Identifying Key Entities in Recipe Data: Report

Problem Statement

The lack of tagging in recipe data restricts the user experience on online cooling and meal-planning platforms. Recipes require manual annotation of ingredients, quantities, and titles which is time-consuming and inconsistent.

Business Objective

The business objective is to leverage the increasing popularity of online cooking platforms and meal-planning apps by enhancing the user experience. This can be achieved by implementing a **custom-named entity recognition (NER)** model to automatically tag ingredients, quantities and recipe names. This automation will streamline the process of organising recipes, improve searchability and enable users to easily find recipes based on available ingredients, portion sizes or specific dietary requirements. This will ultimately reduce the labour-intensive and inefficient manual tagging process, providing a more accessible and efficient way for businesses in the food and recipe industry to manage their recipe databases.

Dataset Description

This data set comprises culinary recipes with a focus on ingredient extraction and analysis. Each recipe features a structured ingredient list with labelled components, identifying ingredients, quantities and units. This diverse collection supports tasks such as understanding recipes and discovering culinary knowledge, enabling the development of models for information extraction in the culinary domain.

Data Structure:

The given data is in JSON format, representing a structured recipe ingredient list with Named Entity Recognition (NER) labels. Below is a breakdown of the data fields:

[
  {
"input": "6 Karela Bitter Gourd Pavakkai Salt 1 Onion 3 tablespoon Gram flour besan 2 teaspoons Turmeric po
wder Haldi Red Chilli Cumin seeds Jeera Coriander Powder Dhania Amchur Dry Mango Sunflower Oil",
"pos": "quantity ingredient ingredient ingredient ingredient ingredient quantity ingredient quantity unit ingredi
ent ingredient ingredient quantity unit ingredient ingredient ingredient ingredient ingredient ingredient ingredi
ent ingredient ingredient ingredient ingredient ingredient ingredient ingredient ingredient ingredient
 },
 {
 "input": "2-
1/2 cups rice cooked 3 tomatoes teaspoons BC Belle Bhat powder 1 teaspoon chickpea lentils 1/2 cumin seeds
white urad dal mustard green chilli dry red 2 cashew or peanuts 1-1/2 tablespoon oil asafoetida",
"pos": "quantity unit ingredient ingredient quantity ingredient unit ingredient ingredient ingredient ingredient
quantity unit ingredient ingredient quantity ingredient ingredient ingredient ingredient ingredient ingredient in
gredient ingredient ingredient ingredient quantity ingredient ingredient ingredient quantity unit ingredient ingr
edient"
  }
]

Key	Description
`input`	Contains a raw ingredient list from a recipe.
`pos`	Represents the corresponding part-of-speech (POS) tags or NER labels, identifying quantities, ingredients, and units.

Identifying Key Entities in Recipe Data: Report

Methodologies And Techniques Used

I used the following steps for overall analysis:

Techniques and Model used:

- NLP: Includes Lexical Processing, Syntactical Processing like POS Tags, Dependency Parsing, Name Entity Recognition (NER).
- Conditional Random Fields (CRF) Model Machine Learning
- Exploratory Data Analysis (EDA)

Libraries Used

- json for handling JSON data
- pandas for data manipulation and analysis
- re for regular expressions (useful for text preprocessing)
- matplotlib.pyplot for visualisation
- seaborn for advanced data visualisation
- sklearn_crfsuite for CRF (Conditional Random Fields) implementation for sequence modeling
- numpy for numerical computations
- joblib
- random
- spacy
- IPython.display for displaying well-formatted output
- fractions for handling fractional values in numerical data
- collections for counting occurrences of elements in a list
- sklearn.model_selection train_test_split for splitting dataset into train and test sets
- sklearn_crfsuite metrics for evaluating CRF models
- sklearn_crfsuite.metrics flat_classification_report
- sklearn.utils.class_weight import compute_class_weight
- collections import Counter
- sklearn.metrics import confusion_matrix

Data Ingestion and Preparation

- Read the data from JSON file.
- Check the dataset and get shape and information on the dataset

	input	pos
0	6 Karela Bitter Gourd Pavakkai Salt 1 Onion 3 tablespoon Gram flour besan 2 teaspoons Turmeric powder Haldi Red Chilli Cumin seeds Jeera Coriander Powder Dhania Amchur Dry Mango Sunflower Oil	quantity ingredient ingredient ingredient ingredient ingredient quantity ingredient quantity unit ingredient ingredient ingredient quantity unit ingredient ingredient ingredient ingredient ingredient ingredient ingredient ingredient ingredient ingredient ingredient ingredient ingredient ingredient ingredient ingredient
1	2-1/2 cups rice cooked 3 tomatoes teaspoons BC Belle Bhat powder 1 teaspoon chickpea lentils 1/2 cumin seeds white urad dal mustard green chilli dry red 2 cashew or peanuts 1-1/2 tablespoon oil asafoetida	quantity unit ingredient ingredient quantity ingredient unit ingredient ingredient ingredient ingredient quantity unit ingredient ingredient quantity ingredient ingredient ingredient ingredient ingredient ingredient ingredient ingredient ingredient ingredient quantity ingredient ingredient ingredient quantity unit ingredient ingredient
2	1-1/2 cups Rice Vermicelli Noodles Thin 1 Onion sliced 1/2 cup Carrots Gajjar chopped 1/3 Green peas Matar 2 Chillies 1/4 teaspoon Asafoetida hing Mustard seeds White Urad Dal Split Ghee sprig Curry leaves Salt Lemon juice	quantity unit ingredient ingredient ingredient ingredient quantity ingredient ingredient quantity unit ingredient ingredient ingredient quantity ingredient ingredient quantity ingredient quantity unit ingredient ingredient ingredient ingredient ingredient ingredient ingredient ingredient ingredient ingredient ingredient unit ingredient ingredient ingredient ingredient ingredient
3	500 grams Chicken 2 Onion chopped 1 Tomato 4 Green Chillies slit inch Ginger finely 6 cloves Garlic 1/2 teaspoon Turmeric powder Haldi Garam masala tablespoon Sesame Gingelly Oil 1/4 Methi Seeds Fenugreek Coriander Dhania Dry Red Fennel seeds Saunf cups Sorrel Leaves Gongura picked and	quantity unit ingredient quantity ingredient ingredient quantity ingredient quantity ingredient ingredient ingredient unit ingredient ingredient quantity unit ingredient quantity unit ingredient ingredient ingredient ingredient ingredient unit ingredient ingredient ingredient ingredient quantity ingredient ingredient ingredient ingredient ingredient ingredient ingredient ingredient ingredient ingredient ingredient ingredient unit ingredient ingredient ingredient ingredient ingredient
4	1 tablespoon chana dal white urad 2 red chillies coriander seeds 3 inches ginger onion tomato Teaspoon mustard asafoetida sprig curry	quantity unit ingredient ingredient ingredient ingredient quantity ingredient ingredient ingredient ingredient quantity unit ingredient ingredient ingredient unit ingredient ingredient unit ingredient

Identifying Key Entities in Recipe Data: Report

Shape: 285 rows and 2 columns

Information:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 285 entries, 0 to 284
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0   input   285 non-null    object
 1   pos     285 non-null    object
dtypes: object(2)
memory usage: 4.6+ KB
```

Recipe Data Manipulation and Validation:

- Split the input and pos columns to create the tokenized columns for both i.e. input_token, pos_token.

Sample below:

	input	pos	input_token	pos_token
0	6 Karela Bitter Gourd Pavakkai Salt 1 Onion 3 tablespoon Gram flour besan 2 teaspoons Turmeric powder Haldi Red Chilli Cumin seeds Jeera Coriander Powder Dhania Amchur Dry Mango Sunflower Oil	quantity ingredient ingredient ingredient ingredient ingredient quantity ingredient quantity unit ingredient ingredient ingredient quantity unit ingredient ingredient ingredient ingredient ingredient ingredient ingredient ingredient ingredient ingredient ingredient ingredient ingredient ingredient ingredient ingredient	[6, Karela, Bitter, Gourd, Pavakkai, Salt, 1, Onion, 3, tablespoon, Gram, flour, besan, 2, teaspoons, Turmeric, powder, Haldi, Red, Chilli, Cumin, seeds, Jeera, Coriander, Powder, Dhania, Amchur, Dry, Mango, Sunflower, Oil]	[quantity, ingredient, ingredient, ingredient, ingredient, ingredient, quantity, ingredient, quantity, unit, ingredient, ingredient, ingredient, quantity, unit, ingredient, ingredient, ingredient, ingredient, ingredient, ingredient, ingredient, ingredient, ingredient, ingredient, ingredient, ingredient, ingredient, ingredient, ingredient, ingredient]

- Checked the length of the input_token and pos_tokens to make sure the lengths are equal. From the equality check, we have the following 5 rows with inequal length: [17,27,79,164,207]
- Checked the unique values in the pos column. The results were {'ingredient', 'quantity', 'unit'}
- Removed the rows with inequal length of input and pos tokens. Post removal, checked the shape of the dataset which was: (280, 6).
- Updated the input_length and pos_length columns to reflect the updated length i.e. 280, 6
- Checked the lengths again to find out if there is any more records of inequal length. Count of rows which are not of equal length :0

Training and Validation data split

- The dataset is split into training and validation sets using a 70:30 ratio using train_test_split with a random_state of 42.
- The output data frames are train_df, val_df. Checked records for both data frames
- Extracted X_train, X_val, y_train and y_val by extracting the list of input_tokens and pos_tokens from train_df and val_df and checked their length.
  - X_train.shape : (196,)
  - X_val.shape : (84,)
  - y_train.shape : (196,)
  - y_val.shape : (84,)
- Checked for unique labels in train_df which was {'ingredient', 'quantity', 'unit'}

Exploratory Data Analysis on Training Dataset

- Extracted the input tokens and its pos tags in training dataset and flattened it.
- Checked the length of input and pos tokens which were equal.
- Categorised tokens into ingredients, units and quantities by using extracted token function get a list of ingredients, units and quantities.

Identifying Key Entities in Recipe Data: Report

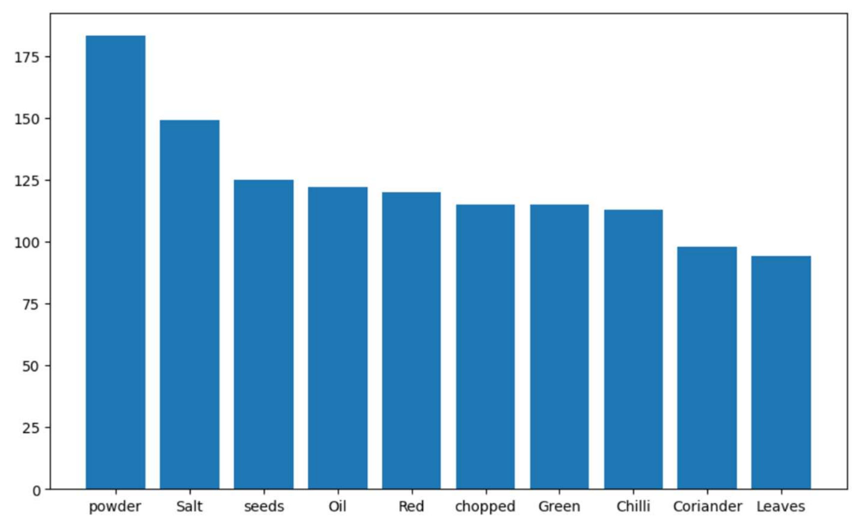
- Created and used a function to get the top ingredients.

```
[('powder', 183),
 ('Salt', 149),
 ('seeds', 125),
 ('Oil', 122),
 ('Red', 120),
 ('chopped', 115),
 ('Green', 115),
 ('Chilli', 113),
 ('Coriander', 98),
 ('Leaves', 94)]
```

- Created and used a function to get the top recipes used in recipes.

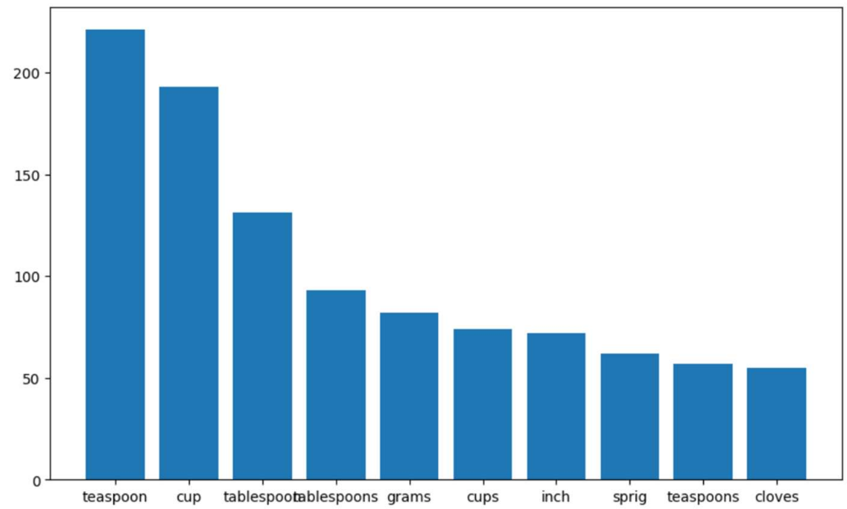
```
[('teaspoon', 221),
 ('cup', 193),
 ('tablespoon', 131),
 ('tablespoons', 93),
 ('grams', 82),
 ('cups', 74),
 ('inch', 72),
 ('sprig', 62),
 ('teaspoons', 57),
 ('cloves', 55)]
```

- Plotted the top 10 frequently used ingredients from training dataset:



- Plotted the top 10 frequently used ingredients from training dataset:

Identifying Key Entities in Recipe Data: Report



Feature Extraction for CRF Model

- Defined keywords for unit and quantity and created a quantity pattern to work on fractions, numbers and decimals.
    - quantity_pattern = re.compile(r'^\d+([/-]\d+)?(\.\d+)?$|^\d+/\d+$|^\d+\.\d+$|^\d+$')
- Loaded the spicy model -> spacy.load("en_core_web_sm")
- Defined a feature function for processing each token in the sentence.
- Converted the X_train, X_val, y_train and y_val into train and validation feature sets and labels
- Checked the length of training and validation features and labels, Output -> 196 and 84 respectively.
- Created ***label_counts*** to count the frequencies of labels present in training dataset y_train_flat (flatteded y_train) and retrieved the total samples.
    - Label count -> Counter ({'ingredient': 5323, 'quantity': 980, 'unit': 811})
    - Total samples -> 7114
- Compute class weights (inverse frequency method) by considering total_samples and label_counts

```
{'quantity': 2.419727891156463,
 'unit': 2.923962186600904,
 'ingredient': 0.44548813325818776}
```

- Applied penalizing factor on the ingredient label.
- Created ***X_train_weighted_features*** and ***X_val_weighted_features*** for extracting training and validation features along with their weights by using a method created for this purpose.

Model Building and Training

- A CRF model was implemented using the sklearn_crfsuite library.
- CRFs are ideal for sequence labelling because they account for contextual dependencies between adjacent labels.

```
CRF
CRF(algorithm='lbfgs', all_possible_transitions=True, c1=0.5, c2=1.0,
    max_iterations=100)
```

Model Evaluation using CRF

- Evaluate on training dataset using
    - CRF by using flat classification report and
    - confusion matrix
- Classification reports were generated using training data set and training data

Identifying Key Entities in Recipe Data: Report

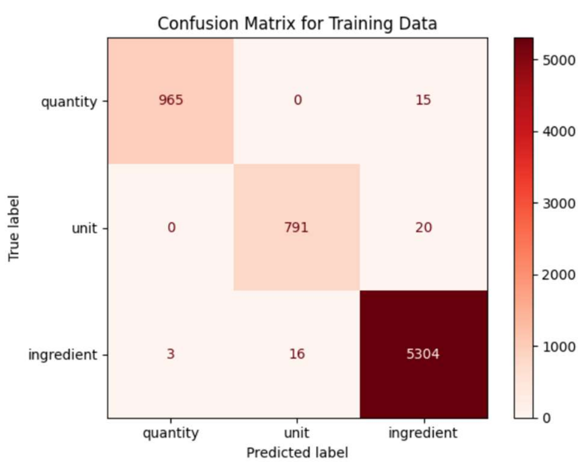
Classification Report on Training Set:

	precision	recall	f1-score	support
ingredient	0.99	1.00	0.99	5323
quantity	1.00	0.98	0.99	980
unit	0.98	0.98	0.98	811
accuracy			0.99	7114
macro avg	0.99	0.99	0.99	7114
weighted avg	0.99	0.99	0.99	7114

Flat Classification Report on Training Data:

	precision	recall	f1-score	support
quantity	0.9969	0.9847	0.9908	980
unit	0.9802	0.9753	0.9778	811
ingredient	0.9934	0.9964	0.9949	5323
accuracy			0.9924	7114
macro avg	0.9902	0.9855	0.9878	7114
weighted avg	0.9924	0.9924	0.9924	7114

- Generated confusion matrix for training for the training data



- Model was saved as crf_model.pkl

Prediction and Model Evaluation

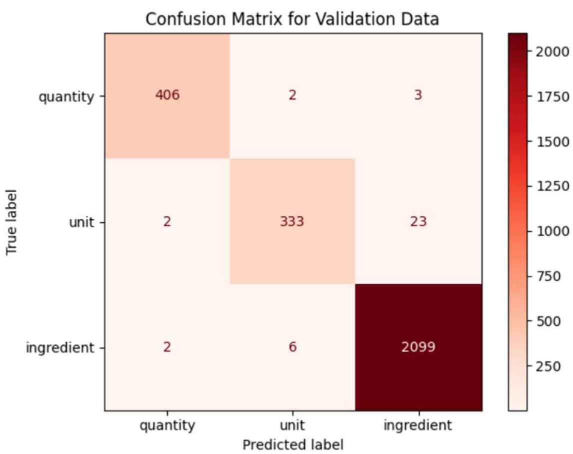
- Similar to training data, used CRF model to generate classification report on validation data.

Flat Classification Report on Validation Data:

	precision	recall	f1-score	support
quantity	0.9902	0.9878	0.9890	411
unit	0.9765	0.9302	0.9528	358
ingredient	0.9878	0.9962	0.9920	2107
accuracy			0.9868	2876
macro avg	0.9848	0.9714	0.9779	2876
weighted avg	0.9867	0.9868	0.9867	2876

- Generated confusion matrix using validation data

Identifying Key Entities in Recipe Data: Report



Error analysis on validation data

- Performed error analysis on validation data to find out mis-classified samples.

  Total validation samples: 2876
  Total errors found: 38
  First 5 errors: [('ingredient', 'unit'), ('unit', 'ingredient'), ('unit', 'ingredient'), ('quantity', 'ingredient'), ('quantity', 'ingredient')]

- Iterated through validation data (X_val, y_val_labels, y_pred_val) and compared true vs. predicted labels. Collected error details, including surrounding context, previous/next tokens, and class weights.

  Total errors collected: 38
  Sample error details:
  {'token': 'cloves', 'true_label': 'ingredient', 'predicted_label': 'unit', 'prev_token': '3', 'next_token': 'garlic', 'class_weight': 0.04454881332581878}
  {'token': 'Spoon', 'true_label': 'unit', 'predicted_label': 'ingredient', 'prev_token': 'big', 'next_token': 'oil', 'class_weight': 2.923962186600904}
  {'token': 'cloves', 'true_label': 'unit', 'predicted_label': 'ingredient', 'prev_token': 'seeds', 'next_token': 'garlic', 'class_weight': 2.923962186600904}
  {'token': 'is', 'true_label': 'quantity', 'predicted_label': 'ingredient', 'prev_token': 'Pur', 'next_token': '2', 'class_weight': 2.419727891156463}
  {'token': 'to', 'true_label': 'quantity', 'predicted_label': 'ingredient', 'prev_token': 'sugar', 'next_token': 'tablespoons', 'class_weight': 2.419727891156463}

- Changed error_data into dataframe and then used it to illustrate the overall accuracy of validation data

```
Validation Error DataFrame:
    token  true_label predicted_label prev_token   next_token  class_weight
0  cloves  ingredient            unit          3       garlic      0.044549
1   Spoon        unit      ingredient        big          oil      2.923962
2  cloves        unit      ingredient      seeds       garlic      2.923962
3      is    quantity      ingredient        Pur            2      2.419728
4      to    quantity      ingredient      sugar  tablespoons      2.419728

 Overall Accuracy on Validation Data: 0.9868
```

- Analysed errors found in the validation data by each label and displayed their class weights along with accuracy and also display the error dataframe with token, previous token, next token, true label, predicted label and context

```
Error Analysis by Label:

Label       Errors    Total     Accuracy    Class Weight
-----------------------------------------------------
quantity    5         411       0.9878      2.4197
unit        25        358       0.9302      2.9240
ingredient  8         2107      0.9962      0.0445
```

Insights from validation dataset

- From the classification report we can draw the below insight:
  - Model performance is excellent with an overall accuracy of **98.68%** and **macro F1-score of 97.79%** indicates that your model is highly reliable across all three entity types
  - Ingredient entity type performs the best with **Precision: 0.9878**, **Recall: 0.9962**, **F1: 0.9920.** The model is quite confident and accurate in recognizing the **ingredients**.
  - **Quantity** entity also performs very well with F1-score of **0.9890.**
  - **Unit** has low score, still very strong performance, but relatively lower **recall** (0.9302). There are chances o mis-classifying the units compared to other entities.
- From the confusion matrix we have the following insights
  - Model accuracy is very high. The **diagonal values (406, 333, 2099)** represent correct predictions.
  - Almost all tokens are classified correctly across all three classes.
  - Similar to classification report, here also **ingredients** perform best where out of 2107 (2+6+2099) only 9 are mis-classified.
  - **Quantity** class is very clean where out of 411, only **5 misclassifications** (2 as unit, 3 as ingredient).
  - **Unit** has major mis-classification where **23 unit tokens** misclassified as ingredients (most significant confusion).

Recommendations

No major recommendations, model looks very good. Units can be looked into a bit for further accuracy.

Conclusion

- The given dataset was overall ok but few data quality issues were identified with 5 records where the input token and corresponding pos tag lengths were not matching.
- The primary entities extracted were 'quantity', 'ingredient', and 'unit'.
- Overall, a very good model with more than 98.68% accuracy and weighted F1-score 98.67%.
- Model is not overfitted as it works very well on unseen data.
- Due to its high-accuracy, The CRF model can reliably replace **manual tagging** of recipe data, saving time and reducing error.