A Statistical Machine Learning Approach to Generating Graph Structures from Food Recipes

Master's Thesis

Presented to

The Faculty of the Graduate School of Arts and Sciences
Brandeis University
Graduate Program in Computational Linguistics
James Pustejovsky, Advisor

In Partial Fulfillment
of the Requirements for the Degree

Master of Arts
in
Computational Linguistics

by
Yuzhe Chen

August 2017

ABSTRACT

A Statistical Machine Learning Approach to Generating Graph Structures from Food Recipes

A thesis presented to the Graduate Program in Computational Linguistics

Graduate School of Arts and Sciences
Brandeis University
Waltham, Massachusetts

By Yuzhe Chen

Food recipes, as a type of instructional text, contain series of instructions that guide human users through cooking processes. Their structures can usually be represented as flow graphs: at the start state, a collection of ingredients are introduced. Then, a series of actions are performed on these ingredients (along with cooking tools). Finally, the completed dish is produced at the end.

A computer system that can extract such structures from food recipes can have many useful applications.

Since food recipes are written by humans, they pose many of the same problems the field of Natural Language Processing is trying to solve. Furthermore, they also have challenges that are not commonly seen in many other types of texts.

This thesis explores a Statistical Machine Learning approach to food recipe text processing. I will detail my work in the creation of an annotated food recipe dataset used in supervised learning. I will also describe my work in training Machine Learning algorithms that perform Named Entity Recognition and Relation Extraction tasks on this dataset.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Motivation

Food preparation is one of the most essential activities throughout human history. It is done by people in different regions and cultures across the world, and it is an activity that is carried out by not only professional chefs, but also people who only cook at home occasionally.

Food recipes are often the form of text people use to document the procedures of making dishes, and they are also one of the ways for people to learn how to make a dish. A large amount of recipes are available today. They can be found in physical cookbooks and online with popular websites like `allrecipe.com` and `epicurious.com`

Despite the large amount being created, food recipes still appear to follow a common pattern (an example is shown in Figure 1.1 and 1.2).

They usually have an ingredients section and an instructions section. The ingredients section specifies the types of ingredients being used in the recipe, and their quantities. Meanwhile, the instructions section is usually a collection of step-by-step instructions a user can follow to reproduce the desired dish. The instructions in a recipe can generally be represented using a directed acyclic graph (DAG) structure [14] $G = \{V, A\}$, where G is a DAG, V is a set of vertices, and A is a set of arcs. Vertices represent the food ingredients, actions, tools, etc. in the recipe, and arcs denote the relationships between the vertices.

A that generates such graph representations from recipes would be a significant step in Natural Language Understanding (NLU) of food recipes. It would be an important component in a system

Banana Nut Oatmeal

Ingredients

1/4 cup quick cooking oats
1/2 cup skim milk
1 teaspoon flax seeds
2 tablespoons chopped walnuts
3 tablespoons honey
1 banana, peeled

Instructions

Combine the oats, milk, flax seeds, walnuts, honey, and banana in a microwave-safe bowl. Cook in microwave on High for 2 minutes. Mash the banana with a fork and stir into the mixture. Serve hot.

Figure 1.1: An example recipe, downloaded from Allrecipes.com*

* http://allrecipes.com/recipe/198639/banana-nut-oatmeal/

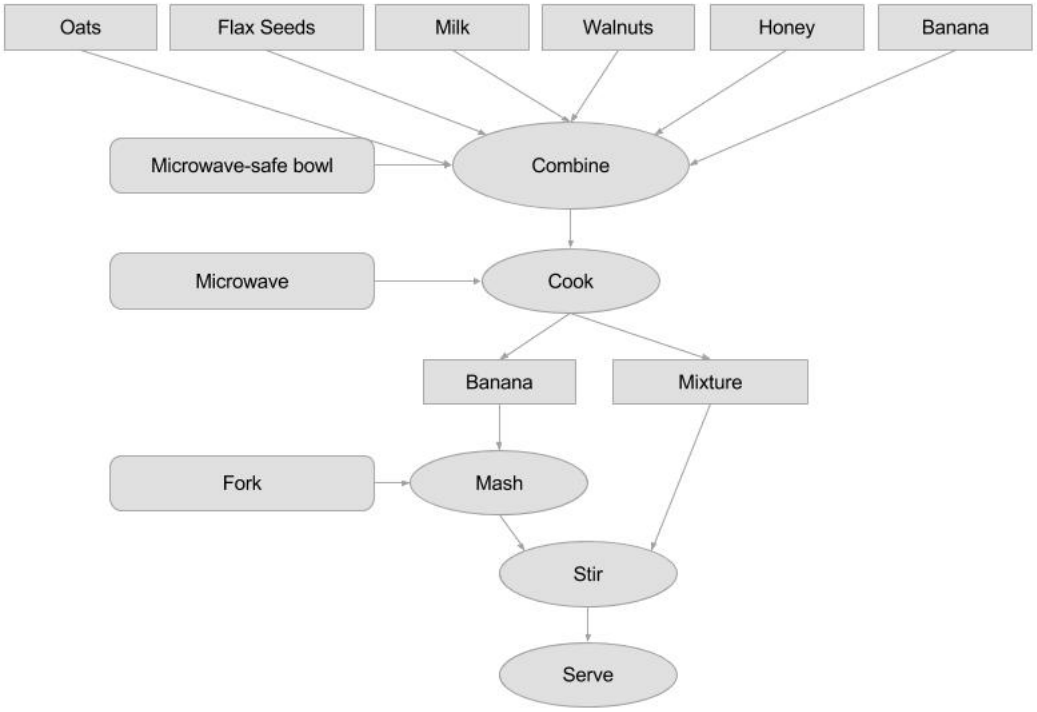

Figure 1.2: A directed acyclic graph (DAG) generated for the recipe text in Figure 1.1. Each vertex represents an ingredient, action, or tool appeared in the recipe text, and the arcs represent the relations between each two vertices.

that understands recipes written by humans, and then automates the cooking process of a dish [2]; it could be used to build more intelligent search engines for food recipes [22], or AI assistants that efficiently plans cooking sessions to help with achieving optimal multi-tasking, according to the steps in a recipe; it could also benefit quantitative studies on food recipes [3] [1], since it would enable us to quickly generate structural representations for a large amount of recipe data.

With that said, creating such a system is not a simple task, since food recipes are written in human natural languages [23]. Natural languages are not interpretable by machines on their own, and they often contain imperfections and ambiguities that hinder the automatic interpretation process.

This thesis explores a statistical machine learning approach to extracting the graph structures from food recipes written in natural language.

Overview

In Chapter 2, previous studies on recipe text processing, and other related studies will be introduced.

Since this thesis explores a supervised machine learning approach for this task. A dataset of recipes annotated by humans should be created. Chapter 3 surveys related existing datasets. Then, the guidelines defined for the annotation task, the issues occurred during the annotation process, and statistics on the dataset will be introduced.

Finally, Chapter 4 contains descriptions of the designs for the experiments performed on various components of a system that generates graph structures, and the performance of the experiments will be reported in Chapter 5.

Chapter 2

Related Work

One of the pioneering work on recipe text processing [6] explored a method to convert a recipe in text form to a flow graph structure. Their method is simply collecting word occurrences and co-occurrences, then categorizing them into various dictionaries. During structural analysis, the dictionaries are used to extract keywords in recipes through simple matching, Then, a rule-based approach is used to associate nouns with verbs, and connect noun-verb sets to form a graph. Since this method requires manually categorized words and their co-occurrences, while recipes are written in human natural languages and therefore are likely to have a lot of variations and unpredictability, a large amount of manual labor is needed in order to compile comprehensive and reliable dictionaries.

Mori et al. [15] proposed a machine learning approach to Japanese recipe text processing. They mainly focused on four tasks: Word Segmentation (more problematic in Japanese which doesn't have explicit word boundaries), Named Entity Recognition, Syntactic Analysis and Predicate-Argument Structure Analysis. Their methods, especially ones used in Named Entity Recognition and Predicate-Argument Structure Analysis, appear to be very informative for this thesis. However, they only covered the task of detecting predicate-argument relations between entities within the same sentence, while this thesis is trying to address the issue of finding the structure of a recipe, which requires finding relations between entities across the whole recipe document.

Tasse et al. [20] proposed MILK (Minimal Instruction Language for the Kitchen), a framework for representing instructional statements in recipes. They have created a database of 300 recipes

annotated in the MILK language. In addition, they have also done some preliminary work on creating a semantic parser for recipes, although, the results are not ideal.

Mori et al. [14] developed a flow graph corpus they call r-FG corpus, their annotation framework not only captures the relations between entities in a recipe, but also the semantic types of each relation. This annotation scheme is similar to the annotation framework developed for this thesis, albeit capturing more semantic information.

Jermsurawong et al. [9] introduced SIMMR (Simplified Ingredient Merging Map in Recipes), which represents a recipe using an ingredient-instruction dependency tree structure. It captures the high-level flow of a recipe but does not model the semantics in each instruction, unlike previous efforts [20], [14]. They transformed the database annotated in MILK [20] into SIMMER trees, and performed linking experiments using this data, which yielded high accuracy. This study differs from mine in that it divides the task of finding relationships between entities into two: Ingredient-Instruction linking and Instruction-Instruction linking. It focuses on linking each ingredient listed in the Ingredients section with a specific instruction in the Instructions section, and also linking instructions such that the workflow in a recipe is captured. It does not cover the food entities that may appear between two actions, which are usually the output of the former action and the input to the latter.

Maeta et al. [13] also proposed a framework for procedural text understanding. They based their work on the annotated recipe corpus developed by Mori et al. [14], and introduced a framework that addresses three major problems: word segmentation, concept identification and flow graph estimation. They used a linear chain conditional random fields (CRF) algorithm for the entity recognition task which they refer to as Concept Identification, and a Maximum Spanning Tree algorithm to tackle the flow graph estimation task (instead of a pairwise classification method which this thesis has taken).

Kiddon et al. proposed an unsupervised machine learning approach to the interpretation of instructional recipes [11], where an unsupervised hard EM approach is used to automatically map recipe instructions to graphs of actions that define what order the actions should take place. The

system demonstrated the ability to produce high quality action graphs, and also learned interesting domain knowledge such as the verb signatures of actions, and ingredient components of composite items.

Since food recipes are a subset of instructional texts, studies on information extraction from instructional texts [24] [19] may also be informative to efforts on recipe text processing.

Chapter 3

Recipe Dataset

Existing Recipe Data

There have been annotated recipe datasets created for the purpose of information extraction research available today. I have found two recipe corpora: the r-FG (Recipe Flow Graph) corpus [14], and the CURD (Carnegie Mellon University Recipe Database) database [20]. For this thesis, I did not use either of the two existing datasets. I will briefly introduce the two corpora, and explain the reasons I decided to create a new dataset for my task.

The r-FG Dataset

The r-FG corpus is a collection of 266 Japanese recipes annotated with the framework in which each recipe is represented by a DAG $G = (V, A)$ where V is a set of vertices that correspond to foods, tools, actions, etc. and A is a set of arcs that denote the relations between the vertices. An example recipe and its visualized graph representation according to the r-FG annotation is shown in Figure 3.1 and 3.2.

There are 8 types of tags that denote the vertices in a recipe graph:

$$V = \{F, T, D, Q, Ac, Af, Sf, St\}$$

the meaning of each tag is explained in Figure 3.3

Title:	
ホットドッグ	(Baked Hot Dog)
Ingredients:	
• フランクフルト 8 つ	(8 frankfurters)
• ホットドッグパン 8 つ	(8 hot dog buns)
• 豆入りのチリ 1 缶 (14.5 オンス)	(1 (14.5 ounce) can of chili with beans)
• カットオニオン 1/2 カップ	(1/2 cup onion (diced))
• 千切りチェダーチーズ 2 カップ	(2 cups shredded cheddar cheese)
• マヨネーズ	(mayonnaise)
• マスタード	(mustard)
• 甘味料	(sweet relish)
Steps:	
1. 各ホットドッグパンの内側にマヨネーズ、マスタード、甘味料を広げる。	
(Spread the inside of each hot dog bun with mayonnaise, mustard and sweet relish.)	
フランクフルトを入れ、13×9”のオーブン皿に置く。	
(Fill with a frankfurter and place into a 13×9” baking dish.)	
2. 各ホットドッグにチリ、チーズ、オニオンをふりかける。	
(Sprinkle each hot dog with chili, cheese, and onion.)	
3. アルミホイルで覆い、オーブンに置く。	
(Cover with aluminum foil and place into the oven)	
そして、350 度で 45 分間焼く。	
(Then bake at 350 degrees for 45 minutes.)	

Figure 3.1: An example recipe, taken from [14]

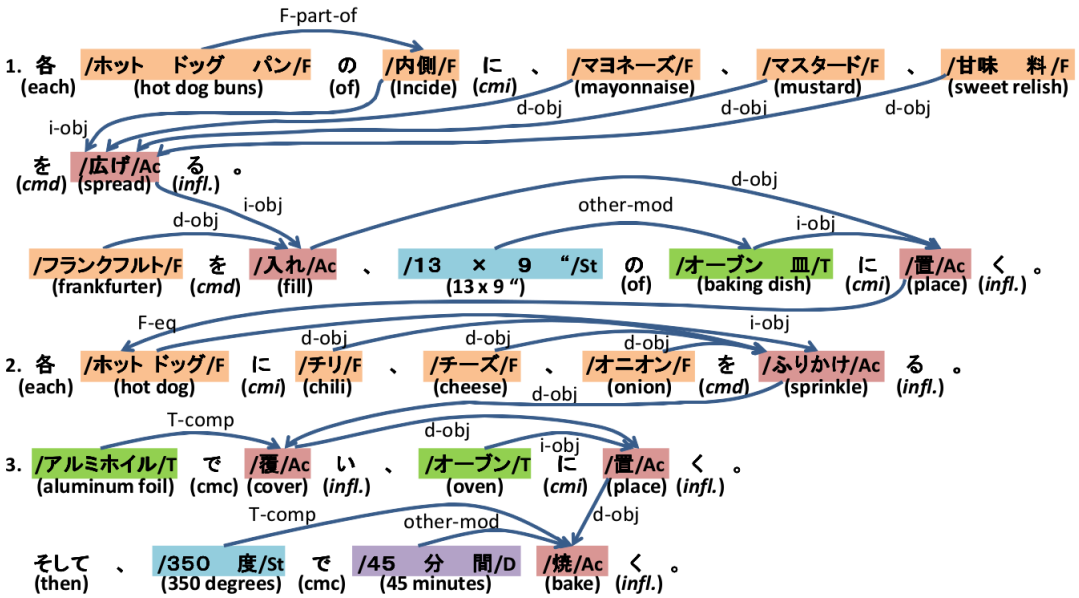

Figure 3.2: A graph representation of the recipe in Figure 3.1, taken from [14]

NE tag	Meaning
F	Food
T	Tool
D	Duration
Q	Quantity
Ac	Action by the chef
Af	Action by foods
Sf	State of foods
St	State of tools

Figure 3.3: Named entity tags in r-FG, taken from [14]

Arcs are used to denote the relationship between two vertices; each arc is associated with an edge label categorizing the type of a relation. A list of edge labels and their meanings have been shown in Figure 3.4

Each recipe is first processed by a word segmenter and a NE tagger, then the word boundaries and IOB-tagged NE annotation are corrected manually. The arcs between vertices are represented in a confusion matrix of vertices and labels, where if there is an arc of label l between vertices v_i and v_j, the ID of v_i is put into the crossing cell between the row of v_j and the column of l.

To determine the language dependency of this annotation framework, an experiment has been performed where two annotators annotated the same recipe in both English and Japanese. The annotators had perfect agreement on vertices, and high agreement on arcs. They concluded that the scheme is mostly language-independent.

CURD Dataset

CURD is a dataset of 300 annotated recipes and 350 unannotated ones. The annotated recipes are annotated using the MILK language, which stands for "Minimal Instruction Language for the Kitchen".

The MILK language defines three primitive types:

- `ingredient`: Food ingredients involved in the recipe.

Edge lable	Meaning
subj	Subject
d-obj	Direct object
i-obj	Indirect object
F-comp	Food complement
T-comp	Tool complement
F-eq	Food equality
F-part-of	Food part-of
F-set	Food set
T-eq	Tool equality
T-part-of	Tool part-of
A-eq	Action equality
V-tm	Head verb of a clause for timing, etc.
other-mod	Other relationships

Figure 3.4: Arc labels in r-FG, taken from [14]

- `tool`: Non-food items in the recipe.
- `string`: Provides extra information and makes the annotation human-readable.

Recipes are defined in terms of states, a state can be described as a tuple $< I, T, S, I_d, T_d, C >$, where:

- I: A set of `ingredient` entities
- T: A set of `tool` entities
- S: A set of `string` entries
- I_d: Relations between ingredients and their descriptions ($I \times S$)
- T_d: Relations between tools and their descriptions ($T \times S$)
- C: Relations describing which tools contain which ingredients ($T \times I$)

```
<line>
  <originaltext>Heat oil in a large pot; brown chicken.</originaltext>
  <annotation>create_tool(t0, "large pot")</annotation>
</line>
<line>
  <originaltext>Heat oil in a large pot; brown chicken.</originaltext>
  <annotation>put(ing0, t0)</annotation>
</line>
<line>
  <originaltext>Heat oil in a large pot; brown chicken.</originaltext>
  <annotation>cook(ing0, t0, ing16, "heated oil", "heat oil")</annotation>
</line>
<line>
  <originaltext>Heat oil in a large pot; brown chicken.</originaltext>
  <annotation>
    combine({ing1, ing16}, ing17, "oil and chicken", "")
  </annotation>
</line>
<line>
  <originaltext>Heat oil in a large pot; brown chicken.</originaltext>
  <annotation>put(ing17, t0)</annotation>
</line>
<line>
  <originaltext>Heat oil in a large pot; brown chicken.</originaltext>
  <annotation>
    cook(ing17, t0, ing18, "browned chicken", "brown chicken")
  </annotation>
</line>
```

Figure 3.5: A sample annotation for the text "Heat oil in a large pot; brown chicken." in the MILK language, found in the CURD dataset.

The sets are empty in the start state, as the recipe progresses, items are added or deleted to the sets.

MILK defines 12 actions, which correspond to functions: `create_ing`, `create_tool`, `combine`, `separate`, `put`, `remove`, `cut`, `mix`, `cook`, `do`, `serve`, `set`, `leave` and `chefcheck`. They are used to interpret and represent actual actions in a recipe.

An example annotation is shown in Figure 3.5

Motivation for Creating a New Dataset

The two existing datasets introduced above are well annotated and have been used in various studies on recipe text processing [9, 13, 21]. However, my goal for this thesis is to perform information extraction experiments on the NER and relation extraction steps involved in the generation of graph structures for food recipes written in English. The r-FG corpus appears to be a good dataset for

NER and relation extraction experiments, but it is composed of only Japanese food recipes; The CURD dataset, on the other hand, contains a large amount of annotated datasets, but the MILK annotation scheme is not detailed enough to identify each action that occurs during cooking, which is important for training models that are able to identify the vertices of a recipe graph structure.

For these reasons, I decided to create a new dataset for my thesis.

Creating the English Recipe Graph Dataset

Software and Tools Used

recipe-scraper

`recipe-scraper`[1] is a Python library for scrapping recipe texts from recipe websites. Its support for Allrecipes and Epicurious had stopped working at the time of recipe collection, due to changes to the two websites. I have forked the project[2] and fixed the issues for the scrapers of these two websites.

MAE (Multi-document Annotation Environment)

MAE[3] [18] is a general-purpose natural language annotation software. In addition to annotation, it also has Adjudication and Inter-Annotator Agreement calculation features integrated.

For annotation, MAE requires a task definition file specifying the elements in an annotation framework, and outputs the annotated documents in XML format.

It uses XML tags as its form of annotation, and it supports two main types of tags: extent tags and link tags. Extent tags are used to mark parts of a file being annotated using character index ranges, while link tags are used to associate multiple extent tags together. Attributes can also be defined for extent tags to encode additional information.

[1] https://github.com/hhursev/recipe-scraper
[2] https://github.com/yzxchn/recipe-scraper
[3] https://github.com/keighrim/mae-annotation

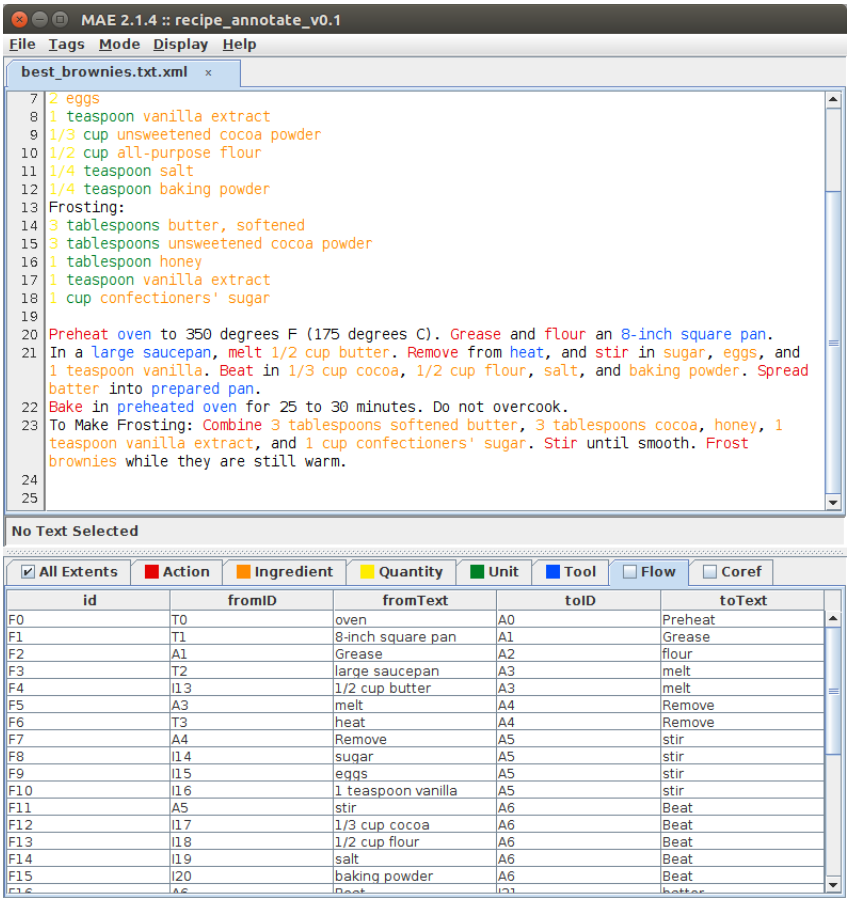

Figure 3.6: The MAE annotation interface

An example of the MAE annotation interface is shown in Figure 3.6, and a sample XML output is shown in Figure 3.7.

```
<TAGS>
  <Action id="A0" spans="344~351" text="Preheat" act_type="SetTool"/>
  <Action id="A1" spans="391~397" text="Grease" act_type="SetTool"/>
  <Action id="A2" spans="402~407" text="flour" act_type="SetTool"/>
  <Action id="A3" spans="451~455" text="melt" act_type="Cook"/>
  <Action id="A4" spans="472~478" text="Remove" act_type="Remove"/>
  <Action id="A5" spans="494~498" text="stir" act_type="Mix"/>
  <Action id="A6" spans="539~543" text="Beat" act_type="Mix"/>
  <Action id="A7" spans="602~608" text="Spread" act_type="Put"/>
  <Action id="A8" spans="635~639" text="Bake" act_type="Cook"/>
```

Figure 3.7: The extent tags of a sample XML output of MAE

```
Amazing Chicken

40

4 skinless, boneless chicken breast halves
salt and pepper to taste
1/2 cup mayonnaise
2 cups Italian seasoned bread crumbs

Preheat the oven to 425 degrees F (220 degrees C). Grease a
shallow baking dish.
Season chicken breasts. Coat chicken on all sides with mayonnaise,
and roll in bread  crumbs until coated. Place coated breasts into
the prepared pan.
Bake uncovered for 30 minutes in the preheated oven, or until
chicken is no longer pink in the middle and the juices run clear.
```

Figure 3.8: An example of an unannotated recipe. The four sections (title, cooking time, ingredients and instructions) are separated by blank links.

Data Collection

100 recipes with various lengths have been text-mined from Allrecipes[4] and Epicurious[5], using the Python library `recipe-scraper` (See Section 3.2.1), and stored in plain text files. Each recipe file contains the title of the recipe, the total cooking time (in minutes), a list of ingredients, and the cooking instructions. Each section is separated by a blank line. A example is shown in Figure 3.8.

Annotation Framework

The task definition for the English recipe graph annotation is similar to the one used in the r-FG corpus (See Section 3.1.1). Two categories of tags are defined in accordance to MAE task definition format: Extent Tags and Link Tags. Next, I will introduce each tag within the two categories.

[4] www.allrecipes.com
[5] www.epicurious.com

Attribute	Meaning	Possible Values
`start_state`	Whether the ingredient is at a start state.	`Yes` / `No`
`composite`	Whether the ingredient is composed of other ingredients.	`Yes` / `No`

Table 3.1: The attributes of an `Ingredient` tag.

Extent Tags

Extent tags are used to label spans of text within a recipe document. There are 5 types of extent tags in this annotation scheme:

- `Ingredient` tags: Used to denote sequences of a recipe where food ingredients are involved, which include all the items listed in the ingredients section in the recipe. Food items that are referred to in the instructions section are also given Ingredient tags. For example, the term "dry ingredients" in the instruction "Put dry ingredients in a mixing bowl" is tagged as an Ingredient.

  An ingredient tag also has two attributes that the annotator needs to determine. They are listed in Table 3.1.

  The `start_state` attribute is used to denote if the ingredient is in a start state in the recipe. It is in a start state when it hasn't participated in any cooking actions. Another way to think of it is a start state ingredient vertex does not have any arcs pointing to it in a recipe graph.

  The `composite` attribute indicates whether an ingredient is composed of other ingredients in a recipe. For example, a recipe may list flour, sugar and instant dry yeast in the ingredients section, then refer to them as "dry ingredients" in a cooking step. The term "dry ingredient" then should be given an `Ingredient` tag that has the `composite` attribute set to `Yes`.

- `Action` tags: Used to mark spans of text in the recipe that are cooking actions. Only actions from the user of the recipes are marked, which correspond to the Ac tags in the r-FG annotation framework. Verbs whose agent is not the user of the recipe are not marked (the type of terms which largely overlaps with the category Af (Action by foods) in the r-FG corpus). For example:

Attribute	Meaning	Possible Values
act_type	The type of the action	combine / separate / put / remove / cut / mix / cook / serve/ settool / leave / check / other

Table 3.2: The attribute of the Action tag, and its possible values

(1) Put dry ingredients in a mixing bowl.

(2) Let mixture sit for an hour.

In (1), "put" is tagged as Action because the agent of the action is the user. Meanwhile, in (2), the verb "let" is tagged, but not the verb "sit", because the agent of "sit" is the food ingredient "mixture".

An Action tag has an attribute act_type, which is shown in Table 3.2. The values of the attribute are largely inspired by the MILK language (See Section3.1.2), which uses a limited set of actions to translate the cooking actions in a recipe. The new annotation scheme defined here uses a limited set of actions as categories for actions tagged, and an annotator is expected to select the category that an action is closest to semantically.

- Tool tags: Tool tags are used to tag sequences of the recipe that are tools, utensils, or other non-food items being used. For example:

  (1) Put dry ingredients in a mixing bowl

  (2) Heat cast iron pan on medium heat

  Not only physical objects, other non-food NPs that are involved in the cooking process are also tagged, such as "medium heat" in (2).

- Quantity tags: Used to denote the quantity of an ingredient in the ingredients section:

  (1) 1 1/2 cups sour cream

  (2) 2 eggs

```
<Flow id="F0" fromID="T0" fromText="a large bowl" toID="A0" toText="dissolve"/>
<Flow id="F1" fromID="I6" fromText="the sugar" toID="A0" toText="dissolve"/>
<Flow id="F2" fromID="I7" fromText="warm water" toID="A0" toText="dissolve"/>
<Flow id="F3" fromID="A0" fromText="dissolve" toID="A1" toText="stir"/>
```

Figure 3.9: Some `Flow` tags in the output XML format

- `Unit` tags: Used to mark the unit of measurement for an ingredient listed in the ingredients section, usually in conjunction with a `Quantity` tag:

  (1) 1 1/2 cups sour cream

  (2) 1 tablespoon cornstarch

Link Tags

Similar to arcs in the r-FG annotation framework, link tags are used to represent relations between entities marked by extent tags. Unlike r-FG, there are only two types of link tags defined for this task:

- `Flow` tags: Used to represent the workflow in a recipe. Each `Flow` tag associates 2 extent tags together by their IDs. They can be seen as the directed edges that connects two entities in a visualized recipe graph (See Figure 1.2 in Section 1.1).

  Since each edge is directed, a `Flow` tag needs to be able to represent directionality between the two associated tags. It does this by denoting the first entity as `from` and the second as `to`. Therefore, if `from` is e_1 and `to` is e_2, the relationship is $e_1 \rightarrow e_2$.

  Several example link tags represented in the XML format is shown in Figure 3.9.

- `Coref` tags: Represent co-reference relationships between entities.

  A co-reference relationship in this annotation task appears when two terms refer to the same item in the same state. An item goes through a state change when it is modified by an action in the cooking procedure. This item then is considered to be in a new state. For example:

  (1) Boil eggs in the pot, then transfer the eggs to cold water.

```
<Coref id="C2" exp_aID="I2" exp_aText="active dry yeast" exp_bID="I8" exp_bText="yeast"/>
<Coref id="C3" exp_aID="I3" exp_aText="salt" exp_bID="I9" exp_bText="salt"/>
<Coref id="C4" exp_aID="I4" exp_aText="vegetable oil" exp_bID="I10" exp_bText="oil"/>
<Coref id="C5" exp_aID="I5" exp_aText="bread flour" exp_bID="I12" exp_bText="flour"/>
```

Figure 3.10: Some `Coref` tags in the output XML format.

(2) Cut cucumber into slices, then sprinkle each cucumber slice with salt.

In (1), the two underlined terms are not co-referencing, because the entity referred to by the first term, "eggs" undergoes a state change after being modified by the action "boil", resulting in the item referred to by the second term, "the eggs".

In (2), however, the two underlined terms are co-referencing, because the two terms are both referencing the sliced cucumbers, and this entity hasn't been modified by any cooking actions between the two terms.

During annotation of a recipe, most of the co-reference relationships seem to occur between food item mentions in the ingredients section and the instructions section, where these food items are first mentioned there. However, co-reference relations may happen within an instructions section as well. An example of this type of co-reference relations is in example (2) above. Such relationships do not occur very often, because a recipe instruction does not typically state the resulting item after being modified by the action of the instruction.

In a `Coref` tag, the two co-referencing terms are denoted as `exp_a` and `exp_b`, which stand for "expression A" and "expression B". Some sample `Coref` tags in the output XML format are shown in Figure 3.10.

Annotation Process

Recipes are annotated in the following procedures:

1. The `Ingredient` items, and their corresponding `Quantity` and `Unit` terms (if they exist) are identified and tagged.

Tag Name	Counts
Ingredient	1756
Action	1068
Tool	493
Quantity	663
Unit	559

Table 3.3: The counts of extent tags in each category in the annotated recipe corpus.

Link Type	Counts
Ingredient-Action	1024
Action-Ingredient	627
Action-Action	315
Tool-Action	475
Action-Tool	98

Table 3.4: The counts of Flow tags in each type in the annotated recipe corpus. An Ingredient-Action type means that the flow link connects from an Ingredient to an Action.

2. Action, Ingredient and Tool tags are given to terms in the instructions section.

3. The attributes of Action and Ingredient tags are selected.

4. Flow and Coref tags are created, then set according to the structure of the recipe.

Since I annotated the whole dataset myself, I did not perform adjudication and inter annotator agreement score calculation.

The statistics on the dataset of 100 recipes are shown in Table 3.3, Table 3.4 and Table 3.5.

Link Type	Counts
Ingredient-Ingredient	709
Tool-Tool	2

Table 3.5: The counts of Coref tags in each type in the annotated recipe corpus. An Ingredient-Ingredient type means that the coref link connect an Ingredient and an Ingredient.

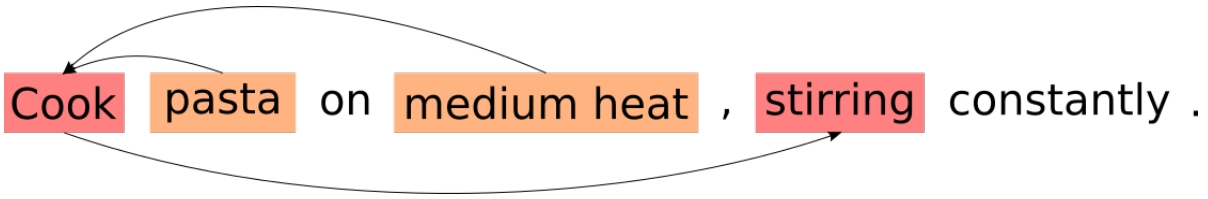

Figure 3.11: When one action occurs in the duration of another action, the annotation is as if the actions are sequential.

Observations and Notes

There have been several issues that were noticed during the annotation process, that the current annotation framework does not handle very well.

One issue is that the framework can't represent the situation where two actions happen concurrently, or an action is "nested" in another action—an action happens within the duration of another action. For example:

> Cook pasta on medium heat, stirring constantly.

"stirring" is in fact happening during the action of "cook", but the annotation framework can not handle actions of this nested nature. Instead, they are annotated as two disjoint actions taking place sequentially. As shown in Figure 3.11

Also, the currently annotation scheme can not represent the detailed interactions between an action and the ingredients and tools that it modifies:

> Mix sugar into the yeast.

The current framework does not represent the details of the action where "sugar" is mixed into "the yeast". Instead, sentences like this are annotated as shown in Figure 3.12.

Finally, in the first version of the annotated recipes, `Quantity` and `Unit` tags were given to corresponding sequences of text in the instructions section. Later, such tags were removed, and such measurement information is instead covered by the following `Ingredient` tag. The reasoning behind this is that there are very few instances of `Quantity` and `Unit` tags in the instruction

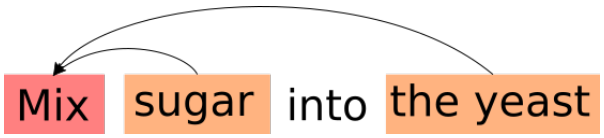

Figure 3.12: The annotation framework does not represent the detailed interactions between an action and the items it modifies.

sections of recipes in this dataset, and the amount may not be enough for training machine learning algorithms that automatically tag such sequences.

Chapter 4

Recipe Text Processing Experiment Design

With the dataset of annotated recipes, experiments on the steps of a system that generates the graph structures for recipes were able to be performed. In this chapter, the data preparation process and the designs of these experiments will be introduced. The evaluation of these experiments, including the comparison to the baselines and error analysis, will be presented in Chapter 5.

Preprocessing

Annotated recipes have been preprocessed and transformed into a format that is easier to work with in the following experiments. I will introduce the tools used, the steps of transforming the data, and the format of the data.

Tools Used

spaCy

spaCy[1] is a Python library for Natural Language Processing. It is chosen for the experiments in this study because of its high accuracy and speed at dependency parsing [4] [8]. In addition, because the annotated recipe dataset is not pre-tokenized, spaCy's non-destructive nature at tokenization is fitting for this task.

[1] https://spacy.io/

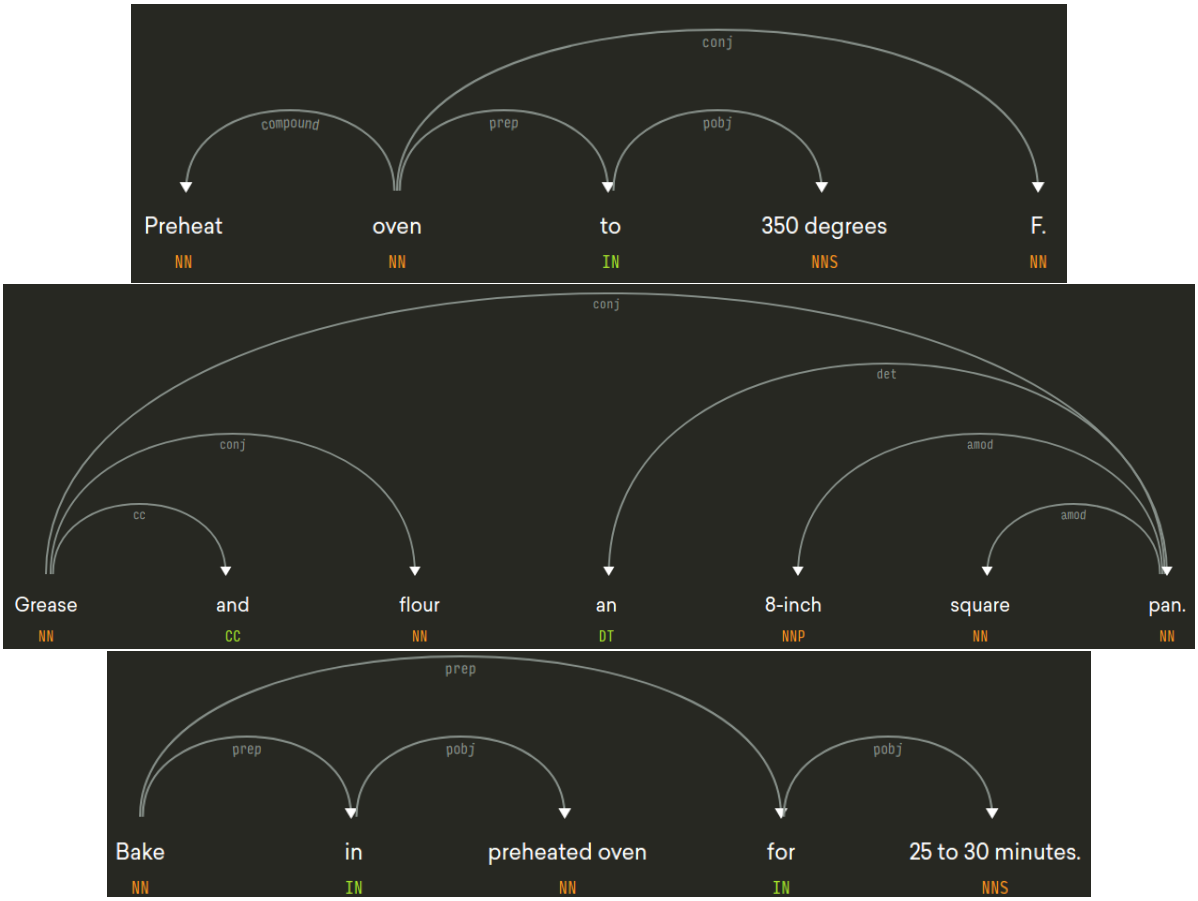

Figure 4.1: It is not difficult to find sentences in the recipes that spaCy would tag and parse incorrectly. Using displaCy[1] (a spaCy POS tagging and parsing visualizer) , we can clearly see that spaCy incorrectly tagged verbs as NN (noun) in three sentences found in a single recipe in the recipe dataset.

[1] https://demos.explosion.ai/displacy/

A point to be noted is that, even though spaCy has been proven to perform well, its accuracy at domain-specific, recipe text parsing is still unknown, and I could not find any study that evaluates its performance on such texts. In fact, recipe text may cause noticeable differences in parsing accuracy, due to the fact that they are overwhelmingly comprised of imperative sentences, and it appears that imperative sentences pose some difficulties in parsing compared to sentences in the general domain [7]. In fact, it is not difficult to find sentences in recipes that spaCy would incorrectly tag and parse (Shown in Figure 4.1). Due to time and resource constraints, I did not perform any evaluation and domain adaption for spaCy on recipe texts.

Procedures

Each annotated recipes in the XML format is preprocesssed and transformed into a format that is easier to work with in the upcoming experiments, following the steps below:

(1) Annotated recipes in XML format are loaded, each recipe text is divided into title, cooking time, ingredients and instructions by blank lines. (An example of the format of a recipe is shown in Figure 3.8)

(2) For each line in the ingredients section, load it into spaCy, and then use spaCy to tokenize and POS tag the sentence, and align tokens with the original text such that the correct index of each token in the text is acquired. The annotation tag of a token is also extracted according to the token's index range in the original text.

(3) The entire body of the instructions section is loaded into spaCy, then use spaCy to perform sentence segmentation, tokenization, POS tagging and dependency parsing on the text. Each token is also aligned with the original text to get its starting and end index in the text, and the annotation tag information of each token is extracted.

(4) The outputs of the preprocessed ingredients section text and instructions section text is stored into two separate files, with the ingredient section's output file containing the annotation information, starting and end indices and POS tag information, and the instructions sections' output file containing the same categories of information, with additional dependency parse information. A more detailed introduction of the formats of the output files will be provided in Section 4.1.3

(5) The link tag information is also stored in a separate file for each recipe.

Thus, each annotated recipe is transformed into 3 files, one containing ingredient section tokens, one containing instruction section tokens, and finally one containing link tag information between the entities in the recipe.

Data Format

All outputs of preprocessing are stored in plain text files. For the ingredient and instruction files, each line in a file represents a token in the original text and additional information associated with it, and each column is a type of additional information corresponding to each token. For link files, each line corresponds to a link tag in the original annotation.

Ingredient File Format

Preprocessed ingredient files contain the following types of information, each type corresponds to one column in the file:

- **Tag ID**: the ID of the annotation extent tag that "covers" this token (the token is part of a phrase that has been given the extent tag)
- **Begin Index**: the beginning index of the token in the original text.
- **End Index**: the end index of the token in the original text.
- **Raw Text**: the raw text of the token.
- **POS tag**: the POS tag in the OntoNote 5 version of the Penn Treebank tag set, given by spaCy.
- **Simple POS tag**: the corresponding tag in the Google Universal POS tag set, given by spaCy.

A sample ingredient file in the format described above is shown in Figure 4.2.

Instruction File Format

A preprocessed instruction file contains similar fields of information for each token as an instruction file. In addition, the following fields of information are also included in the transformed files.

```
Q0      19      22      1/2         CD      NUM
U0      23      26      cup         NN      NOUN
I0      27      33      butter      NN      NOUN

Q1      34      35      1           CD      NUM
U1      36      39      cup         NN      NOUN
I1      40      45      white       JJ      ADJ
I1      46      51      sugar       NN      NOUN

Q2      52      53      2           CD      NUM
I2      54      58      eggs        NNS     NOUN

Q3      59      60      1           CD      NUM
U2      61      69      teaspoon            NN      NOUN
I3      70      77      vanilla     NN      NOUN
I3      78      85      extract     NN      NOUN

Q4      86      89      1/3         CD      NUM
U4      90      93      cup         NN      NOUN
I4      94      105     unsweetened         VBD     VERB
I4      106     111     cocoa       NN      NOUN
I4      112     118     powder      NN      NOUN
```

Figure 4.2: A sample ingredient file showing its transformed format.

```
-       430     432     In          in          IN      ADP     prep    melt        melt        VB      VERB    ROOT    melt        melt        VB      VERB    ROOT
-       433     434     a           a           DT      DET     det     saucepan            saucepan            NN      NOUN    pobj        In          in      IN      ADP     prep
T2      435     440     large       large       JJ      ADJ     amod    saucepan            saucepan            NN      NOUN    pobj        In          in      IN      ADP     prep
T2      441     449     saucepan                saucepan            NN      NOUN    pobj        In          in      IN      ADP     prep        melt        melt    VB      VERB    ROOT
-       449     450     ,           ,           ,       PUNCT   punct   melt        melt        VB      VERB    ROOT    melt        melt        VB      VERB    ROOT
A3      451     455     melt        melt        VB      VERB    ROOT    melt        melt        VB      VERB    ROOT    melt        melt        VB      VERB    ROOT
I13     456     459     1/2         1/2         CD      NUM     nummod  cup         cup         NN      NOUN    compound            butter      butter  NN      NOUN    dobj
I13     460     463     cup         cup         NN      NOUN    compound            butter      butter  NN      NOUN    dobj        melt        melt    VB      VERB    ROOT
I13     464     470     butter      butter      NN      NOUN    dobj    melt        melt        VB      VERB    ROOT    melt        melt        VB      VERB    ROOT
-       470     471     .           .           .       PUNCT   punct   melt        melt        VB      VERB    ROOT    melt        melt        VB      VERB    ROOT
```

Figure 4.3: A sample instruction file shown in its transformed format.

- **Dependency Label**: the dependency label of the arc that connect to this token as its head.
- **Head Text**: the text of the head token of this token.
- **Head POS tag**: the POS tag of the head token of the current token in the OntoNote5 version of the Penn Treebank tag set.
- **Head Simple POS tag**: the POS tag of the head token of the current token in the Google Universal POS tag set.

Additionally, the same types of information for the head of the head of the current token are also included.

A sample instruction file is shown in Figure 4.3.

```
F11     A5      stir    A6      Beat
F12     I17     1/3 cup cocoa   A6      Beat
F13     I18     1/2 cup flour   A6      Beat
F14     I19     salt    A6      Beat
F15     I20     baking powder   A6      Beat
F16     A6      Beat    I21     batter
F17     I21     batter  A7      Spread
F18     A2      flour   T4      prepared pan
F19     T4      prepared pan    A7      Spread
F20     A7      Spread  A8      Bake
F21     A0      Preheat T5      preheated oven
F22     T5      preheated oven  A8      Bake
F23     I22     3 tablespoons softened butter   A9      Combine
F24     I23     3 tablespoons cocoa     A9      Combine
F25     I24     honey   A9      Combine
F26     I26     1 teaspoon vanilla extract      A9      Combine
```

Figure 4.4: A sample link instruction file in the transformed format.

Link File Format

Each link file is simply a list of link tags, with the ID and raw text information of the two extent tags associated with each link tag. An example is shown in Figure 4.4.

Experiment on Recipe Entity Extraction from Ingredient Sections

Introduction

Entity extraction from ingredient sections is the task of giving `Quantity`, `Unit` or `Ingredient` tags to words or phrases in each specification of an ingredient in the ingredient section of an recipe. For example, given the line in the ingredient section:

1 1/4 cups water

We would like to create a model that tags the sentence as:

[$_{\text{Quantity}}$ 1 1/4][$_{\text{Unit}}$ cups][$_{\text{Ingredient}}$ water]

Approach

Since this task involves giving labels to sequences of words in a document, it can be modeled as a NER task. Following the standard approach to NER [10], I treated it as a sequence labeling problem, and used Conditional Random Field (CRF) [12] as the classification algorithm. The toolkit CRFsuite [16] was used for this purpose.

In order to train the learning algorithm, the data needed to be converted to the IOB style encoding. Each token is associated with an I, O or B tag, affixed with the type of entity the token belongs to. For example, if the token "chicken" is *inside* an ingredient entity, then the correct IOB tag for this token should be "I-ING".

Each IOB-tagged recipe ingredient document is run through a feature extraction script, the extracted features are then fed to CRFsuite for training.

Baseline

The text for each ingredient listing in the ingredient section is very structured and predictable, therefore, it is useful to set a baseline in order to see how much feature engineering could improve the algorithm's performance. For this task, I chose the baseline of this experiment as the evaluation result of the same CRF algorithm trained with only the surface text of each token as the feature of the token.

Feature Extraction

The following categories of features were used for training the NER algorithm:

- **Word Features**: the raw text of the current token, the text of the previous token, combination of the text of the current and the previous tokens, etc.
- **POS Features**: the Penn Treebank POS tag of the current token, the Google Universal POS tag of the current token, POS tags of the previous tokens, etc.

- **Word Shape Features**: whether the current token is a number, whether the previous token is a number, etc.
- **Domain Knowledge Features**: whether the current token exists in a dictionary of food items created from the dataset used in [1].

Experiment on Recipe Entity Extraction from Instruction Sections

Introduction

Entity extraction from an instruction section is the task of giving `Action`, `Ingredient` and `Tool` tags to sequences of text in the instruction section of a food recipe. For example, the instruction:

> In a large bowl, dissolve the sugar in warm water, and then stir in yeast.

should be given the following tags:

> In [$_{\text{Tool}}$ a large bowl], [$_{\text{Action}}$ dissolve] [$_{\text{Ingredient}}$ the sugar] in [$_{\text{Ingredient}}$ warm water], and then [$_{\text{Action}}$ stir] in [$_{\text{Ingredient}}$ yeast]

Approach

Similar to the approach used for entity extraction from the ingredients section, this experiment is also modeled as a sequence labeling task, but with `Action`, `Tool` and `Ingredient` labels instead. Therefore, CRFsuite is also used for this task. The dataset is transformed to add IOB style tags, then run through feature extraction and training.

Baseline

The baseline for this task is defined as the evaluation result of the algorithm trained with the surface text of each token as the feature of that token. The result will be presented in Section 5.3.

Feature Extraction

The following features have been considered for the training of a model for this task:

- **Word Features**: the token's text, and surrounding tokens' text are treated as features, similar to the previous experiment.
- **POS Features**: the token's POS tags, and surrounding token's POS tags, are treated as features, similar to the previous experiment.
- **Dependency Parse Features**: in Section 4.1.3, I introduced the format of the preprocessed data from annotated recipe instruction sections. It includes the dependency parse information for each sentence in the instruction sections. Such dependency information was treated as features for the training of the algorithm.
- **Domain Knowledge Features**: similar to the previous experiment, the existence of the token in a dictionary of food item words is treated as a feature.
- **Features Utilizing the Ingredient Section**: whether the token exists in a set of tokens created from the text sequences tagged as `Ingredient` in the ingredient section of the current recipe.

Experiment on Relation Extraction between Ingredient Section and Instruction Section

Introduction

The task of finding relations between entities in the ingredients section and entities in the instructions section is the task of finding the co-reference relationships between an ingredient listed in the instructions section and the first mention of the ingredient before any cooking action has taken

place on it. Linking these entities correctly would enable the recipe text processing system to understand in which cooking steps each ingredient is used in.

Not every mention of an ingredient in the instructions section should be linked to an ingredient in the ingredients lists, however. Since both the first mention of an ingredient and an ingredient that has been modified by an action in the instructions are marked as `Ingredient`, the recipe text processing system should be able to distinguish these two types of entities.

One thing to note is that the experiment is based on the assumption that all food ingredients involved in the recipe are listed in the ingredients section of the recipe, which means that any ingredient that is in a `start_state` (See Section 3.2.3) in the instructions should have a corresponding ingredient in the ingredient list.

Approach

Two approaches for this task had been considered initially: in one approach, we iterate over every ingredient in the ingredients list, and try to find the most likely ingredient entity in the instructions section to associate it with, using a ranking algorithm (inspired by the study done by Jermsurawong et al. [9]); in the second approach, we iterate over every ingredient entity that is tagged to have the `start_state` attribute set to `Yes`, and associate it with the most likely ingredient in the ingredient list.

Neither approach seemed to be appropriate for this task.

With the first approach, we wouldn't be able to account for the case that one entry in the ingredient list can co-reference with multiple ingredient entities in the instructions section. For example, the ingredient entry:

> 2 tablespoons vegetable oil

could have co-reference relations with both the entity in the instruction:

> Mix 1 tablespoon oil with warm water.

and then later, co-reference with another entity in the instruction:

Heat the rest of the oil in a cast iron skillet.

We cannot account for such cases by selecting the most likely entity to associate the ingredient with.

The second approach would require the ingredient entities in the instructions section to have their `start_state` attributes identified, which itself seems to not be a trivial task, and we would run into a chicken-and-egg type of problem trying to identify whether an ingredient entity is in a start state before linking the ingredients in the ingredient list with ingredients in the instructions.

The approach taken eventually for this task is a modified version of the first approach above. We iterate over each ingredient in the ingredient list, and find *all* the ingredient entities that this ingredient can be linked to, using a SVM (Support Vector Machine [5]) classifier.

During training, the correct co-reference links are used as positive examples, and the negative examples are the pairs of each ingredient entity in the ingredient list and all the ingredient entities in the instructions that it does not have co-reference relations with.

In testing, each ingredient in the ingredient list is iterated over, and for each ingredient, all ingredient entities in the instructions are looped over to check if it has co-reference relationship with this ingredient.

Baseline

The baseline for this experiment is decided to be the result of using a SVM classifier trained with only the edit distance between the text of each pair of entities to be the features.

Feature Extraction

Below is a list of categories of features used in this experiment.

- **String Distance**: the edit distance between each pair of entities, the edit distance between the stemmed text of each pair of entities.

- **Lemma Matches**: the stemmed tokens between each pair that are the same, the POS tags of these tokens, the dependency information of these tokens, etc.
- **Word Sense Similarity**: similarity of the words in the two entities in the WordNet dataset and the similarity of the vectors for the two words.
- **Corresponding Action Features**: the surface text of the nearest action from the entity to be examined, the `act_type` (See Section 3.2.3) of these actions
- **Relative Distance**: the relative distance in the ingredient list and the relative distance in the instructions for the two entities.

Experiment on Relation Extraction within Instruction Section

Introduction

In order to generate a graph structure for a food recipe, we need to recognize the vertices and arcs of the graph. The vertices correspond to the `Ingredient`, `Action` and `Tool` entities in the recipe. The identification of these entities has been addressed in Section 4.3. In this section, the task of finding the arcs will be described, which is the task of identifying relations between each entity in the instruction section.

In the annotated dataset, two categories of link tags were given to pairs of entities within the instructions section—`Flow` tags and `Coref` tags. The goal of this task then is to construct a model that is able to automatically recognize such relations between entities.

This experiment mostly focuses on the identification of the relations equivalent to the `Flow` relations in the corpus, due to the sparsity of `Coref` relationships within the instructions sections in the dataset.

The task of Relation Extraction in the instructions section can be divided into 3 sub-tasks:

(1) At the beginning, each `Action` entity in the instructions is loaded. For each action, link it with

Ingredient and Tool entities that this action "modifies". The resulting links correspond to Flow tags that connect from Ingredient or Tool entities to Action entities.

(2) For each action, identify Ingredient and/or Tool entities in other sentences to connect this action with. This could be thought of as identifying the ingredients or tools that the action "produces". Equivalent to identifying the outgoing edges from Action entities to Ingredient or Tool entities in the annotated corpus.

(3) For each action, find other following actions to link it to. This is equivalent to finding out Flow edges that connect two Action entities in the original corpus.

For the first task, it is possible to achieve high performance by using a simple rule-based approach, without training a machine learning algorithm. This is because the majority of Ingredient and Tool entities connect to the closest preceding Action entity in the same sentence. In Section 5.5, the performance of a rule-based decision function that links an ingredient or tool entity to its most adjacent action entity is reported. A statistical approach has not been tested for this task.

Meanwhile, based on my analysis on the dataset, 97% of Flow links that connect from an Action tag to another Action tag are connecting two immediately adjacent cooking actions (where there is no other cooking actions in-between). Therefore, the sub-task (3) of identifying relations between Action entities can be seen as a task of identifying whether an Action that does not "produce" any Ingredient or Tool entities (there is no directed edges connecting from this action to an ingredient or tool entity) should be connected to the immediately following action. And if not, classify which other following actions this action should be connected to. Due to time constraints and the sparsity of pairs of actions that are not immediately adjacent in the annotated dataset (Only 3% of action pairs, which also includes incorrectly annotated Flow link tags and ambiguities that the annotation scheme doesn't cover, such as disjunction of actions), this experiment is left for future studies.

Therefore, this experiment mostly focuses on the extraction of relations from an Action entity to an Ingredient entity in a instructions section.

Approach

A binary classification approach is taken using a linear SVM classifier. In order to train and test the classifier, positive and negative samples need to be generated. The positive samples are pairs of `Action-Ingredient` entities that are connected by a `Flow` link tag, and for each ingredient in each positive sample, generate negative examples where each action between the positive action and ingredient entities is paired with the same ingredient entity, and then select pairs where there is no `Flow` links connecting. Only pairs where the ingredient is not in a start state are considered. In a recipe text processing system, whether an ingredient is in a start state can be determined following the procedures described in Section 4.4

Baseline

The baseline of the experiment is determined to be the performance of a simple decision function on the test dataset. The decision function returns true if the action entity and the ingredient entity are one action entity apart—there is exactly one action entity between the pair of entities in the sample data. The intuition behind this is that it is common for recipes to have sentences such as:

> Place the pork tenderloin in a slow cooker; pour the root beer over the meat.

where there should be a `Flow` link connecting the action place and the ingredient the meat, and there is another action pour between the two connected entities.

Feature Extraction

Several Categories of features have been considered for this experiment:

- **Baseline Decision**: the prediction from the baseline rule-based function
- **Sentence Distance**: the number of sentences between the pair of entities.
- **Number of Actions In-between**: the number of `Action` entities between the two entities.

- **Matching Ingredients Connected to Action**: string matches between ingredient names connected to the action and the ingredient in the sample pair.
- **Shared Lemmas**: whether the lemmatized action and ingredient share any tokens.

Chapter 5

Experiment Evaluation and Conclusion

Evaluation Metrics

The dataset of 100 recipes is randomly divided into Train, Test, and DevTest sets, by a ratio of 8:1:1. The Train set is used for training the machine learning algorithms in each experiment, the Test set is for evalutating the performance of each algorithm on the tasks, and the DevTest set is for evaluating and tuning potential feature functions.

The entity recognition tasks described in Section 4.2 and 4.3 use the IOB-style tags. Therefore I calculated each algorithm's precision, recall, accuracy and F1 score for each tag, in each respective task.

For the relation extraction tasks in Section 4.4 and 4.5, precision, recall, accuracy and F1 score have also been calculated. Since these experiments were framed as binary classification tasks, only one value of each metric is calculated, compared to the multiple values in the multi-class classification tasks above.

During evaluation, each task's best performance will be compared to its baseline.

Class	Precision	Recall	Accuracy	F-1 Score
B-Quantity	0.99	1.00	1.00	1.00
I-Quantity	1.00	0.86	1.00	0.92
B-Unit	0.92	0.98	0.98	0.95
I-Unit	1.00	0.96	1.00	0.98
B-Ingredient	0.89	0.88	0.96	0.88
I-Ingredient	0.84	0.97	0.92	0.90
O	1.00	0.19	0.94	0.32
Micro-Averaged Scores	0.90	0.90	0.97	0.90

Table 5.1: The baseline scores for the entity recognition task on the ingredients sections

Class	Precision	Recall	Accuracy	F-1 Score
B-Quantity	0.98	1.00	1.00	0.99
I-Quantity	0.92	0.86	1.00	0.89
B-Unit	0.96	0.99	0.99	0.97
I-Unit	1.00	1.00	1.00	1.00
B-Ingredient	0.95	0.96	0.98	0.95
I-Ingredient	0.93	0.99	0.97	0.96
O	1.00	0.57	0.97	0.73
Micro-Averaged Scores	0.95	0.95	0.99	0.95

Table 5.2: Experiment scores for the entity recognition task on the ingredients sections

Evaluation of Entity Extraction on Ingredient Lists

Performance

The baseline performance is shown in Table 5.1. The text in the ingredients sections is very structured, so it is not surprising that the sequential classification algorithm performs well with only a limited set of features.

The best performance I was able to achieve in this experiment is shown in Table 5.2.

The baseline appears to have very low recall on the O tag, the added features were able to improve the recall score from 0.19 to 0.57, but it's still low compared to the scores on other classes.

Class	Precision	Recall	Accuracy	F-1 Score
B-Action	0.96	0.72	0.96	0.82
B-Tool	0.86	0.71	0.98	0.78
I-Tool	0.74	0.83	0.97	0.78
B-Ingredients	0.85	0.59	0.94	0.70
I-Ingredients	0.87	0.63	0.95	0.73
O	0.81	0.96	0.86	0.88
Micro-Averaged Scores	0.83	0.83	0.95	0.83

Table 5.3: The baseline scores for the entity recognition task on the instructions sections

Analysis

The sentences' structures in the ingredients section are mostly very predictable: each line usually begins with a Quantity entity, oftentimes followed by an Unit entity, then an Ingredient entity. Therefore it is expected that these tags have relatively high performance with a sequential classification algorithm.

The O tags are not very commonly seen in a ingredient list. In fact, only 0.06% of the tokens in the dataset are O tags. This could be the cause of the poor performance of the algorithm on the O class.

Evaluation of Entity Extraction on Instructions

Performance

The instructions sections are not as structured as the ingredient lists, therefore the overall performance is not as high as the ones on the ingredient lists. The baseline performance is shown in Table 5.3 and the best performance achieved in this experiment is shown in Table 5.4.

Analysis

It appears that the POS tags are very important features in this task. They noticeably improved the test performance in this experiment. Yet, it seems that several errors were caused by incorrect POS

Class	Precision	Recall	Accuracy	F-1 Score
B-Action	1.0	0.91	0.99	0.95
B-Tool	0.81	0.86	0.98	0.83
I-Tool	0.81	0.92	0.98	0.86
B-Ingredients	0.89	0.90	0.97	0.89
I-Ingredients	0.86	0.73	0.96	0.79
O	0.94	0.96	0.95	0.95
Micro-Averaged Scores	0.92	0.92	0.97	0.92

Table 5.4: The highest scores achieved for the entity recognition task on the instructions sections

tagging by spaCy. For example, a common error is a verb tagged as noun, which sometimes causes the algorithm to not register it as an Action, or tag it as an ingredient or tool entity.

Another prevalent error is that the algorithm would classify an ingredient as a tool, or a tool as an ingredient. This is likely to be caused by the fact that both tools and ingredients are nouns, but the algorithm isn't able to distinguish them very easily. The feature of checking whether a term is contained in a collection of food item names has been used in the experiment, but it seems that this feature is not enough for the algorithm to distinguish these two classes.

Evaluation of Relation Extraction between Ingredients Section and Instructions Section

Performance

This is the task of linking each ingredient in the ingredient list with mentions of the ingredient in the instructions where it is first being introduced. The positive examples are the correct linkings and the negative examples are all other links with each ingredient entity in the instructions for each ingredient in the ingredient list. The results of the baseline and best performance achieved in the experiment is shown in Table 5.5

Experiment	Precision	Recall	Accuracy	F-1 Score
Baseline	0.74	0.83	0.97	0.78
Improved	0.88	0.76	0.98	0.81

Table 5.5: The baseline and the best result achieved through feature engineering for the experiment of linking ingredients in the ingredients section with ingredient mentions in the instructions section.

Analysis

The best result achieved in this experiment did not show significant improvements on precision, accuracy or F-1 score over the baseline of using only the edit distance between the last noun tokens in each of the two entities as feature. In fact, the recall dropped compared to the baseline, which indicates that the improvement on precision may be partially caused by an overall decrease in positive decisions from the algorithm.

Several notable errors have appeared during this experiment.

One type of error is that the algorithm does not classify as *True* the existence of a link between an ingredient and a reference to this ingredient in the instructions using its ontological class name. For example, when the ingredients are

1/2 teaspoon dried oregano

1/2 teaspoon chopped parsley

and the instruction is

mix together the herbs

The algorithm usually would not recognize the links between dried oregano—the herbs, and chopped parsley—the herbs. I tried to address this problem using word sense similarity features, by calculating the WordNet similarity scores and word vector similarity scores between the noun tokens in each pair. However, the test results show that these features do not significantly reduce such errors.

Another type of errors happened when there are ingredients that have some shared tokens in their names. For example, the algorithm sometimes would incorrectly link “green bell pepper” with ”red bell pepper”, and “red bell pepper” with “black pepper”. I attempted to address this

Experiment	Precision	Recall	Accuracy	F-1 Score
Rule-based Linking	0.94	0.98	0.95	0.96

Table 5.6: The performance of a rule-based decision function for ingredient-action linking over the test dataset.

issue by using the frequency sum of each token shared between a pair of entities. The frequency of a token is calculated as

$$Freq(t) = \frac{1}{|T|}$$

where T is all the occurrences of the token in the instructions section. The intuition behind the frequency sum is that the more frequent a token is seen elsewhere in a recipe, the less likely that it is a good indicator for ingredient linking, therefore it should contribute less to the similarity score between two entities. This feature was able to reduce the number of such errors, but not completely eliminate them.

Finally, several errors were caused by incorrect annotation where `Coref` links were not added for some pairs of entities.

Simple Rule-based Approach to Ingredient-Action Linking in the Instructions Section

Performance

Described in Section 4.5.1, the test result of the simple rule-based approach to linking ingredient and tool entities to action entities is shown in Table 5.6.

Analysis

The performance of the rule-based decision function confirms the assumption that most ingredient entities in an instructions section, within the same sentence, connect to the most adjacent preceding action entity.

One type of errors appeared seem to be caused by the disjunction of actions in a sentence. For example,

Pour or scoop the batter onto the griddle

The rule-based function would connect the batter and the griddle to the action scoop, but not pour, while they are supposed to be linked to both actions in case of a disjunction according to the annotation.

Action conjunction structures can also cause trouble for the decision function. For example, one of the incorrectly linked sentence is:

Peel and chop eggs

the algorithm would link eggs to chop, while the correct linking in cases of action conjunction according to the annotation should be assuming the actions happen sequentially. Therefore the correct linking should be:

eggs→peel→chop

Finally, several false positive errors were due to incorrect annotation where some `Flow` link tags that were supposed to be added were not included in the annotated recipes.

Experiment on Action-Ingredient Linking

Performance

The performance of the rule-based baseline approach is compared to a statistical approach using binary classification in Table 5.7.

The F-1 scores show that the statistical approach did not significantly improve over the rule-based baseline.

Experiment	Precision	Recall	Accuracy	F-1 Score
Rule-based Baseline	0.57	0.47	0.70	0.52
Statistical Approach	0.58	0.50	0.70	0.53

Table 5.7: The performance of the statistical, binary classification approach compared to the rule-based baseline approach on action-ingredient linking

Analysis

The statistical algorithm treats word matches between the action's linked ingredients and the other ingredient entity in a sample as a feature. For example, for the sample sentence below:

> Chop eggs, transfer chopped eggs to a plate.

eggs is connected to chop through ingredient-action linking. When the algorithm is trying to determine if there is a link between chop and chopped eggs, the token matching between eggs and chopped eggs is considered a feature, since eggs connects to the action chop.

One problem encountered for this approach is the fact that not all actions have ingredients linked to it, and some ingredients are linked to an action further up in a cooking action chain. For example:

> Cook pasta in boiling water. Drain and set aside.
>
> ...
>
> Combine cooked pasta and sauce.

the correct linking should be between set and cooked pasta. However, due to the fact that no matching ingredient is connected to set (the pasta entity is connected to cook further up the action chain), the feature function cannot match any tokens between set and cooked pasta, causing the algorithm to produce a false negative prediction.

The feature of matching lemmatized action token with lemmatized ingredient tokens appears to be important in this algorithm. The intuition behind it is that an ingredient may contain a past-participle of the action that links to it. For example, in the sentence

> Cook pasta

Cook may be linked to an ingredient cooked pasta in an instruction later.

Conclusion and Future Work

In this thesis, I first proposed and presented an annotation framework for English recipe text processing for the purpose of generating graph structures from food recipes. Then I annotated and created a dataset of food recipes annotated under the proposed annotation framework. Then, I performed a series of experiments using the annotated data on tasks ranging from extracting entities from recipes to extracting the relations between such entities, and reported the test results of these experiments.

One of the tasks that could be tackled in future work is to expand the size of the annotated dataset, which would allow us to perform experiments using categories of data whose sizes were previously too small.

Furthermore, the annotation framework could also be improved, addressing issues reported in Section 3.4.

In addition, the experiments, especially the experiment on action-ingredient linking, presented in Section 5.6 could also see more improvements through better feature engineering.

Finally, a complete system that extracts graph structures from recipes should also be created and tests, based on the experiments presented in this study and future studies.

Bibliography

[1] Yong-Yeol Ahn, Sebastian E. Ahnert, James P. Bagrow, and Albert-Lszl Barabsi. Flavor network and the principles of food pairing. *Scientific Reports*, 1:196, 12 2011.

[2] Mario Bollini, Stefanie Tellex, Tyler Thompson, Nicholas Roy, and Daniela Rus. Interpreting and executing recipes with a cooking robot. In *Experimental Robotics*, pages 481–495. Springer, 2013.

[3] Minsuk Chang, Vivian M. Hare, Juho Kim, and Maneesh Agrawala. Recipescape: Mining and analyzing diverse processes in cooking recipes. In *Proceedings of the 2017 CHI Conference Extended Abstracts on Human Factors in Computing Systems*, CHI EA '17, pages 1524–1531, New York, NY, USA, 2017. ACM.

[4] Jinho D Choi, Joel R Tetreault, and Amanda Stent. It depends: Dependency parser comparison using a web-based evaluation tool. In *ACL (1)*, pages 387–396, 2015.

[5] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, Sep 1995.

[6] Reiko Hamada, Ichiro Ide, Shuichi Sakai, and Hidehiko Tanaka. Structural analysis of cooking preparation steps in japanese. In *Proceedings of the Fifth International Workshop on on Information Retrieval with Asian Languages*, IRAL '00, pages 157–164, New York, NY, USA, 2000. ACM.

[7] Tadayoshi Hara, Takuya Matsuzaki, Yusuke Miyao, and Jun'ichi Tsujii. Exploring difficulties in parsing imperatives and questions. In *IJCNLP*, pages 749–757, 2011.

[8] Matthew Honnibal, Mark Johnson, et al. An improved non-monotonic transition system for dependency parsing. In *EMNLP*, pages 1373–1378, 2015.

[9] Jermsak Jermsurawong and Nizar Habash. Predicting the structure of cooking recipes. In *EMNLP*, 2015.

[10] D. Jurafsky and J.H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Prentice Hall Series in Artifi. Pearson Prentice Hall, 2009.

[11] Chloé Kiddon, Ganesa Thandavam Ponnuraj, Luke Zettlemoyer, and Yejin Choi. Mise en place: Unsupervised interpretation of instructional recipes.

[12] John Lafferty, Andrew McCallum, and Fernando CN Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. 2001.

[13] Hirokuni Maeta, Tetsuro Sasada, and Shinsuke Mori. A framework for procedural text understanding. In *Proceedings of the 14th International Conference on Parsing Technologies*, pages 50–60, 2015.

[14] Shinsuke Mori, Hirokuni Maeta, Yoko Yamakata, and Tetsuro Sasada. Flow graph corpus from recipe texts. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Hrafn Loftsson, Bente Maegaard, Joseph Mariani, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, Reykjavik, Iceland, may 2014. European Language Resources Association (ELRA).

[15] Shinsuke Mori, Tetsuro Sasada, Yoko Yamakata, and Koichiro Yoshino. A machine learning approach to recipe text processing. In *Proceedings of the 1st Cooking with Computer Workshop*, pages 29–34, 2012.

[16] Naoaki Okazaki. Crfsuite: a fast implementation of conditional random fields (crfs), 2007.

[17] James Pustejovsky and Amber Stubbs. *Natural Language Annotation for Machine Learning: A guide to corpus-building for applications*. " O'Reilly Media, Inc.", 2012.

[18] Kyeongmin Rim. Mae2: Portable annotation tool for general natural language use. In *Proceedings of the 12th Joint ACL-ISO Workshop on Interoperable Semantic Annotation*, pages 75–80, 2016.

[19] P. Schumacher and M. Minor. Extracting control-flow from text. In *Proceedings of the 2014 IEEE 15th International Conference on Information Reuse and Integration (IEEE IRI 2014)*, pages 203–210, Aug 2014.

[20] Dan Tasse and Noah A Smith. Sour cream: Toward semantic processing of recipes, 2008. *Report of preliminary work*.

[21] Yoko Yamakata, Shinji Imahori, Yuichi Sugiyama, Shinsuke Mori, and Katsumi Tanaka. *Feature Extraction and Summarization of Recipes Using Flow Graph*, pages 241–254. Springer International Publishing, Cham, 2013.

[22] Yoko Yamakata, Hirokuni Maeta, Takuya Kadowaki, Tetsuro Sasada, Shinji Imahori, and Shinsuke Mori. Cooking recipe search by pairs of ingredient and action word sequence v.s. flow-graph representation. *ł*, 32(1):WII–F_1–9, 2017.

[23] Ziqi Zhang, Philip Webster, Victoria S. Uren, Andrea Varga, and Fabio Ciravegna. Automatically extracting procedural knowledge from instructional texts using natural language processing. In *LREC*, 2012.

[24] Ziqi Zhang, Philip Webster, Victoria S Uren, Andrea Varga, and Fabio Ciravegna. Automatically extracting procedural knowledge from instructional texts using natural language processing. In *LREC*, pages 520–527, 2012.