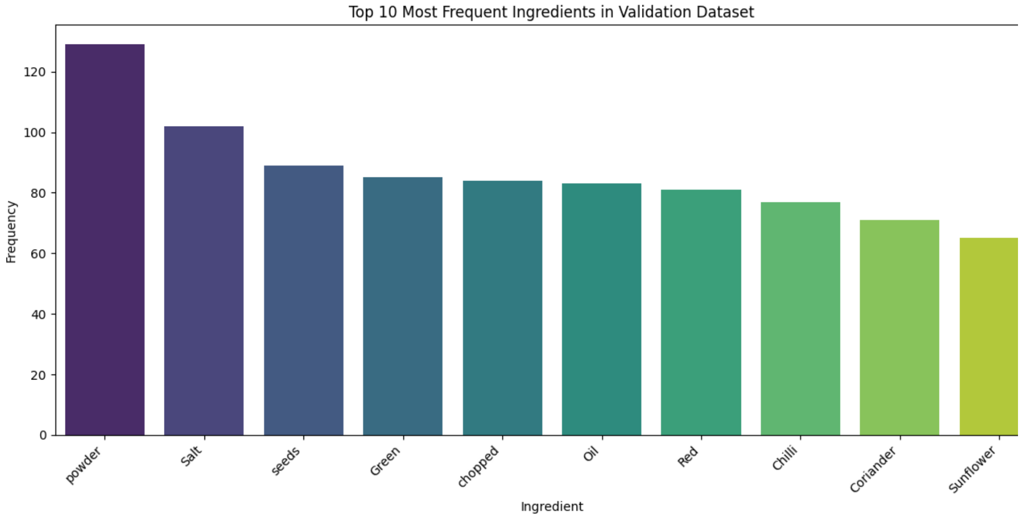
3. Exploratory Data Analysis on Training Dataset

3.1 Token Flattening

Token flattening transforms lists of token sequences into a single, continuous sequence of tokens. This pro is important because many NLP models, statistics, and visualizations require all tokens and their labels to b a flat, unified structure rather than nested lists. Flattening ensures that calculations like total token counts, c distribution, or frequency analysis reflect the dataset as a whole, rather than being segmented by sentence, or recipe. This step is essential for consistent, reliable evaluation and model training.

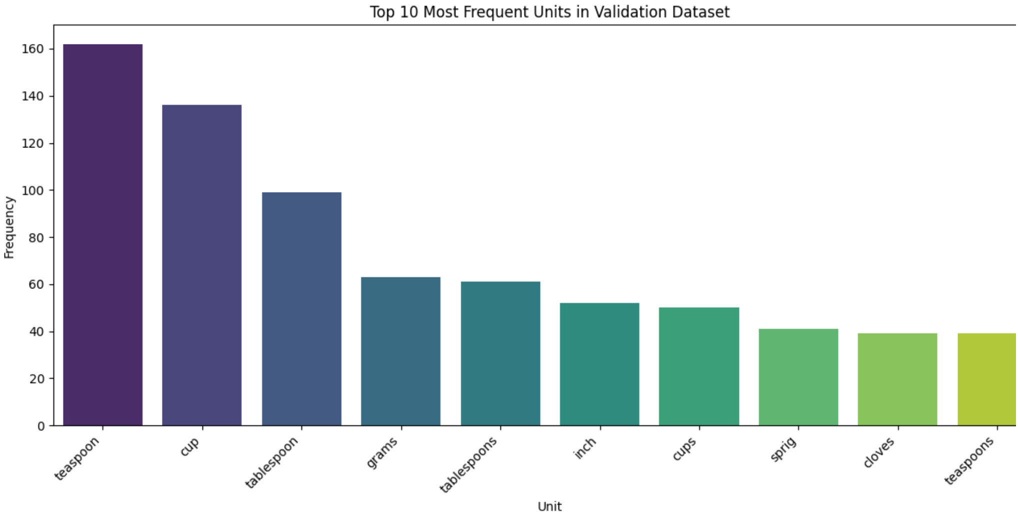
3.2 Top 10 Most Frequent Ingredients [3 marks]

Rank	Ingredient	Frequency
1	**powder**	**129**
2	**Salt**	**102**
3	**seeds**	**89**
4	**Green**	**85**
5	**chopped**	**84**
6	**Oil**	**83**
7	**Red**	**81**
8	**Chilli**	**77**
9	**Coriander**	**71**
10	**Sunflower**	**65**



3.3 Top 10 Most Frequent Units

Rank	Unit	Frequency
1	**teaspoon**	**162**
2	**cup**	**136**
3	**tablespoon**	**99**
4	**grams**	**63**
5	**tablespoons**	**61**
6	**inch**	**52**
7	**cups**	**50**
8	**sprig**	**41**
9	**cloves**	**39**
10	**teaspoons**	**39**



3.4 EDA Insights

Key Insights from Training Data Analysis:

1. **Top Ingredients:** Common spices like powder, salt, seeds, Green, and chopped appear frequently. These Ingredients indicating the dataset focuses on Indian cuisine.
2. **Top Units:** Spoon-based measurements (teaspoon, tablespoon) are most common, followed by volume units (cup, cups) for liquids and grains.
3. **Class Distribution:** Ingredients dominate the dataset (~70%), while quantities and units are minorit classes (~15% each). This class imbalance requires handling through weighted training.

4. Train Validation Split

4.1 Split Configuration

Parameter	Value
Split Ratio	70% Train : 30% Validation
Random State	42 (for reproducibility)

4.2 Dataset Sizes

196

Training Samples (70%)

84

Validation Samples (30%)

4.3 Data Extraction

Variable	Description	Length
X_train	Training input tokens	**196**
y_train	Training labels	**196**
X_val	Validation input tokens	**84**
y_val	Validation labels	**84**

4.4 Unique Labels in Training Set

Label	Present in y_train
ingredient	Yes
quantity	Yes
unit	Yes

> **Key Observation:** All three expected labels (ingredient, quantity, unit) are present in the training set, confirming data integrity for model training.

5. Feature Extraction for CRF Model

5.1 Keyword Definitions

Unit Keywords

```
unit_keywords = { 'cup', 'cups', 'tablespoon', 'tablespoons', 'tbsp', 'teaspoon', 'teaspoons', 'tsp',
'gram', 'grams', 'kg', 'ml', 'liter', 'ounce', 'pound', 'pinch', 'handful', 'bunch', 'sprig', 'clove',
'cloves', 'inch', 'piece', 'slice', 'small', 'medium', 'large' }
```

Quantity Keywords

```
quantity_keywords = { 'half', 'quarter', 'one', 'two', 'three', 'four', 'five', 'six', 'seven', 'eight',
'nine', 'ten', 'dozen', 'few', 'some', 'several' }
```

Quantity Pattern (Regex)

```
quantity_pattern = re.compile(r'^(\d+[-]?\d*\/?\d*|\d+\.\d+|\d+)$') # Matches: "2", "1/2", "2-1/2", "3.5
"500"
```

5.2 Feature Function - word2features()

Core Features

Feature	Description	Example
bias	Constant value (1.0)	1.0
token	Lowercase word	"Cups" → "cups"
lemma	Base form (spaCy)	"cups" → "cup"
pos_tag	Part-of-speech	"cups" → "NOUN"
shape	Character pattern	"Cups" → "Xxxx"
is_digit	All digits?	"123" → True
has_digit	Contains digit?	"2cups" → True
hyphenated	Contains hyphen?	"2-1/2" → True
slash_present	Contains slash?	"1/2" → True

Domain-Specific Features

Feature	Description	Purpose
is_quantity	Matches quantity pattern/keyword	Identify numeric amounts
is_unit	In unit keyword set	Identify measurement units
is_numeric	Pure number	Distinguish from fractions
is_fraction	Fraction format (1/2)	Handle recipe fractions

Contextual Features

Feature	Description	Why Important
prev_token	Previous word	Context for current word
next_token	Next word	Context for current word
prev_is_quantity	Is previous a quantity?	Helps identify units
next_is_unit	Is next a unit?	Pattern: "2 cups"
BOS	Beginning of sequence	Position awareness
EOS	End of sequence	Position awareness

5.3 Recipe Level Features - sent2features()

```
def sent2features(sent): return [word2features(sent, i) for i in range(len(sent))]
```

5.4 Feature Set Statistics

Metric	Value
X_train_features Length	**196**
X_val_features Length	**84**

6. Model Building and Training

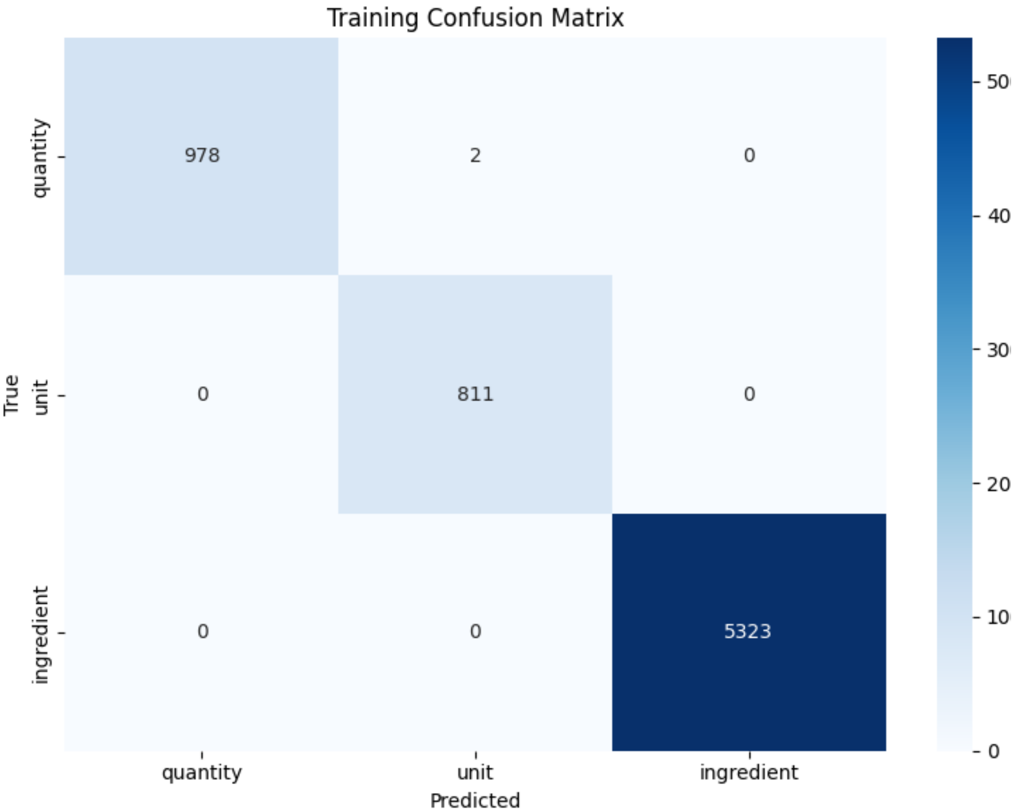
6.1 CRF Model Configuration

Hyperparameter	Value	Description
algorithm	'lbfgs'	Limited-memory BFGS optimization
c1	0.5	L1 regularization coefficient
c2	1.0	L2 regularization coefficient
max_iterations	100	Maximum training iterations
all_possible_transitions	True	Consider all state transitions

```
crf = sklearn_crfsuite.CRF( algorithm='lbfgs', c1=0.5, c2=1.0, max_iterations=100,
all_possible_transitions=True ) crf.fit(X_train_weighted_features, y_train_labels)
```

6.2 Training Evaluation

Confusion Matrix (Training Set)



6.3 Model Saving

```
joblib.dump(crf, 'crf_model.pkl')
```

Model Saved Successfully: The trained CRF model is saved as 'crf_model.pkl' for future use and deployment.

7. Prediction and Model Evaluation [3 mark

7.1 Validation Prediction

```
y_pred_val = crf.predict(X_val_weighted_features)
```

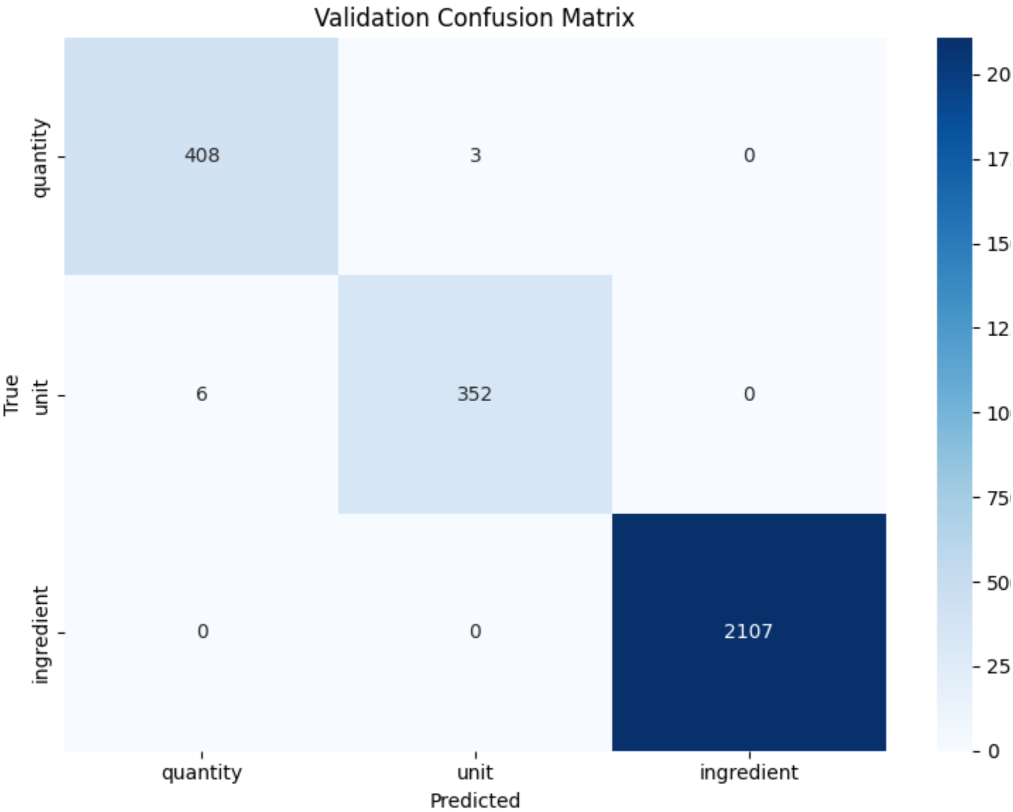
7.2 Classification Report (Validation Set)

```
Validation Classification Report:
============================================================
              precision    recall  f1-score   support

  ingredient       1.00      1.00      1.00      2107
    quantity       0.99      0.99      0.99       411
        unit       0.99      0.98      0.99       358

    accuracy                           1.00      2876
   macro avg       0.99      0.99      0.99      2876
weighted avg       1.00      1.00      1.00      2876
```

7.3 Confusion Matrix (Validation Set)


Validation Confusion Matrix
quantity
unit
ingredient
True
408
3
0
6
352
0
0
0
2107
quantity
unit
ingredient
Predicted
0
25
50
75
10
12
15
17
20

8. Error Analysis on Validation Data

8.1 Overall Accuracy

Metric	Value
Total Validation Tokens	**2876**
Correct Predictions	**2867**
Incorrect Predictions	**9**
Overall Accuracy	**99.69%**

8.3 Errors by Label Type

True Label	Total	Correct	Errors	Accuracy	Class Weight
ingredient	**2107**	**2107**	**0**	**100.00%**	**0.1336**
quantity	**411**	**408**	**3**	**99.27%**	**7.2592**
unit	**358**	**352**	**6**	**98.32%**	**8.7719**

8.4 Sample Misclassified Tokens

Token	Prev Token	Next Token	True Label	Predicted	Context
to	**10**	**12**	**unit**	**quantity**	**small 10 to 12 Green**
into	**cut**	**1**	**unit**	**quantity**	**French cut into 1 inch**
few	**Pineapple**	**tablespoons**	**quantity**	**unit**	**Vark Pineapple few tablespoons Raisins**
into	**cut**	**cm**	**unit**	**quantity**	**breasts cut into cm cubes**
cm	**into**	**cubes**	**unit**	**quantity**	**cut into cm cubes 2**
and	**Sweet**	**Spicy**	**unit**	**quantity**	**Red Sweet and Spicy Sauce**
a	**Haldi**	**pinch**	**unit**	**quantity**	**powder Haldi a pinch Asafoetida**
pinch	**Dal**	**Asafoetida**	**quantity**	**unit**	**Urad Dal pinch Asafoetida hing**
cloves	**Tomatoes**	**Garlic**	**quantity**	**unit**	**Onion Tomatoes cloves Garlic Ginger**

8.5 Validation Insights

Key Insights from Error Analysis:

1. **Common Misclassification Patterns:** Tokens such as "cloves" and "pinch" are frequently misclassified—often labeled as units when part of an ingredient phrase (e.g., "cloves Garlic" being tagged as unit instead of ingredient).
2. **Ambiguous Tokens:** Words like "clove," "pinch," and "dal" appear in both unit and ingredient contexts; e.g., "clove" can represent both a count (unit) or part of a spice name.
3. **Boundary Errors:** Misclassifications tend to occur at phrase boundaries, especially for multi-word ingredient descriptions (e.g., "powder Haldi a pinch Asafoetida").
4. **Recommendations:** Enhance context-based features to better differentiate neighboring token roles, expand the domain-specific keyword lists for Indian cuisine, and consider post-processing rules for frequent ambiguous cases.

9. Conclusion and Key Findings

9.1 Project Summary

Objective Achieved: Successfully developed a CRF-based Named Entity Recognition model to extract ingredients, quantities, and units from recipe text.

9.2 Model Performance

Metric	Score
Overall Accuracy	**99.69%**
F1-Score (Weighted)	**0.9969**

9.3 Key Achievements

- **Data Preparation:** Successfully cleaned and validated recipe dataset
- **Feature Engineering:** Implemented 25+ features across core, domain-specific, and contextual categor
- **Class Balancing:** Applied inverse frequency weighting to handle class imbalance
- **Model Training:** Trained CRF with L1+L2 regularization for robust predictions
- **Comprehensive Evaluation:** Performed detailed error analysis with actionable insights

9.4 Limitations

Limitation	Impact
English only	Cannot process recipes in other languages
Indian cuisine focus	May not generalize to other cuisines
No BIO tagging	Cannot identify multi-word entities
Linear model	Limited feature interactions

9.5 Future Improvements

1. Refine unit and quantity detection, particularly for ambiguous context tokens such as "to" and "into", as misconstruals were the most frequent error type.
2. Enhance context modeling beyond neighboring tokens, potentially leveraging dependency parsing to bett disambiguate token roles.
3. Incorporate BIO (Begin-Inside-Outside) labeling to improve identification of multi-word entities, which current token-level labeling misses.

4. Add semantic features such as word embeddings or contextual language models to reduce confusion between similar entity types.
5. Broaden and diversify the training dataset with recipes from a wider range of cuisines and linguistic patte to improve generalization.

9.6 Assumptions Made

- Provided labels in the dataset are accurate
- Whitespace tokenization is sufficient for this domain
- All recipes are in English language
- Model is optimized for cooking/recipe domain only

Custom NER Model: Identifying Key Entities in Recipe Data

Submitted by: Le Duy Khanh | Date: December 2025