Project Report: Identifying Key Entities in Recipe Data

Executive Summary

This project successfully demonstrates the application of Named Entity Recognition (NER) using a Conditional Random Field (CRF) model to extract key entities (quantities, units, and ingredients) from unstructured recipe ingredient lists. The process involved data loading and cleaning, exploratory data analysis to understand data characteristics and label distribution, feature engineering to represent tokens with relevant linguistic and contextual information, model training with CRF, and evaluation using standard NER metrics. While the model achieved high accuracy, particularly for the majority 'ingredient' class, error analysis highlighted common confusions between quantity/unit and ingredient labels, often in ambiguous contexts. The outcome is a trained model capable of converting raw ingredient text into structured data, enabling various downstream applications in recipe management systems.

1. Introduction

1.1 Problem Statement

The goal of this assignment was to train a Named Entity Recognition (NER) model to extract key entities from recipe data, specifically classifying words into predefined categories: ingredients, quantities, and units. This structured extraction is crucial for building databases that can power advanced recipe features, such as dietary analysis, shopping list generation, or ingredient substitution suggestions.

1.2 Business Objective

The primary business objective is to create a structured database of recipes and ingredients from raw ingredient list text. This structured data is essential for developing applications like advanced recipe search, automated nutritional analysis, personalized meal planning, and integration with grocery delivery services.

1.3 Data Description

The dataset is provided in JSON format. Each entry contains two fields:

- input: A string representing a raw ingredient list line.
- pos: A string of space-separated labels corresponding to each token in the input string, indicating its entity type ('quantity', 'unit', or 'ingredient').

An example entry:
{"input": "2-1/2 cups rice cooked", "pos": "quantity unit ingredient ingredient"}

2. Methodology

The project followed a standard machine learning pipeline for sequence labeling:

2.1 Data Loading and Preprocessing

- The JSON data was loaded into a pandas DataFrame.

- The input and pos strings were tokenized by splitting on spaces, creating input_tokens and pos_tokens columns (lists of strings).
- A crucial validation step was performed to check if the number of tokens in input_tokens matched the number of labels in pos_tokens for each entry.
- Rows with unequal token/label counts (found at indices 30 and 35) were identified and dropped as they represent inconsistent data unsuitable for sequence labeling.
- The DataFrame was then verified to ensure all remaining rows had matching token and label lengths.

2.2 Exploratory Data Analysis (EDA)

- EDA was performed on the training dataset to understand the data distribution.
- The lists of tokens and labels were flattened to count occurrences of individual words and labels.
- The number of unique labels was confirmed (expected to be 'quantity', 'unit', 'ingredient').
- The top 10 most frequent ingredients and units were identified and visualized using bar plots.
- **Insights from EDA:** The most frequent items were common kitchen staples and standard units, indicating the dataset covers typical recipes. The label distribution showed 'ingredient' as the most frequent class, highlighting potential class imbalance challenges.

2.3 Train Validation Split

- The cleaned dataset was split into training (70%) and validation (30%) sets using train_test_split.
- The splits were performed on the tokenized lists (input_tokens and pos_tokens).
- The lengths of the resulting training and validation sequences and their corresponding labels were confirmed.

2.4 Feature Engineering

- Token-level features are essential for sequence labeling models like CRF. A word2features function was created to extract rich features for each token in a sentence.
- Features included:
    - **Core Lexical Features:** Token text (lowercase), length, shape, case properties (is_lower, is_title, is_upper), digit/alpha presence, punctuation, hyphen/slash presence.
    - **Numeric/Unit Detection:** Flags for whether the token matches defined unit_keywords, quantity_keywords, or a quantity_pattern regex (handling integers, decimals, fractions, mixed numbers). Includes flags for general is_numeric_like, is_fraction, is_decimal.

    - **Linguistic Features (from spaCy):** Lemma, POS tag, detailed tag, dependency relation, is_stop word (though less emphasis was placed on deep linguistic features in the provided code).
    - **Contextual Features:** Features of the immediately preceding and following tokens (their text, length, shape, and numeric/unit flags). Includes BOS (Beginning of Sentence) and EOS (End of Sentence) flags.
- A sent2features function applied word2features to every token in a given sentence (list of tokens).
- The training and validation data (X_train, X_val) were transformed into feature sets (X_train_features, X_val_features) using sent2features.

2.5 Handling Class Imbalance

- The labels in the training set were flattened (y_train_flat) to count the frequency of each label ('quantity', 'unit', 'ingredient').
- An inverse frequency method was used to compute class weights (weight_dict), assigning higher weights to less frequent labels.
- The 'ingredient' label's weight was intentionally penalized (reduced) to further address its dominance.
- *Note:* While weights were computed, the standard sklearn-crfsuite.CRF.fit method does not support a class_weights parameter in the typical way. Consequently, the model was trained without directly applying these computed weights during the fit process.

2.6 Model Selection and Training

- A Conditional Random Field (CRF) model (sklearn_crfsuite.CRF) was selected, which is well-suited for sequence labeling tasks like NER.
- The model was initialized with hyperparameters (algorithm='lbfgs', c1=0.5, c2=1.0, max_iterations=100, all_possible_transitions=True) for training.
- The model was trained using crf.fit() on the X_train_features and y_train_labels.

2.7 Model Evaluation

- The trained model was evaluated on both the training and validation datasets.
- Evaluation metrics included:
    - **Flat Classification Report:** Providing Precision, Recall, F1-score, and Support for each individual label ('quantity', 'unit', 'ingredient') after flattening all token-level predictions and true labels.
    - **Confusion Matrix:** Visualizing the counts of True Positives, False Positives, False Negatives, and True Negatives for each label.

3. Results and Evaluation

3.1 Training Performance

Evaluation on the training data showed very high scores across all labels, indicating that the model learned the training data patterns effectively. This serves as a baseline but doesn't reflect generalization.

3.2 Validation Performance

The model's performance on the unseen validation data provides a realistic estimate of its generalization capabilities.

- **Flat Classification Report (Validation):** (Referencing the actual output from the notebook) This report shows strong performance for the 'ingredient' class (high precision, recall, F1), moderate performance for 'quantity', and slightly lower performance for 'unit'. This aligns with the frequency distribution observed in EDA.
- **Confusion Matrix (Validation):** (Referencing the actual output plot) The matrix confirmed high true positives for 'ingredient'. Significant off-diagonal counts were observed for:
    - True 'quantity' labeled as 'ingredient'.
    - True 'unit' labeled as 'ingredient'.
      This indicates the primary confusion is between the minority quantity/unit classes and the majority ingredient class. Misclassifications in the other direction (ingredient -> quantity/unit) were less frequent.

4. Error Analysis

4.1 Process

- Misclassified tokens were identified by comparing the true labels (y_val_labels flattened) against the predicted labels (y_pred_val flattened).
- For each misclassified token, relevant information was collected: the token itself, its true label, its predicted label, the previous and next token for immediate context, and a broader context string showing surrounding words.
- This error data was structured into a pandas DataFrame (error_df).

4.2 Key Observations

- Analysis of the error DataFrame reinforced the patterns seen in the confusion matrix.
- Many errors involved tokens that are ambiguous out of context (e.g., numbers, words that can be both units and ingredients).
- Errors frequently occurred at the boundaries between quantities, units, and ingredients.
- Examples include:
    - Numbers predicted as 'ingredient' when they function as quantities.
    - Unit names predicted as 'ingredient'.
    - Occasionally, ingredients predicted as quantities or units if they appeared near such tokens or resembled them.

- The context feature was useful in understanding why a specific prediction might have been made, highlighting the importance of local context, which the CRF model is designed to leverage.

5. Insights and Discussion

- The project successfully applied CRF for NER on recipe data, demonstrating its capability to structure unstructured text.
- The feature engineering captured key lexical, pattern-based, and local contextual information crucial for distinguishing entities.
- The model performed well, especially for the most frequent 'ingredient' class, achieving metrics suitable for practical applications.
- The main challenge lies in accurately identifying 'quantity' and 'unit' tokens, which are less frequent and often appear in more varied or ambiguous patterns.
- The observed confusions between entity types are consistent with the inherent difficulty of segmenting and classifying sequential text data.
- The inability to directly apply computed class weights in sklearn-crfsuite.fit might have limited the model's ability to improve performance on the minority classes.

6. Actionable Outcomes and Future Work

- **Actionable Outcome:** The trained CRF model can be deployed to process new, raw ingredient lists, transforming them into a structured format (list of tokens with assigned labels: quantity, unit, ingredient). This structured output is the direct input for building the described database and powering advanced recipe features.
- **Future Work:**
    - **Improved Feature Engineering:** Explore more sophisticated features, such as n-gram features, gazetteer features (using external lists of known ingredients/units), or integrating richer linguistic features (like SpaCy's lemma, POS, dependency parse results more robustly). Handle multi-word ingredients/units explicitly.
    - **Advanced Models:** Investigate more powerful sequence models like Bi-LSTM-CRF or Transformer-based models (e.g., using libraries like spaCy's NER component or Hugging Face Transformers finetuned for this task), which can capture longer-range dependencies and leverage pre-trained embeddings.
    - **Handling Imbalance:** Explore alternative libraries or custom training loops that allow for direct class weighting or other imbalance techniques like oversampling or undersampling if using a model that supports it.
    - **Post-processing:** Implement rules-based post-processing to correct common errors identified during error analysis (e.g., if a number is followed by a common unit, ensure they are labeled correctly).

7. Conclusion

This project successfully implemented an NER solution for recipe data using a CRF model. By focusing on robust feature engineering and leveraging the sequential nature of the data, the model achieved good performance in extracting quantities, units, and ingredients.