Identifying Key Entities in Recipe Data

Submitted by:
Prashant Saxena
Ph. 9818059522
Prashantonly2024@gmail.com

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 organizing 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.

Problem Statement

The goal of this assignment is to identify and classify key entities in cooking recipe data using Named Entity Recognition (NER). Specifically, we aim to extract and correctly label entities such as 'ingredient', 'quantity', and 'unit' from textual recipe instructions. This task is vital for structuring and digitizing recipe data to make it searchable and usable in digital applications.

Assumptions Made

During the development of the Named Entity Recognition (NER) pipeline for identifying key entities in recipe data, several assumptions were made to simplify the modeling process and focus the scope of the problem. These assumptions influenced data preprocessing, feature extraction, and model design:

- **Entity Classes Are Mutually Exclusive**
  Each token is assumed to belong to only one of the three entity classes: 'ingredient', 'quantity', or 'unit'. No overlapping or nested entities are considered.
- **Labels Are Assigned at the Token Level**
  The model processes and assigns labels at the token level. Multi-word entities are expected to be captured by a sequence of token-level predictions.
- **Context Is Limited to Neighboring Tokens**
  Feature engineering includes context from a fixed number of preceding and following tokens, without considering the entire sentence or paragraph.
- **Class Weights Are Used to Address Imbalance**
  To address imbalance in the dataset (e.g., more 'ingredient' tokens), class weights are computed and used in the training process.
- **No External Knowledge Base Is Used**
  The model relies solely on patterns learned from the training data, without referencing external databases or lexicons for ingredients or units.
- **Evaluation Ignores Sentence-Level Entity Boundaries**
  Evaluation is based on token-level accuracy and confusion matrices. Span-level or partial entity recognition is not accounted for.
- **Noise and Misspellings Are Minimal**
  The code assumes a clean dataset without significant typos, OCR errors, or noisy data entries.

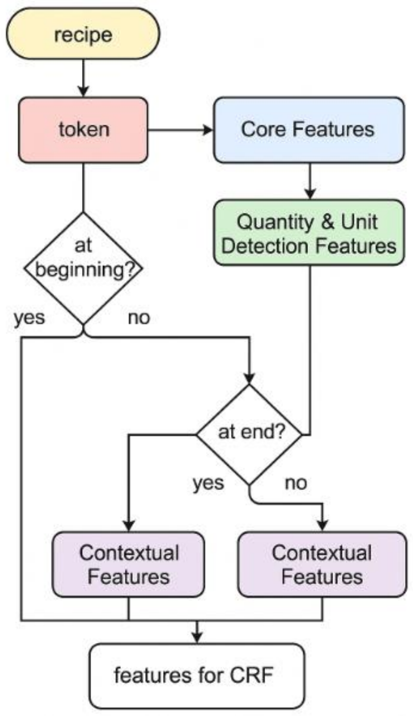
Methodology

The approach involves preprocessing recipe texts, tokenizing them, and then applying NER to classify each token. We used traditional sequence labeling techniques along with feature engineering to build our models. The key stages in our methodology include:

- **Data Ingestion and Preparation**
  - Dataset is provided in Jason format which is ingested and prepared before further analysis.
  - Tokenization - Input and pos are split into tokens and then Derived Metrics such as input_length and pos_length are used to validate the data.
  - Further rows are dropped where input length doesn’t match pos_length.
  - Dataset is now split into training and validation dataset in 70:30 ratio.
  - Nested lists of tokens are flattened
  - These steps enhance the model's ability to learn meaningful patterns from the data.

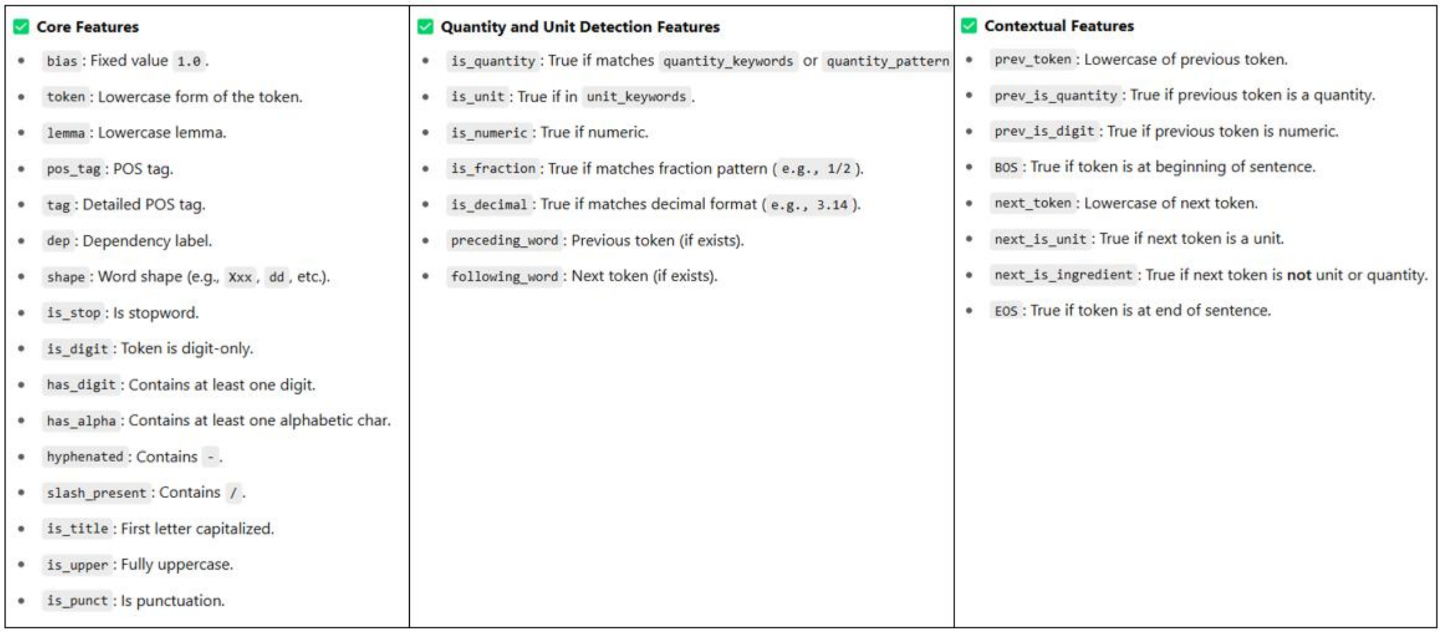
- **Exploratory Data Analysis**
  - EDA is performed on both the training and validation dataset
  - Top 10 most frequent ingredients and units are identified
  - Frequencies of occurrence most common ingredients and units are plotted to develop a sense of how data looks like

- **<u>Feature Extraction using linguistic and contextual cues</u>**



spacy-based features per token:

- POS tags
- Lemma
- Dependency
- Capitalization, punctuation, etc



✅ **Core Features**	✅ **Quantity and Unit Detection Features**	✅ **Contextual Features**
• `bias` : Fixed value `1.0` .	• `is_quantity` : True if matches `quantity_keywords` or `quantity_pattern`	• `prev_token` : Lowercase of previous token.
• `token` : Lowercase form of the token.	• `is_unit` : True if in `unit_keywords` .	• `prev_is_quantity` : True if previous token is a quantity.
• `lemma` : Lowercase lemma.	• `is_numeric` : True if numeric.	• `prev_is_digit` : True if previous token is numeric.
• `pos_tag` : POS tag.	• `is_fraction` : True if matches fraction pattern (`e.g., 1/2`).	• `BOS` : True if token is at beginning of sentence.
• `tag` : Detailed POS tag.	• `is_decimal` : True if matches decimal format (`e.g., 3.14`).	• `next_token` : Lowercase of next token.
• `dep` : Dependency label.	• `preceding_word` : Previous token (if exists).	• `next_is_unit` : True if next token is a unit.
• `shape` : Word shape (e.g., `Xxx` , `dd` , etc.).	• `following_word` : Next token (if exists).	• `next_is_ingredient` : True if next token is **not** unit or quantity.
• `is_stop` : Is stopword.		• `EOS` : True if token is at end of sentence.
• `is_digit` : Token is digit-only.		
• `has_digit` : Contains at least one digit.		
• `has_alpha` : Contains at least one alphabetic char.		
• `hyphenated` : Contains `-` .		
• `slash_present` : Contains `/` .		
• `is_title` : First letter capitalized.		
• `is_upper` : Fully uppercase.		
• `is_punct` : Is punctuation.		

- <u>**Model Selection and Training**</u>
    - Model used: **Conditional Random Fields (CRF)** via sklearn_crfsuite
        - It is a probabilistic model that considers the context of neighboring labels.
        - CRF captures sequential token dependencies.
    - Model is trained on 70% of dataset
    - **Handling Class Imbalance**
        - Compute **class weights** using sklearn.utils.class_weight
        - Ingredient label is penalized by applying a multiplier to handle class imbalance

- <u>**CRF Model Hyperparameters**</u>

Parameter	Description
algorithm='lbfgs'	Optimisation algorithm used for training. `lbfgs` (Limited-memory Broyden–Fletcher–Goldfarb–Shanno) is a quasi-Newton optimisation method.
c1=0.5	L1 regularisation term to control sparsity in feature weights. Helps in feature selection.
c2=1.0	L2 regularisation term to prevent overfitting by penalising large weights.
max_iterations=100	Maximum number of iterations for model training. Higher values allow more convergence but increase computation time.
all_possible_transitions=True	Ensures that all possible state transitions are considered in training, making the model more robust.

- <u>**Model Evaluation**</u>
    - **Classification Report** - a summary of metrics used to evaluate the performance of a classification model. It provides a detailed analysis of the model's performance, including precision, recall, F1-score, and support for each class. These metrics offer insights into how well the model predicts different classes and helps identify areas for improvement.

    - **Confusion Matrix** - a table used to evaluate the performance of a classification model. It compares the model's predictions against the actual outcomes, providing a visual representation of how the model performs on different classes.

    - **Model Accuracy** - the percentage of correct predictions a model makes, indicating how often it gets things right. It's a fundamental metric for evaluating a model's performance in classification tasks, providing a high-level overview of how well it's trained to identify patterns and make accurate predictions.

    - **Error Analysis** – to improve model performance
      A detailed examination of model errors provides insights into its limitations:
        - <u>Mislabeling Patterns</u>: Identifying frequent confusions between entity types
        - <u>Contextual Challenges</u>: Analyzing errors from ambiguous contexts
        - <u>Token-Level Analysis</u>: Reviewing tokens with high error rates.

      This informs model refinement and data augmentation strategies.

Techniques Used

Several techniques were utilized in this assignment:

- **Spacy for tokenization and linguistic features**
    - Spacy, a powerful NLP library, was used to process the recipe text.
    - It provided tokenization, breaking the sentence into individual tokens (e.g., words or punctuation).
    - Additionally, Spacy supplied linguistic features for each token:
    - Lemma: The base form of the word.
    - POS tags: Grammatical categories like noun, verb, etc.
    - Dependency parsing: Syntactic relationships between words.
    - Shape, is_stop, is_alpha, is_digit, etc. to enrich the feature set.

- **Custom feature engineering for contextual understanding**
    - Beyond basic token properties, custom features were designed to add semantic and contextual insight.
    - These features captured patterns relevant to ingredient quantities and units in recipes:
        - Regex-based detection of fractions and decimals.
        - Sets of keywords to identify measurement units and quantity terms.
        - Contextual cues from neighboring tokens, like whether the next word is a unit or the previous word is numeric.
    - Also included indicators for start (BOS) and end (EOS) of the sentence for sequence modeling.

- **CRF (Conditional Random Fields) for sequence labeling**
    - CRF is a probabilistic model used to label sequences, ideal for structured prediction tasks like tagging ingredients.
    - It takes the engineered features and learns dependencies between labels in a sequence (e.g., B-QUANTITY, I-UNIT, B-INGREDIENT).
    - CRF is particularly suited for this because it:
        - Considers contextual dependencies.
        - Can enforce constraints like proper label transitions.
    - The model was trained on labeled recipe data and used to predict labels for new text.

- **Sklearn for computing evaluation metrics and confusion matrix**
  Scikit-learn (sklearn) was used for model **evaluation**:
    - Calculated accuracy, precision, recall, and F1-score.
    - Helped quantify how well the CRF model performed on test data.
    - A confusion matrix was generated to visually assess which labels were misclassified.

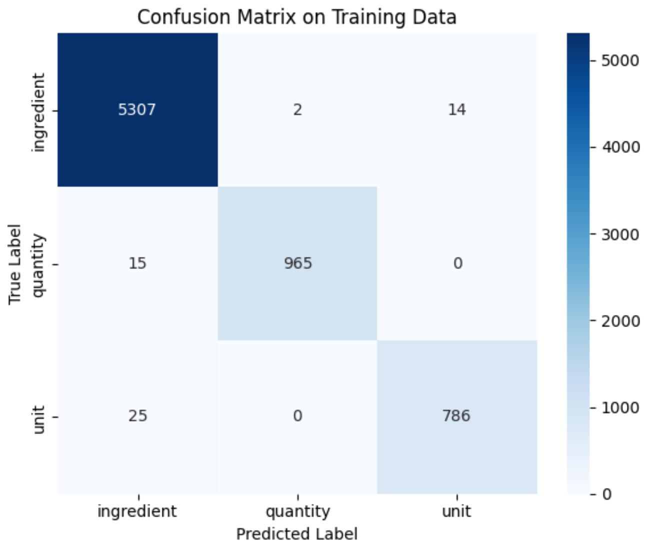
Model Evaluation

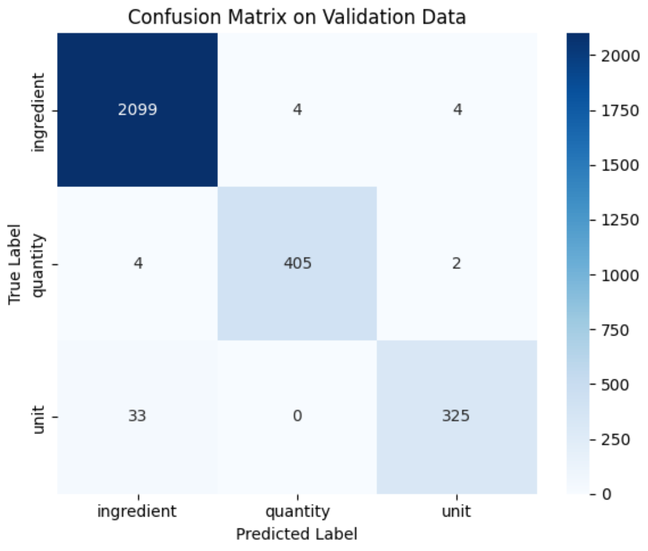
Classification Report

```
Flat Classification Report on Training Data:
              precision    recall  f1-score   support

  ingredient       0.99      1.00      0.99      5323
    quantity       1.00      0.98      0.99       980
        unit       0.98      0.97      0.98       811

    accuracy                           0.99      7114
   macro avg       0.99      0.98      0.99      7114
weighted avg       0.99      0.99      0.99      7114
```

```
Flat Classification Report on Validation Data:
              precision    recall  f1-score   support

  ingredient       0.98      1.00      0.99      2107
    quantity       0.99      0.99      0.99       411
        unit       0.98      0.91      0.94       358

    accuracy                           0.98      2876
   macro avg       0.98      0.96      0.97      2876
weighted avg       0.98      0.98      0.98      2876
```

Confusion Matrix





Model Accuracy

Training Accuracy – **99.2%**

Validation Accuracy – **98.4%**

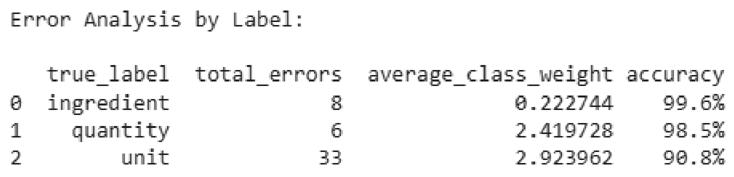
Evaluation Summary and Insights

- The CRF model performs reasonably well in identifying ingredients, units, and quantities, with certain confusions between closely related classes (e.g., ingredient vs. unit).
- 99.2% Training accuracy and 98.4% Validation accuracy suggests minimal overfitting, and hence a robust model
- Individual accuracies of true ingredient, quantity and unit labels are all above 90%
- The model could predict the labels correctly for the validation data with an accuracy rate of 98.4%
- Out of 2876 flattened labels, only 47 were incorrectly predicted, error percentage being only 1.6%
- Most incorrect predictions are made in units

Error Analysis Summary

To better understand the performance of the NER model, an error analysis was conducted on the validation data. This analysis focused on tokens where the model's predictions did not match the true labels. The errors were grouped by the true label to compute the total number of errors, the average class weight, and the accuracy per label.

This helped identify that 'unit' and 'ingredient' were often confused, especially in cases involving short context windows or ambiguous expressions like 'few', 'small', 'medium'. These findings suggest the need for better context understanding or data augmentation.



```
Error Analysis by Label:

   true_label  total_errors  average_class_weight accuracy
0  ingredient             8              0.222744    99.6%
1    quantity             6              2.419728    98.5%
2        unit            33              2.923962    90.8%
```

From the table above, it's evident that the 'unit' label has the highest total error count. This suggests it may be beneficial to refine the model or apply post-processing rules to improve accuracy.

Key Insights

Key observations from data sanity checks:

- There are 5 records having indices - [17, 27, 79, 164, 207], which have different input and pos length
- For 4 indices - [27, 79, 164, 207], the mismatch occurs due to compound numbers (e.g. 1 1/2). The quantity token has a space between digits, which upon split results in two tokens instead of one, and hence length mismatch
- For index 17, there is an extra word 'cor' which does not have a corresponding pos token.

Through this assignment, we observed that:

- The CRF model performs reasonably well in identifying ingredients, units, and quantities, with certain confusions between closely related classes (e.g., ingredient vs. unit).
- Balanced class weights improve accuracy for underrepresented labels.
- Visualizing prediction mismatches using confusion matrices helps pinpoint problematic labels.
- Most incorrect predictions are made in units
  - Many of these incorrect predictions are on connecting words like a, to, into, etc.
- Certain tokens such as 'cloves' and 'few' are often misclassified due to their ambiguity.
- "few", "pinch", "some" are context-sensitive
- Entity boundaries can be ambiguous (e.g., "2 tbsp sugar")
- Most prominent source of incorrect predictions is the ingredients which are measured in number, such as clove(s)
  - This is because the model learned from the common pattern where a numeric value is usually followed by a unit, e.g. 1 tsp, 200 gram, etc. So, the model incorrectly predicts the ingredient in phrase "4 cloves" for instance.
- It is also important to note that the in quite a few cases, pos labels in dataset looks intuitively incorrect, which leads to sub-optimal learning.
  - For instance, 'green' is labelled as unit and 'few' is labelled as 'ingredients' in provided dataset.

Scope of Improvement

The model demonstrated good accuracy overall, especially for 'ingredient' and 'quantity' labels. However, confusion between semantically similar classes like 'unit' and 'quantity' highlights a limitation in the model's context understanding.

To improve performance, the following steps are recommended:

- Use contextual embeddings such as BERT or spacy transformers.
- Increase the dataset size through augmentation.
- Incorporate domain-specific rules or gazetteers.
- Analyze token-level errors to design targeted heuristics.

Conclusion

- This project successfully tackled the task of Named Entity Recognition on recipe data, identifying key entities like ingredients, units, and quantities.
- Using CRF and spacy-based pipelines, we achieved high accuracy, especially for common and well-contextualized terms.
- Further improvements can be made through contextual embeddings and rule-based refinements informed by error analysis.
- The structured approach and error diagnostics demonstrate the effectiveness of combining statistical and rule-based NLP methods.
- The CRF model performs reasonably well in identifying ingredients, units, and quantities, with certain confusions between closely related classes (e.g., ingredient vs. unit).
- Visualizations such as confusion matrices provide insights into the types of errors made, and error analysis highlights the need for additional contextual features.