



Syntactic Processing – Deep Learning Assignment

Identifying Key Entities in Recipe Data

Author: Tanvir Ahmed

Batch: AI/ML

Purpose: Deep learning Assignment

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.

Introduction

1. Background

A recipe database with POS industry specific POS tagging is provided to build a model which should be able to find 3 key tags found in any recipe as given below:

1. Ingredient
2. Quantity
3. Unit for Quantity

2. Objectives

1. Import Necessary Libraries
2. Data Ingestion and Preparation
3. Train-Validation Split
4. Exploratory Data Analysis on the Training and validation Data Set
5. Feature Extraction for the CRF Model
6. Model Building and Training
7. Prediction and Model Evaluation
8. Error Analysis on the Validation Data
9. Conclusion

3. Problem

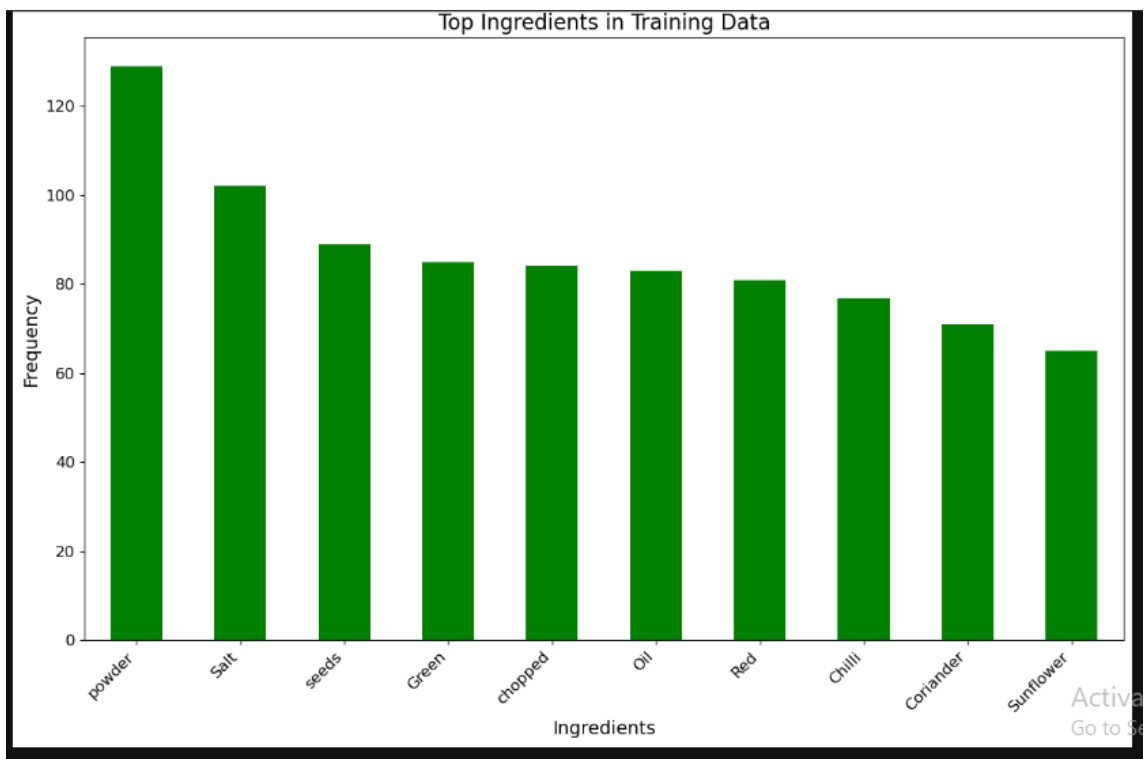
Such model could be used to solve following industry specific problems:

1. Providing filtering for restaurant review to highlight user targeted product review at top.

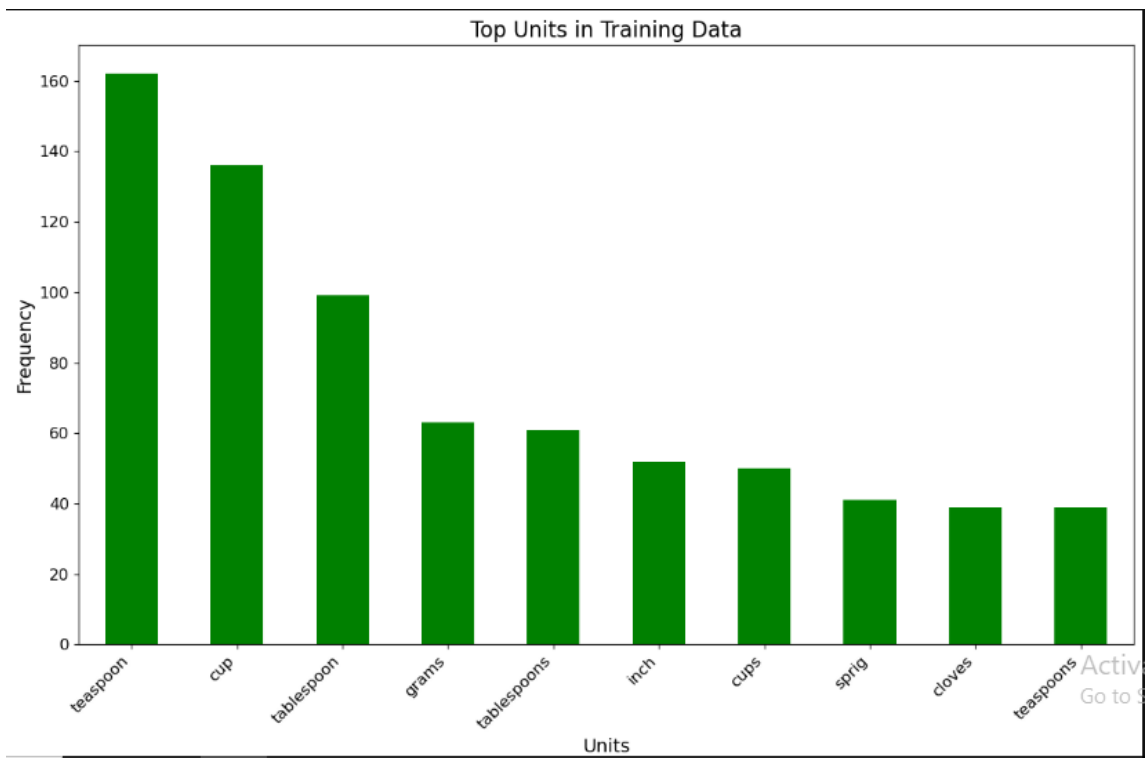
2. It could provide automatic tagging for recipe database, along with could provide context aware search feature
3. It could be also used to track the important ingredient requirement for restaurants based on catalog and sales data for different items.

EDA Analysis

Top 10 Ingredients in Training data:



Top 10 Units in dataset:



Features extracted for each token to provide ML model context about data:

- **Core Features** - The core features of a token should capture its lexical and grammatical properties. Include attributes like the raw token, its lemma, part-of-speech tag, dependency relation, and shape, as well as indicators for whether it's a stop word, digit, or punctuation. The details of the features are given below:
 - bias - Constant feature with a fixed value of 1.0 to aid model learning.
 - token - The lowercase form of the current token.
 - lemma - The lowercase lemma (base form) of the token.
 - pos_tag - Part-of-speech (POS) tag of the token.
 - tag - Detailed POS tag of the token.
 - dep - Dependency relation of the token in the sentence.
 - shape - Shape of the token (e.g., "Xxx" for "Milk").
 - is_stop - Boolean indicating if the token is a stopword.
 - is_digit - Boolean indicating if the token consists of only digits.
 - has_digit - Boolean indicating if the token contains at least one digit.
 - has_alpha - Boolean indicating if the token contains at least one alphabetic character.
 - hyphenated - Boolean indicating if the token contains a hyphen (-).
 - slash_present - Boolean indicating if the token contains a slash (/).
 - is_title - Boolean indicating if the token starts with an uppercase letter.
 - is_upper - Boolean indicating if the token is fully uppercase.
 - is_punct - Boolean indicating if the token is a punctuation mark.

- **Improved Quantity and Unit Detection** - Use key-value pairs to mark the presence of quantities and units in the features dictionary. Utilise the `unit_keywords`, `quantity_keywords`, and `quantity_pattern` to identify and flag these elements. The details of the features are given below:
 - `is_quantity` - Boolean indicating if the token matches a quantity pattern or keyword.
 - `is_unit` - Boolean indicating if the token is a known measurement unit.
 - `is_numeric` - Boolean indicating if the token matches a numeric pattern.
 - `is_fraction` - Boolean indicating if the token represents a fraction (e.g., 1/2).
 - `is_decimal` - Boolean indicating if the token represents a decimal number (e.g., 3.14).
 - `preceding_word` - The previous token in the sentence, if available.
 - `following_word` - The next token in the sentence, if available.
- **Contextual Features** - Incorporate contextual information by adding features for the preceding and following tokens. Include indicators like BOS and EOS to mark the beginning and end of the sequence, and utilise `unit_keywords`, `quantity_keywords`, and `quantity_pattern` to identify the types of neighboring tokens. The features are given below:
 - `prev_token` - The lowercase form of the previous token.
 - `prev_is_quantity` - Boolean indicating if the previous token is a quantity.
 - `prev_is_digit` - Boolean indicating if the previous token is a digit.
 - `BOS` - Boolean indicating if the token is at the beginning of the sentence.
 - `next_token` - The lowercase form of the next token.
 - `next_is_unit` - Boolean indicating if the next token is a unit.
 - `next_is_ingredient` - Boolean indicating if the next token is not a unit or quantity.
 - `EOS` - Boolean indicating if the token is at the end of the sentence.
 -

Weights used for each tag:

quantity: 7.2592

unit: 8.7719

ingredient: 0.3341 (A penalty of 50% applied to put the weight down.)

CRF Model Hyperparameters Explanation

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

Model Performance metrics in Training Data:

Precision, recall and f1-score matrix:

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

Confusion Matrix for Training data:

```
Confusion Matrix:
Labels: ['unit', 'ingredient', 'quantity']
[[ 811   0   0]
 [   0 5323   0]
 [    2    0 978]]
```

Model Performance metrics in validation Data:

Precision, recall and f1-score matrix:

	precision	recall	f1-score	support
ingredient	1.00	1.00	1.00	2107
quantity	1.00	0.99	0.99	411
unit	0.99	0.99	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

Confusion Matrix for Validation data:

```
Confusion Matrix:
Labels: ['unit', 'ingredient', 'quantity']
[[ 355   1   2]
 [   0 2107   0]
 [   4   1 406]]
```


Error Analysis on validation data:

```
Error Analysis by Label:
Label: quantity | Errors: 5 | Class Weight: 7.26
Label: unit | Errors: 3 | Class Weight: 8.77
```

	token	true_label	pred_label	prev_token	next_token	class_weight
0	is	quantity	unit	Pur	2	7.259184
1	+1/3	quantity	ingredient	Milk	extra	7.259184
2	for	quantity	unit	Oil	kneading	7.259184
3	diced	unit	ingredient	Gajjar	small	8.771887
4	to	unit	quantity	10	12	8.771887
5	a	unit	quantity	Haladi	pinch	8.771887
6	pinch	quantity	unit	Dal	Asafoetida	7.259184
7	cloves	quantity	unit	Tomatoes	Garlic	7.259184

Overall accuracy on validation data: **0.9972183588317107**

Evaluation and Conclusions

 **Overview** The model demonstrates **exceptional generalization performance** on the validation set, achieving a near-perfect overall accuracy of **99.72%**. The precision, recall, and F1-scores across all labels are uniformly high, indicating strong alignment between predicted and true labels.

Classification Metrics

	precision	recall	f1-score	support
ingredient	1.00	1.00	1.00	2107
quantity	1.00	0.99	0.99	411
unit	0.99	0.99	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

Confusion Matrix

Confusion Matrix: Labels:

True \ Pred	Unit	Ingredient	Quantity
Unit	355	1	2
Ingredient	0	2107	0
Quantity	4	1	406

Observations:

Unit: Out of 358 true "Unit" labels, 355 were correctly predicted, 1 was misclassified as "Ingredient", and 2 as "Quantity".

Ingredient: The model performed very well here — all 2107 "Ingredient" instances were correctly classified with no confusion.

Quantity: Out of 411 true "Quantity" instances, 406 were correct, 4 were incorrectly predicted as "Unit", and 1 as "Ingredient".

✗ Error Analysis for Quantity Class (Class Weight = 7.26) and Unit Class (Class Weight = 8.77)

The model's misclassifications are concentrated around the "quantity" and "unit" labels, with a small number of confusions but important semantic implications.

◆ Label: quantity Errors: 5

Class Weight: 7.26

The model often confuses quantity with unit or ingredient.

Examples:

"is" (following "Pur") was incorrectly labeled as unit instead of quantity. "+1/3"

(between "Milk" and "extra") was misclassified as ingredient.

"for" (following "Oil" and before "kneading") was labeled as unit instead of quantity.

Insight: Tokens representing fractional amounts, measurements within instructions, or non-numeric indicators (e.g., "pinch", "cloves") are difficult for the model to classify correctly.

◆ Label: unit Errors: 3

Class Weight: 8.77

The confusion is mainly with quantity and ingredient.

Examples:

"diced" (following "Gajjar", before "small") was predicted as ingredient.

"to" and "a" (near numeric tokens like "10", "12" or ingredients like "Haldi", "pinch") were misclassified as quantity.

Insight: Descriptive tokens like "diced" and contextually vague determiners like "a" and "to" can mislead the model.

[Write your answer]

Overview

The model demonstrates **exceptional generalization performance** on the validation set, achieving a near-perfect overall accuracy of **99.72%**. The precision, recall, and F1-scores across all labels are uniformly high, indicating strong alignment between predicted and true labels.

Classification Metrics

precision recall f1-score support

ingredient 1.00 1.00 1.00 2107 quantity 1.00 0.99 0.99 411 unit 0.99 0.99 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

Confusion Matrix

Confusion Matrix: Labels:

True \ Pred	Unit	Ingredient	Quantity
Unit	355	1	2
Ingredient	0	2107	0
Quantity	4	1	406

Observations:

- Unit: Out of 358 true "Unit" labels, 355 were correctly predicted, 1 was misclassified as "Ingredient", and 2 as "Quantity".
- Ingredient: The model performed very well here — all 2107 "Ingredient" instances were correctly classified with no confusion.

Quantity: Out of 411 true "Quantity" instances, 406 were correct, 4 were incorrectly predicted as "Unit", and 1 as "Ingredient".

Error Analysis for Quantity Class (Class Weight = 7.26) and Unit Class (Class Weight = 8.77)

The model's misclassifications are concentrated around the "quantity" and "unit" labels, with a small number of confusions but important semantic implications.

♦ Label: quantity Errors: 5

Class Weight: 7.26

The model often confuses quantity with unit or ingredient. Examples:

- "is" (following "Pur") was incorrectly labeled as unit instead of quantity.
- "+1/3" (between "Milk" and "extra") was misclassified as ingredient.
- "for" (following "Oil" and before "kneading") was labeled as unit instead of quantity.

Insight: Tokens representing fractional amounts, measurements within instructions, or non-numeric indicators (e.g., "pinch", "cloves") are difficult for the model to classify correctly.

♦ Label: unit Errors: 3

Class Weight: 8.77

The confusion is mainly with quantity and ingredient.

Examples:

- "diced" (following "Gajjar", before "small") was predicted as ingredient.
- "to" and "a" (near numeric tokens like "10", "12" or ingredients like "Haldi", "pinch") were misclassified as quantity.
- Insight: Descriptive tokens like "diced" and contextually vague determiners like "a" and "to" can mislead the model.

Model Strengths

The model shows high accuracy but contextual ambiguity between units and quantities leads to misclassifications.

Class imbalance (indicated by weights like 7.26 and 8.77) may also contribute to the challenge in learning accurate boundaries between these two classes.

Errors typically occur with:

- ambiguous measurement-related tokens,
- fractional or descriptive values,
- or tokens appearing in procedural phrases (e.g., "for kneading").

Recommendations for Further Refinement

- The model achieves an impressive *99.72% accuracy*, with minor misclassifications primarily between quantity and unit due to contextual ambiguity.
- Improvement can be made by *augmenting underrepresented classes*, enhancing *contextual features*, and refining edge-case annotations.
- Incorporating *transformer-based models* or *syntactic cues* could further boost performance and robustness.

Conclusion

The model demonstrates **robust and reliable performance** on the validation dataset, achieving near-perfect metrics across all classes. The strong alignment between training and validation performance suggests effective generalization with no evidence of overfitting.

Notable strengths include:






- **Perfect classification** of high-frequency labels such as ingredient and unit.
- **High precision and recall** across all classes.
- **Minimal misclassifications**, restricted to a small number of ambiguous cases within the quantity class.

While current performance is already at a production-ready level, further gains can be made by refining semantic understanding around context-sensitive or lexically ambiguous tokens. Incorporating domain-specific lexicons, improving contextual modeling, and analyzing hard examples can serve as effective next steps for continued enhancement.

Overall, the model is well-calibrated, highly accurate, and structurally sound for deployment in practical use cases involving food or recipe token classification tasks.

Areas for Improvement

While the model exhibits high accuracy on both training and validation sets, the current dataset size (**204 samples**) imposes significant constraints on the model's **generalization capacity**. Key areas for improvement include:

-  **Dataset Expansion**
 - A limited sample size increases the risk of overfitting and reduces the model's exposure to linguistic and contextual variability.
 - Collecting a larger and more diverse dataset—particularly for underrepresented contexts and rare label combinations— would improve robustness and real-world applicability.
-  **Semantic Enrichment**
 - Incorporating syntactic and semantic features (e.g., part-of-speech tags, dependency parsing) could help resolve confusion in ambiguous phrases like "little" or "taste."
-  **Hard Example Mining**
 - Systematically identify and augment training data with examples that the model misclassifies to refine decision boundaries, especially for the quantity class.
-  **Domain-Specific Lexicons**
 - Introducing curated vocabularies for food-related units, quantities, and ingredients may improve token disambiguation and reinforce class-specific priors.
-  **Data Augmentation**
 - Apply NLP-driven data augmentation techniques (e.g., synonym substitution, sentence paraphrasing, entity shuffling) to synthetically enlarge the dataset while preserving label semantics.

```
# Import necessary Libraries
import json # For handling JSON data
import pandas as pd # For data manipulation and analysis
import re # For regular expressions (useful for text preprocessing)
import matplotlib.pyplot as plt # For visualisation
import seaborn as sns # For advanced data visualisation
import sklearn_crfsuite # CRF (Conditional Random Fields) implementation for sequence modeling
import numpy as np # For numerical computations
# Saving and Loading machine Learning models
import joblib
import random
import spacy

from IPython.display import display, Markdown # For displaying well-formatted output
!pip install sklearn-crfsuite
from fractions import Fraction # For handling fractional values in numerical data
# Importing tools for feature engineering and model training
from collections import Counter # For counting occurrences of elements in a List
from sklearn.model_selection import train_test_split # For splitting dataset into train and test sets
from sklearn_crfsuite import metrics # For evaluating CRF models
from sklearn_crfsuite.metrics import flat_classification_report
from sklearn.utils.class_weight import compute_class_weight
from collections import Counter
from sklearn.metrics import confusion_matrix
```