Recipe Ingredient Classification

Domain Background

I have chosen to implement this project around the Natural Language Processing (NLP) domain. A problem that arises often is the necessity to apply a category to certain parts of an unstructured text. One approach to solve this problem could be named-entity recognition (NER), where e.g. a name of a person, say "Albert Einstein", is assigned a label, say “PERSON”. This is helpful in order to extract relevant information from an unstructured text and storing it in a structured manner.

Personal motivation: I am currently working on a recipes app. One important feature of the app should be the ability to quickly add recipes without the need to type too much. So instead of offering a form to manually enter all the ingredients and preparation steps of a recipe found on the Internet, the user can just enter the link and the app should be able to recognize e.g. what is an ingredient and what is a preparation step

Problem Statement

This is a classification problem where the model takes as input plain text and produces a list of words contained in that text and classifies them as corresponding to a particular entity.

Given the plain text of a recipe’s ingredient, find and label its parts: quantity, unit of measure, ingredient name.

For example, given following list of ingredients in plain text

> INGREDIENTS
> 4 large eggs
> 1/4 cup milk
> pinch salt
> pinch pepper
> 2 tsp. butter

Extract the information in this manner:

> Ingredients = [eggs, milk, salt, pepper, butter]
> Quantities = [4, 1/4, null, null, 2]
> Units = [null, cup, pinch, pinch, tsp.]

Datasets and Inputs

Since I already have a list of recipes (in german), I will use those ingredients and label all parts myself in a semiautomatic way, since they are already stored in a structured manner in the database.

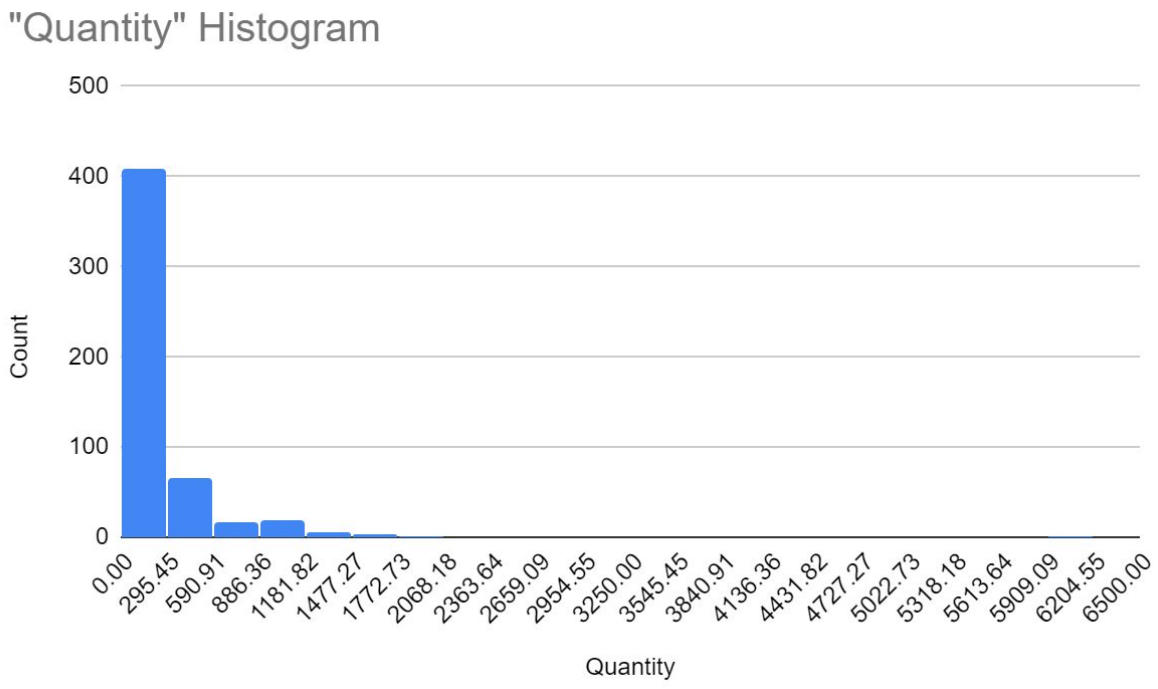
Below you can see an excerpt of the data which contains 573 rows in total:

ingredient	quantity	unit	name
2.00 Becher süße Sahne	2	Becher	süße Sahne
6.00 Blätter Basilikum	6	Blätter	Basilikum
4.00 Blätter Petersilie	4	Blätter	Petersilie
0.52 Bund Minze	0.52	Bund	Minze
0.52 Bund Petersilie	0.52	Bund	Petersilie
1.00 Bund Frühlingszwiebeln	1	Bund	Frühlingszwiebeln
0.52 Bund Petersilie	0.52	Bund	Petersilie
1.00 Bund Frühlingszwiebeln	1	Bund	Frühlingszwiebeln
1.00 Bund Koriander	1	Bund	Koriander
0.52 Bund Petersilie	0.52	Bund	Petersilie
...	...	...	...

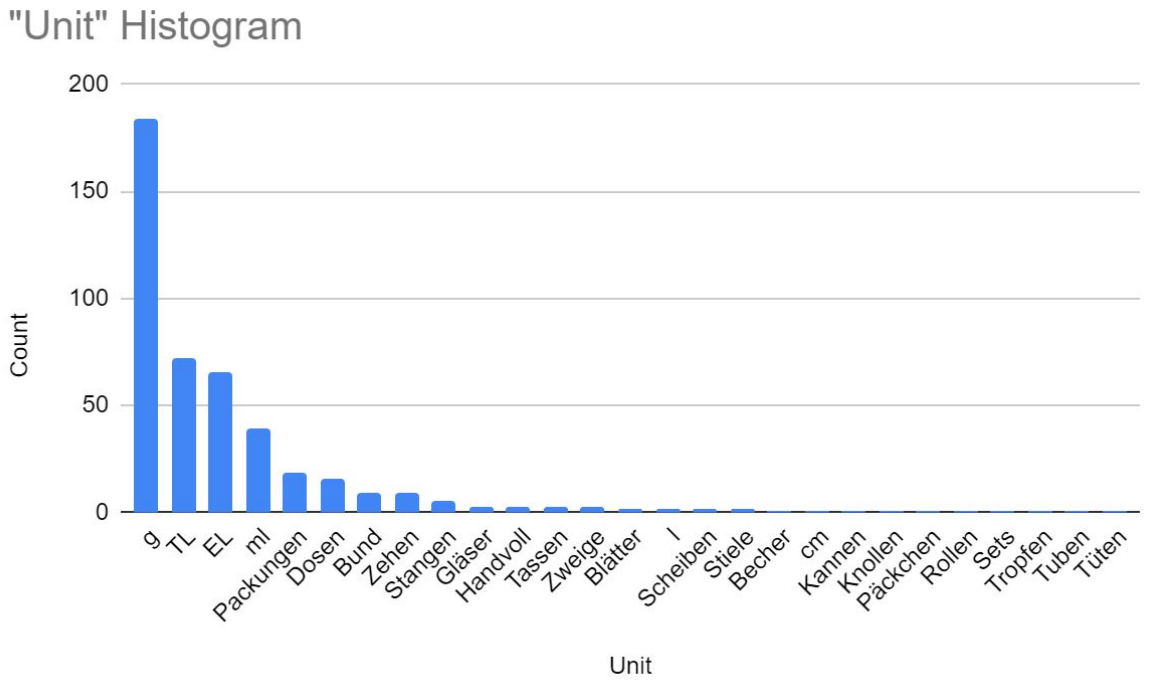
The first column **ingredient** contains the ingredient plain text as we would find it in a recipe description. This plain text contains normally three pieces of information: the quantity, the unit of measure and the ingredient name. This parts were extracted into the following three columns:

The second column **quantity** contains the numeric amount and is somewhat skewed as shown below so it may need some adjustment in the training phase.

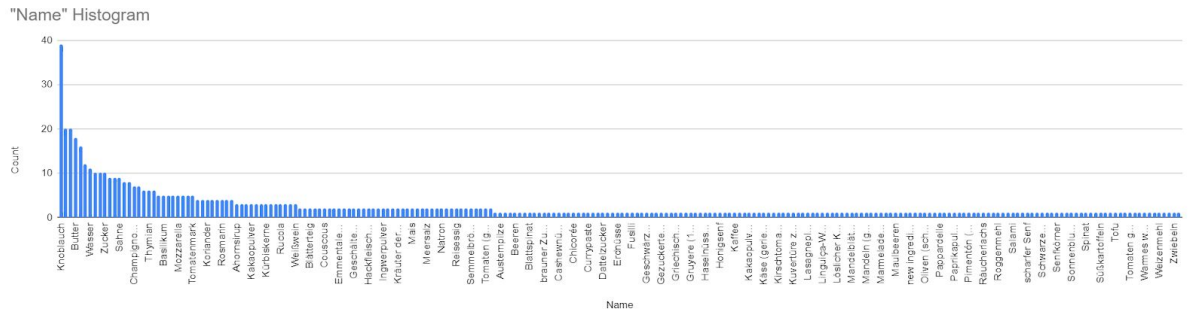
Min: 1
Max: 6000
Average: 179.43



The third column **unit** contains the different types of units of measurement for the ingredients and contains 27 unique values as shown in the diagramm below.



Finally we have the **name** column which contains the name of the ingredients without the quantity and unit. The column contains 230 unique values and has the distribution shown below, showing that common ingredients such as garlic are used in many recipes, whilst other less common ones like wheat germ only appear once.



Solution Statement

This problem can be solved with named-entity recognition. This is a supervised learning problem, so I will need labeled data consisting of plain text representing the full ingredient description and the mapping of the relevant words to their corresponding entity (quantity, unit, name).

Benchmark Model

Whilst doing some online research for solving above problem I came across following two websites:

1. https://www.depends-on-the-definition.com/identify-ingredients-with-neural-networks/
2. https://www.depends-on-the-definition.com/named-entity-recognition-with-bert/

It's not exactly the problem that I'm trying to solve but it can serve as a benchmark model, since it is similar enough but uses a different approach.

Evaluation metrics

I will use **accuracy** as a metric to compare my solution to the benchmark model, although I will play around with other metrics as well (e.g. **f1 score**) and see which one provides best real world performance.

Project Design

I will be using the NLP library spaCy (https://spacy.io) for solving this problem since it offers an easy way to perform named-entity recognition.

The solution is based on the guide at https://spacy.io/usage/training#ner and consists of following steps:

1. Data preparation: Transform the raw labeled data into training data as needed for spaCy's optimizer.
2. Training: Train the model by iterating over the labeled data.

3. Tuning: Tune hyperparameters like number of iterations, dropout, batch size as needed in order to optimize the relevant metrics. Consider integrating rule-based approaches or collect more data if necessary.
4. Testing: Test the model on real unseen data.