Identifying Key Entities in Recipe Data

Business Objective:

The goal of this assignment is to train a Named Entity Recognition (NER) model using Conditional Random Fields (CRF) to extract key entities from recipe data. The model will classify words into predefined categories such as ingredients, quantities and units, enabling the creation of a structured database of recipes and ingredients that can be used to power advanced features in recipe management systems, dietary tracking apps, or e-commerce platforms.

1. Data Ingestion and Preparation

1.1. Loading the dataset

1.1.1. Import the dataset

1.1.1.1. There is a json file named ingredient_and_quantity.json which is imported and converted to the dataframe.

1.1.1.2. There are 285 rows and 2 columns(input and pos) in the dataset.

2. Recipe Data Manipulation

2.1.1. Create input_tokens and pos_tokens columns by splitting the input and pos from the dataframe

2.1.1.1. Both the input_tokens and pos_tokens are split by splitting on space character because the results are satisfactory

2.1.2. Provide the length for input_tokens and pos_tokens and validate their length

2.1.2.1. There are five rows where the length of input_tokens and pos_tokens are not same

2.1.3. Define a unique_labels function and validate the labels in pos_tokens

2.1.3.1. There are three unique levels i.e. ['ingredient', 'unit', 'quantity']

2.1.4. Provide the insights seen in the recipe data after validation

2.1.4.1. All the rows where the length of position tags and input tokens were removed as it contained some human data collection error

3. Train Validation Split

3.1.1. The data was split into a ratio of 70 percent training data and 30 percent validation data

3.1.2. It resulted into 196 training samples and 84 validation samples

3.1.3. The X_train data and Validation data was converted to list

3.1.4. It was made sure that all the labels are present in the training data

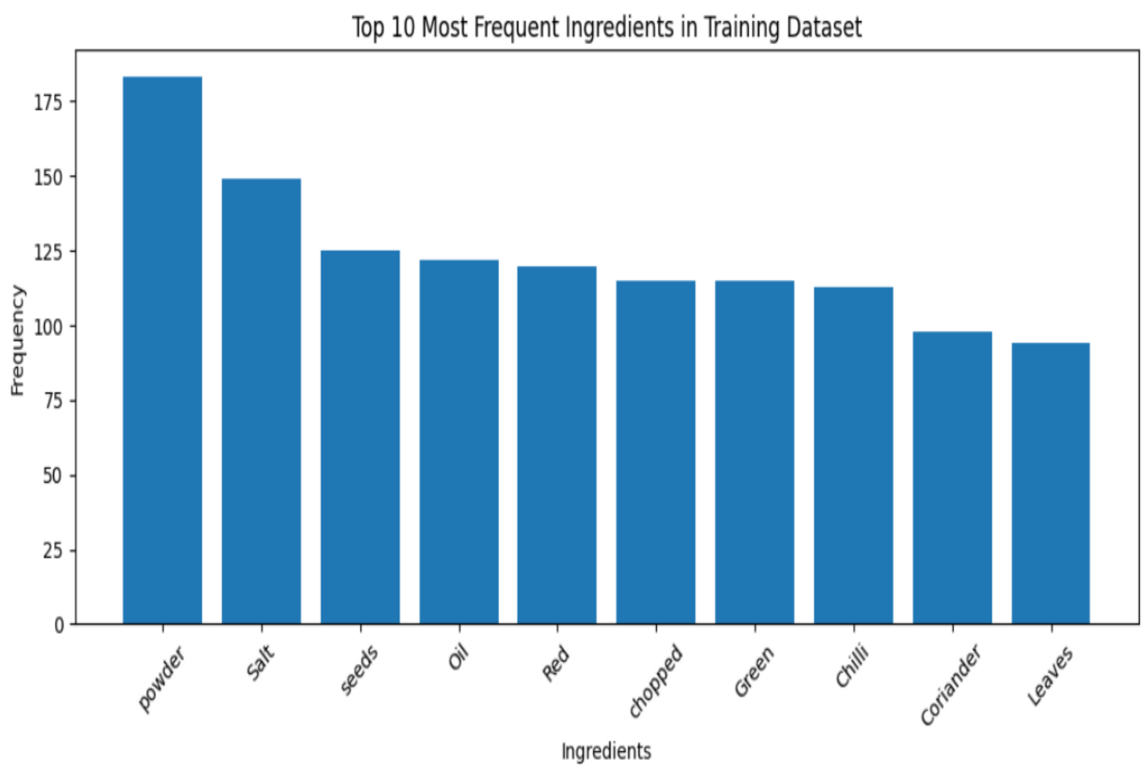
4. Exploratory Recipe Data Analysis on Training Dataset

4.1. To get the distribution of input tokens and pos tokens, both the lists are flattened

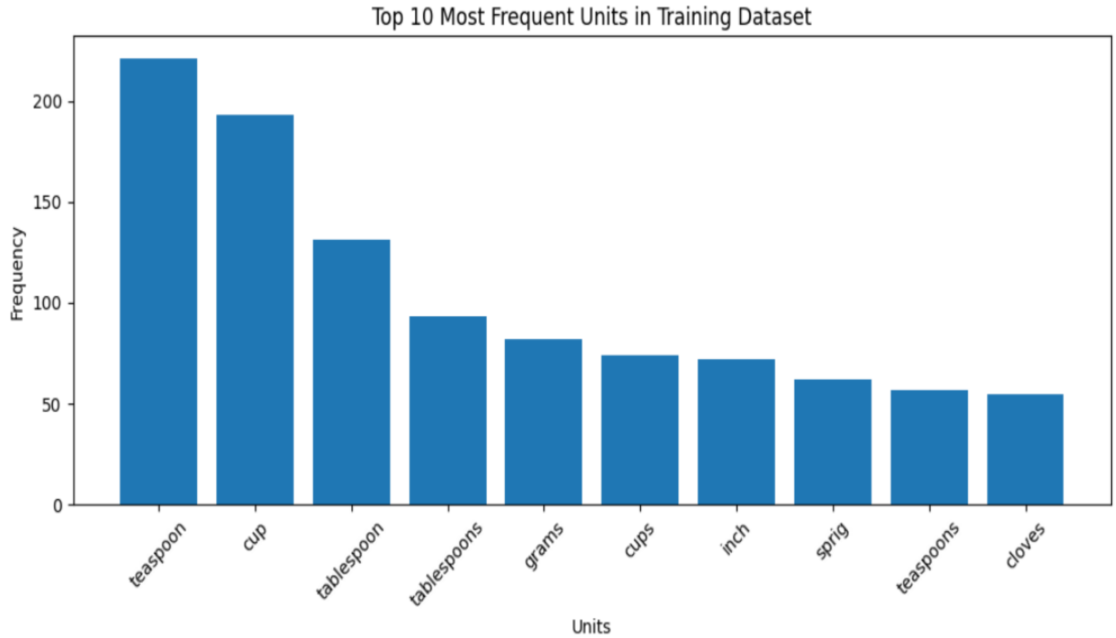
4.2. It was validated that the number of tokens and pos are same

4.3. Input tokens are categorised into ingredients, units and quantities

4.4. Top 10 items in category ingredient:



4.5. Top 10 items in category units



5. Feature Extraction for CRF model

5.1. Different features are created for each input token using spcy library

5.1.1. - \`bias\` - Constant feature with a fixed value of 1.0 to aid model learning.

5.1.2. - \`token\` - The lowercase form of the current token.

5.1.3. - \`lemma\` - The lowercase lemma (base form) of the token.

5.1.4. - \`pos_tag\` - Part-of-speech (POS) tag of the token.

5.1.5. - \`tag\` - Detailed POS tag of the token.

5.1.6. - \`dep\` - Dependency relation of the token in the sentence.

5.1.7. - \`shape\` - Shape of the token (e.g., "Xxx" for "Milk").

5.1.8. - \`is_stop\` - Boolean indicating if the token is a stopword.

5.1.9. - \`is_digit\` - Boolean indicating if the token consists of only digits.

5.1.10. - \`has_digit\` - Boolean indicating if the token contains at least one digit.

5.1.11. - \`has_alpha\` - Boolean indicating if the token contains at least one alphabetic character.

5.1.12. - `hyphenated` - Boolean indicating if the token contains a hyphen (-).

5.1.13. - `slash_present` - Boolean indicating if the token contains a slash (/).

5.1.14. - `is_title` - Boolean indicating if the token starts with an uppercase letter.

5.1.15. - `is_upper` - Boolean indicating if the token is fully uppercase.

5.1.16. - `is_punct` - Boolean indicating if the token is a punctuation mark.

5.1.17. - `is_quantity` - Boolean indicating if the token matches a quantity pattern or keyword.

5.1.18. - `is_unit` - Boolean indicating if the token is a known measurement unit.

5.1.19. - `is_numeric` - Boolean indicating if the token matches a numeric pattern.

5.1.20. - `is_fraction` - Boolean indicating if the token represents a fraction (e.g., 1/2).

5.1.21. - `is_decimal` - Boolean indicating if the token represents a decimal number (e.g., 3.14).

5.1.22. - `preceding_word` - The previous token in the sentence, if available.

5.1.23. - `following_word` - The next token in the sentence, if available.

5.1.24. - `prev_token` - The lowercase form of the previous token.

5.1.25. - `prev_is_quantity` - Boolean indicating if the previous token is a quantity.

5.1.26. - `prev_is_digit` - Boolean indicating if the previous token is a digit.

5.1.27. - `BOS` - Boolean indicating if the token is at the beginning of the sentence.

5.1.28. - `next_token` - The lowercase form of the next token.

5.1.29. - `next_is_unit` - Boolean indicating if the next token is a unit.

5.1.30. - `next_is_ingredient` - Boolean indicating if the next token is not a unit or quantity.

5.1.31. - `EOS` - Boolean indicating if the token is at the end of the sentence.

5.1.32. A new feature **class_weight** is created for each token:

5.1.32.1. To give priority to those categories which are less frequent

5.1.32.2. Frequency counts were calculated for each category and weight is given inversely proportion to the frequency counts

6. Model Building

6.1. **CRF model** was intialaised and fit on the data

6.2. Following hyperparameters were used while fitting the model:

6.2.1. algorithm='lbfgs' - Optimisation algorithm used for training. `lbfgs` (Limited-memory Broyden–Fletcher–Goldfarb–Shanno) is a quasi-Newton optimisation method. |

6.2.2. c1=0.5 - L1 regularisation term to control sparsity in feature weights. Helps in feature selection. |

6.2.3. c2=1.0 - L2 regularisation term to prevent overfitting by penalising large weights. |

6.2.4. max_iterations=100 - Maximum number of iterations for model training. Higher values allow more convergence but increase computation time. |

6.2.5. all_possible_transitions=True - Ensures that all possible state transitions are considered in training, making the model more robust.

7. Evaluation of Training dataset using CRF model

7.1. Classification Report

7.1.1. The results are pretty good at the training data with most of the metrics being close to 1

	precision	recall	f1-score	support
ingredient	1.00	1.00	1.00	5100
quantity	0.99	0.98	0.99	958
unit	0.98	0.99	0.98	822
accuracy			1.00	6880
macro avg	0.99	0.99	0.99	6880
weighted avg	1.00	1.00	1.00	6880

7.2. Confusion Matrix

7.2.1. Ingredient category is classified with the 100% accuracy

7.2.2. There are some misclassified tokens which are classified wrongly with Units and Quantity interchangeably

8. Prediction and Model Evaluation

8.1. Classification report

8.1.1. The metrics have somehow dropped a little bit as compared to the model performance on the training data

```
              precision    recall  f1-score   support

  ingredient       1.00      1.00      1.00      2330
    quantity       0.97      0.94      0.96       433
        unit       0.93      0.97      0.95       347

    accuracy                           0.99      3110
   macro avg       0.97      0.97      0.97      3110
weighted avg       0.99      0.99      0.99      3110
```

8.2. Confusion Matrix

8.2.1. Same as the above observation

```
Labels: ['ingredient', 'quantity', 'unit']
[[2330    0    0]
 [   0  407   26]
 [   0   12  335]]
```

9. Error analysis on Validation Data

9.1. Provide Insights from the Validation dataset

9.1.1. All the tokens which are classified wrongly are stored in a dataframe

	sentence_index	token	true_label	predicted_label	prev_token	next_token	class_weight
0	0	rice	quantity	unit	cups	cooked	None
1	0	3	unit	quantity	cooked	tomatoes	None
2	0	dry	quantity	unit	chilli	red	None
3	2	to	quantity	unit	6	8	None
4	4	1/4	unit	quantity	-	cups	None

9.1.2. Overall accuracy of the model on validation dataset came out to be **0.9877**

9.2. Analyse errors by label type

9.2.1. Ingredient

9.2.1.1. There are no misclassified tokens in ingredients category

9.2.2. Units

	sentence_index	token	true_label	predicted_label	prev_token	next_token	class_weight
0	0	rice	quantity	unit	cups	cooked	None
2	0	dry	quantity	unit	chilli	red	None
3	2	to	quantity	unit	6	8	None
6	19	mustard	quantity	unit	teaspoon	asafoetida	None
8	21	cups	quantity	unit	oil	rice	None
9	21	cubed	quantity	unit	onions	carrots	None
10	22	grams	quantity	unit	500	tomatoes	None
11	22	tomatoes	quantity	unit	grams	8	None
13	22	crushed	quantity	unit	corns	sugar	None
16	25	tablespoon	quantity	unit	shallots	ginger	None
17	25	(	quantity	unit	cardamom	elaichi	None
18	29	cup	quantity	unit	1	water	None
19	29	teaspoon	quantity	unit	chillies	oil	None
21	36	inch	quantity	unit	sliced	ginger	None
22	36	red	quantity	unit	masala	chilli	None
23	45	seeds	quantity	unit	cumin	jeera	None

9.2.3. Quantities

	sentence_index	token	true_label	predicted_label	prev_token	next_token	class_weight
1	0	3	unit	quantity	cooked	tomatoes	None
4	4	1/4	unit	quantity	-	cups	None
5	8	stalks	unit	quantity	tomatoes	spring	None
7	21	3	unit	quantity	salt	teaspoons	None
12	22	8	unit	quantity	tomatoes	2	None
14	23	peanuts	unit	quantity	raw	moongphali	None
15	25	4	unit	quantity	rice	-	None
20	29	1/2	unit	quantity	oil	mustard	None
27	47	curry	unit	quantity	sprig	leaves	None
30	60	dhania	unit	quantity	(	)	None
31	60	pizza	unit	quantity	teaspoon	seasoning	None
36	77	mustard	unit	quantity	teaspoon	seeds	None

9.3. Provide Insights from the Validation dataset

9.3.1. Ingredient words are predicted very correctly and there are almost no errors for ingredient.

9.3.2. The model clearly understands ingredient tokens like rice, mustard, tomatoes and onion.

9.3.3. Quantity tokens are often predicted as unit.

9.3.4. Many quantity errors come from words that are not real quantities, such as rice, dry, red, seeds and powder.

9.3.5. These words are wrongly labelled as quantity in the data, which confuses the model.

9.3.6. Unit tokens are sometimes predicted as quantity.

9.3.7. Fraction values like 1/2 and 1/4 look like numbers, so the model thinks they are quantity.

9.3.8. Some ingredient words are wrongly labelled as unit, which causes more confusion.

9.3.9. All errors happen only between quantity and unit.

9.3.10. Ingredient is never confused with quantity or unit.

9.3.11. This shows the model logic is correct and consistent.

9.3.12. Bias features strongly support ingredient and unit labels.

9.3.13. The is_quantity feature helps the model predict quantity.

9.3.14. The is_unit feature helps the model predict unit.

9.3.15. The class_weight feature reduces the dominance of ingredient label.

9.3.16. Context features like previous and next token are very important.

9.3.17. Quantity and unit words appear close to each other in recipes, so confusion is expected.

9.3.18. Most errors come from language ambiguity and noisy labels, not from poor model performance.

9.3.19. Overall, the CRF model works very well and learns useful patterns from the data.