We present an annotated corpus of English cooking recipe procedures, and describe and evaluate computational methods for learning these annotations. The corpus consists of 300 recipes written by members of the public, which we have annotated with domain-specific linguistic and semantic structure. Each recipe is annotated with (1) ‘recipe named entities’ (r-NEs) specific to the recipe domain, and (2) a flow graph representing in detail the sequencing of steps, and interactions between cooking tools, food ingredients and the products of intermediate steps. These two kinds of annotations, inter-annotator agreement ranges from 82.3 to 90.5 F1, indicating that our annotation scheme is appropriate and consistent. We experiment with producing these annotations automatically. For r-NE tagging we train a deep neural network NER tool; to compute flow graphs we train a dependency-style parsing procedure which we apply to the entire sequence of r-NEs in a recipe. In evaluations, our systems achieve 71.1 to 87.5 F1, demonstrating that our annotation scheme is learnable.

**Keywords:** cooking recipe corpus, English recipes, recipe named entity, recipe flow graph, procedural text annotation

## 1. Introduction

Procedural text describes in an objective way how to carry out some task. Although this kind of text is widespread, it is found particularly in instruction manuals and cookbooks. The information in procedural text could be used to support a wide range of computer-based applications involving intelligent search, reasoning, and non-textual forms of human-computer interaction; but first the text would need to be transformed into a more computationally tractable representation. At minimum, this representation would represent domain entities, actions, and relationships between them.

A flow graph is one kind of computationally tractable representation: it contains steps linking actions with entities, and specifies the obligatory sequencing relationships between steps. A flow graph abstracts away from differences in surface linguistic expression that are irrelevant to correctly performing a procedure. For example, when cooking a dish, the instructions *add diced carrots to the pot* and *dice carrots and add to the pot* are different textually, but would correspond to the same flow graph since they involve the same actions carried out in the same order.

Our research focuses on the cooking domain, and specifically the procedural part of cooking recipes. Our general approach and some of our computational techniques are inspired by recent research on recipes written in Japanese (Mori et al., 2014). We extend that work by adapting the annotation to English, annotating more data, investigating the consistency of annotation, and improving accuracy of processing.

In this paper, we describe how we annotated a corpus of 300 English recipes, which were contributed to a recipes web site by members of the public. We first annotated each recipe with ‘recipe named entities’ (or r-NEs). These r-NEs are specific to the recipe domain, and account for cooking actions, cooking implements (tools), ingredients, intermediate products, durations and quantities. The r-NE annotation formed the basis for a second level of annotation: a flow graph representing in detail the relationships between the r-NEs (e.g. an action carried out with a certain tool and food items) and the sequencing of the actions.

We used the annotated corpus to investigate approaches for automatic r-NE tagging and flow graph computation. For r-NE tagging, we train a deep neural network NER tool, treating the task as a sequence labelling problem. To derive the flow graph representation, we consider the entire sequence of r-NEs in a recipe as a single input string, and extract a flow graph using a dependency-style maximal spanning tree parser. In contrast to standard sentence-by-sentence parsing, the parser creates flow graph edges that span different sentences in exactly the same way as within-sentence edges; this means we do not need any extra processing stages such as co-reference resolution. Our evaluation results are very promising.

The rest of the paper is structured as follows. Section 2 surveys approaches to representing procedural text, and cooking recipe text annotation and processing. Section 3 describes the r-NE tagset, annotation, and an evaluation of automatic r-NE tagging accuracy. Section 4 describes the flow graph representation, annotation, the parsing algorithm for computing a flow graph, and an evaluation of its accuracy. Section 5 discusses the main findings and proposes directions for future research.

---

2. Related Work

There has been much research into representing the semantics of natural language sentences (Banerwch et al., 2013, *inter alia*) and developing parsers that output such representations (Flanigan et al., 2014, for example). Semantic representations encode the meanings of linguistic units within a sentence, but do not attempt to capture domain-specific constraints or real-world context. In some domains and applications these latter factors may be very important. (For example, in cooking, a *mixture* can sometimes be *beaten* and sometimes not, depending on which ingredients have been added to it.) For this reason, approaches to processing text describing procedures often do not attempt to construct general semantic representations, but instead construct more specialised domain- and genre-specific representations. For example, Momouchi (1980) proposed representing the meaning of procedural text as a flow graph, and described algorithms to convert text documents to flow graphs. Hamada et al. (2000) adopted a similar graph representation for analysing recipes.

More recent studies have created annotated corpora of procedural text, in order to build machine learning-based systems that extract entity and action information. In the domain of cooking recipes, such information has been shown to be useful for the so-called ‘smart kitchen’ (Hashimoto et al., 2008), cooking robots (Bollini et al., 2013), etc. In one recent study, Mori et al. (2014) produced flow graph annotations for 266 cooking recipes written in Japanese. That corpus has been used for testing empirical methods for natural language understanding (Maeta et al., 2015) and intelligent search (Yamakata et al., 2013). Similarly, Jermsurawong and Habash (2014) described a corpus of ingredient trees for recipes, and evaluated methods for constructing these automatically. There have been some attempts at unsupervised processing of recipes (Kiddon et al., 2015, *inter alia*) but the outputs are much less rich than those obtained by supervised methods.

This paper concerns a flow graph corpus for recipes in English; the flow graphs are based on the proposals of Mori et al. (2014) in which the graph nodes are ‘recipe named entities’ (r-NEs). The r-NE types cover domain entities such as ingredients; in addition, Mori et al. argue that r-NE types should also cover actions by the cook / chef (which are often expressed as verbs).

Unlike the set of NE types commonly used for general text (Sang and Meulder, 2003)—person name, location, organization, etc.—r-NEs are an example of a domain-specific NE definition. Domain-specific NE definitions are widely used in fields such as bioinformatics (Ben Abacha and Zweigenbaum, 2011). A domain-specific NE definition facilitates the development of intelligent applications that process documents in that particular domain.

The task of identifying NEs is called ‘named entity recognition’ (NER). NER is usually formulated as a sequence labeling problem, in which each input token is tagged as being inside or outside an NE (and in some approaches, beginning an NE). In this framework many techniques and refinements have been investigated (Borthwick, 1999; Sang and Meulder, 2003; Ratinov and Roth, 2009), and many general-purpose NER tools have been developed. In this study, we use one such tool, PWNER (Sasada et al., 2015a), which computes probabilities for all possible NE tags based on pointwise prediction, and searches for the best sequence of tags under the tag sequence constraints. The tool is distinctive in being trainable on data that has been only partially annotated.

3. Recipe Named Entities

3.1. Recipe Named Entity Tags

In previous work (Yamakata et al., 2017), we created a corpus of 100 recipes written in English, sampled from the Allrecipes UK/Ireland web site and annotated for ‘recipe named entities’ (r-NEs). The r-NE tag types were based on a set of eight tags originally devised for annotating recipes written in Japanese (Mori et al., 2014), but with the addition of a further two tags. One is introduced in order to account for linguistic phenomena that occur in English and many other languages but not Japanese: discontinuous multi-word expressions (Constant et al., 2017). The other is to cover expressions for the actions by automatic cooking tools (e.g. food processors), which are not yet common in Japan, so the tag was not necessary for recipes in Japanese. Table 1 lists the ten r-NE tags that we use for English recipe annotation. Note that this tag set is compatible with the original.

![Table 1: Recipe named entity (r-NE) tags.](http://allrecipes.co.uk/)

Figure 1 shows the annotation that would be given to the recipe step *Preheat oven to 180 C / Gas mark 4*. The tag suffixes -B and -I (abbreviating Begin and Inside respectively) indicate NE spans according to the IOB2 text chunking representation (Sang and Veenstra, 1999). A tag is assigned to each word or word-sequence designating a single and indivisible object/action/phenomenon in the cooking domain. For example, *Gas mark 6* in the example sentence designates the state of the dial of the oven being at 6, so it is annotated as a single r-NE. Therefore, the first word *Gas* is annotated as St-B and the subsequent words *mark* and 6 are both annotated as St-I. The word *to* is annotated O because it is Outside any named entity.
3.2. Annotating Recipe Named Entities

Our corpus of English recipes consists of the 100 recipes mentioned above, plus a further 200, which we have selected and annotated recently.

The first, 100-recipe subset (which we call ‘100-r’) was sampled from Allrecipes, selecting the most popular recipes within each dish category (e.g. ‘Main course’, ‘Dessert’) in proportion to the total number of recipes in that category. These recipes were annotated by an English native speaker, and all annotations were verified by the first author, as described by Yamakata et al. (2017).

The second, 200-recipe subset (‘200-r’) was sampled from Allrecipes completely randomly. These recipes were annotated by two Japanese homemakers who had previously annotated Japanese recipes in the study of Mori et al. (2014). Both are good English readers; in preparation, they re-familiarized themselves with the annotation guidelines. They annotated the first half of 200-r completely manually; the average annotation time was 19 minutes per recipe (including initial preparation). To speed up the workflow, we automatically tagged the remainder of 200-r using the PWNER NE recognizer trained on 100-r; the annotators then checked and corrected the recognition results. The 300-recipe corpus comprises 38,062 tokens in total.

To estimate inter-annotator agreement, we asked the second annotator to re-annotate 100-r independently. Unfortunately we cannot use Cohen’s kappa or related measures, since they assume that each annotation decision is independent of any other; however, with IOB2 some combinations of tags are illegal, e.g. an Inside tag may only occur after a Begin or Inside with the same tag prefix. Instead, we took one of the annotators as the gold standard and evaluated the accuracy of the other with respect to this. We measured the agreement as 89.9% precision, 92.2% recall, and 90.5 F1. We consider this to be a high level of agreement.

3.3. Automatic Recognition of r-NEs

Having created the 300-recipe r-NE annotated English corpus, we investigated the extent to which the annotation supports NER, and whether the r-NE tagging accuracy is comparable to previous results for Japanese recipes.

Although PWNER works well for r-NE tagging, we decided instead to experiment with the more modern deep learning named entity recognizer BERT-NER, a state-of-the-art tool based on the BERT neural network architecture (Devlin et al., 2019). For this evaluation we used 10-fold cross-validation, splitting the data into 80% training, 10% hyperparameter tuning, and 10% test. The overall F1 for r-NE recognition is 87.6. Accuracies by recipe subset are shown in Table 2. Table 3 shows accuracy by tag; F1 ranges from a low of 17.1 for the rarest tag At to 92.7 for the very common tag Ac.

In previous research, Sasada et al. (2015b) trained PWNER on a corpus of 193 Japanese recipes of comparable complexity annotated with the same set of r-NE tags, and reported 84.4 F1; our F1 for 200-r is comparable. We also note that F1 is 2.3 points higher for 200-r than 100-r; this indicates that annotating more data is likely to lead to further worthwhile improvements in NE recognition accuracy. It is likely that annotating more data would be particularly beneficial for the tags Ac2, At and At, which are relatively rare.

### Table 2: r-NE recognition accuracy by corpus subset and overall.

<table>
<thead>
<tr>
<th>Tag</th>
<th>Precision</th>
<th>Recall</th>
<th>F1</th>
<th>#NEs</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>90.5%</td>
<td>93.9%</td>
<td>92.1</td>
<td>5007</td>
</tr>
<tr>
<td>T</td>
<td>87.7%</td>
<td>90.1%</td>
<td>88.9</td>
<td>1904</td>
</tr>
<tr>
<td>D</td>
<td>87.4%</td>
<td>90.2%</td>
<td>88.8</td>
<td>589</td>
</tr>
<tr>
<td>Q</td>
<td>70.9%</td>
<td>76.8%</td>
<td>73.7</td>
<td>529</td>
</tr>
<tr>
<td>Ac</td>
<td>92.3%</td>
<td>93.1%</td>
<td>92.7</td>
<td>4977</td>
</tr>
<tr>
<td>Ac2</td>
<td>43.8%</td>
<td>46.8%</td>
<td>45.3</td>
<td>178</td>
</tr>
<tr>
<td>At</td>
<td>51.4%</td>
<td>50.4%</td>
<td>50.9</td>
<td>255</td>
</tr>
<tr>
<td>At</td>
<td>60.0%</td>
<td>10.0%</td>
<td>17.1</td>
<td>15</td>
</tr>
<tr>
<td>St</td>
<td>66.1%</td>
<td>71.4%</td>
<td>68.6</td>
<td>1105</td>
</tr>
<tr>
<td>St</td>
<td>81.6%</td>
<td>82.5%</td>
<td>82.0</td>
<td>876</td>
</tr>
<tr>
<td>Total</td>
<td>86.5%</td>
<td>88.8%</td>
<td>87.6</td>
<td>15435</td>
</tr>
</tbody>
</table>

Table 2: r-NE recognition accuracy by tag for the full 300-recipe corpus.

### Table 3: r-NE recognition accuracy by tag for the full 300-recipe corpus.

<table>
<thead>
<tr>
<th>Tag</th>
<th>Precision</th>
<th>Recall</th>
<th>F1</th>
<th>#NEs</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>90.5%</td>
<td>93.9%</td>
<td>92.1</td>
<td>5007</td>
</tr>
<tr>
<td>T</td>
<td>87.7%</td>
<td>90.1%</td>
<td>88.9</td>
<td>1904</td>
</tr>
<tr>
<td>D</td>
<td>87.4%</td>
<td>90.2%</td>
<td>88.8</td>
<td>589</td>
</tr>
<tr>
<td>Q</td>
<td>70.9%</td>
<td>76.8%</td>
<td>73.7</td>
<td>529</td>
</tr>
<tr>
<td>Ac</td>
<td>92.3%</td>
<td>93.1%</td>
<td>92.7</td>
<td>4977</td>
</tr>
<tr>
<td>Ac2</td>
<td>43.8%</td>
<td>46.8%</td>
<td>45.3</td>
<td>178</td>
</tr>
<tr>
<td>At</td>
<td>51.4%</td>
<td>50.4%</td>
<td>50.9</td>
<td>255</td>
</tr>
<tr>
<td>At</td>
<td>60.0%</td>
<td>10.0%</td>
<td>17.1</td>
<td>15</td>
</tr>
<tr>
<td>St</td>
<td>66.1%</td>
<td>71.4%</td>
<td>68.6</td>
<td>1105</td>
</tr>
<tr>
<td>St</td>
<td>81.6%</td>
<td>82.5%</td>
<td>82.0</td>
<td>876</td>
</tr>
<tr>
<td>Total</td>
<td>86.5%</td>
<td>88.8%</td>
<td>87.6</td>
<td>15435</td>
</tr>
</tbody>
</table>

Table 3: r-NE recognition accuracy by tag for the full 300-recipe corpus.

### 4. Recipe Flow Graphs

#### 4.1. Flow Graph Representation

Mori et al. (2014), in a study of Japanese recipe text, showed how the procedural aspects of a recipe can be represented as a flow graph. A flow graph is a directed acyclic graph (DAG) with a single root. Graph nodes are r-NEs and labelled edges represent the relationships between these nodes. Table 4 lists the flow graph edge labels.

Figure 2 shows the procedural text in a recipe “Almost no fat banana bread” on the Allrecipes web site. Figure 3 contains the flow graph for this recipe. The flow graph makes explicit what actions are applied to which ingredients and using what tools. The graph is read downwards from the top; nodes without any incoming edges represent ingredients, and a node with no outgoing edges denotes an end product. A path of directed edges from an action A to an action B implies that A must be carried out before B. For example, the flow graph explicitly represents the fact that batter in sentence 4 is the end product of stirring the contents of the large bowl; this relationship is only implicit.

3https://github.com/kyzouhza/BERT-NER

in the recipe text since batter is not previously mentioned. Please note that a flow graph is an internal, symbolic representation designed to support any kind of computation over recipes, and we do not intend that it be displayed verbatim to a cook in a recipe visualization application, for instance.

Starting from the r-NE annotations, the same Japanese homemakers annotated all 300 sentences. The average annotation time was 37 minutes per recipe (including initial preparation); at first it was longer but as the annotators gained experience they got faster. The average annotation time for the final ten recipes was 26 minutes per recipe. To measure inter-annotator agreement, we arranged for both annotators to annotate the 100-r subset. We compute agreement similarly to the previous section, for similar reasons. Taking one of the annotators as the gold standard, the other’s annotations had 84.4% precision, 80.4% recall, and 82.3 F1. Although lower than the r-NE agreement, this is still a good result bearing in mind the very large space of possible edge annotations.

Maeta et al. (2015) describe a procedure for computing a flow graph from recipe text. The procedure consists of three steps, which are applied in turn to the entire r-NE tagged recipe, and output a set of directed edges between r-NEs:

1. Calculate weights of labelled edges between all pairs of r-NEs, using an SVM-based machine learning model.
2. Given the resulting weighted digraph, select a root node and compute a spanning arborescence of minimum weight. A reliable heuristic is to select as root node the Ac that occurs last in the input.
3. Since the previous step can only produce a tree, extend this to a DAG by adding further edges whose weights are below a threshold and which satisfy certain conditions to ensure consistency.

In our work, we apply a similar procedure but ignore labels in step 1 and omit step 3; we output a spanning arborescence with r-NEs as nodes connected by (directed) unlabelled edges.

We estimate the weight of a directed edge from node \( u \) to \( v \) with label \( l \) as follows:

\[
s(u, v, l) = \frac{\exp(\Theta \cdot f(u, v, l))}{\sum_{(x, r) \in (V \setminus \{u\}) \times L} \exp(\Theta \cdot f(u, x, r))}
\]

where \( V \) is a set of nodes, \( L \) is the set of edge labels (see Table 4), \( \Theta \) is a weight vector, and \( f(u, v, l) \) is a function returning the feature vector for the edge. Intuitively, the right hand side of the equation computes the total weight of the edge’s features as a proportion of the grand total of the weights of all edges starting at node \( u \).

Figure 2: The recipe “Almost no fat banana bread”, comprising 7 sentences (numbered 0 to 6), grouped into 3 steps. Sentences used as examples are in bold.

<table>
<thead>
<tr>
<th>Label</th>
<th>Meaning</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Agent</td>
<td>Subject</td>
<td>Relationship with actions (Ac or Af)</td>
</tr>
<tr>
<td>Targ</td>
<td>Direct object</td>
<td></td>
</tr>
<tr>
<td>Dest</td>
<td>Indirect object (container)</td>
<td></td>
</tr>
<tr>
<td>t-comp</td>
<td>tool complement</td>
<td>Tool used in an action</td>
</tr>
<tr>
<td>F-comp</td>
<td>Food complement</td>
<td>Food used as a tool</td>
</tr>
<tr>
<td>F-eq</td>
<td>Food equality</td>
<td>Identical food</td>
</tr>
<tr>
<td>F-part-of</td>
<td>Food part-of</td>
<td>Refer to a part of a food</td>
</tr>
<tr>
<td>F-set</td>
<td>Food set</td>
<td>Refer to a set of foods</td>
</tr>
<tr>
<td>T-eq</td>
<td>Tool equality</td>
<td>Identical tool</td>
</tr>
<tr>
<td>T-part-of</td>
<td>Tool part-of</td>
<td>Refer to a part of a tool</td>
</tr>
<tr>
<td>A-eq</td>
<td>Action equality</td>
<td>Identical action (Ac, Af)</td>
</tr>
<tr>
<td>V-tm</td>
<td>Head verb for timing, etc.</td>
<td></td>
</tr>
<tr>
<td>other-mod</td>
<td>Other relationships</td>
<td></td>
</tr>
</tbody>
</table>

Table 4: Flow graph edge labels.
Method:

1. Preheat oven to 180°C / Gas mark 4. Lightly grease a 20x10cm (8x4 in) loaf tin.
2. In a large bowl, stir together flour, sugar, baking powder, bicarbonate of soda and cinnamon. Add egg whites, bananas and apple purée; stir just until combined. Pour batter into prepared tin.
3. Bake in preheated oven for 50 to 55 minutes, until a skewer inserted into centre of loaf comes out clean. Turn out onto wire rack and allow to cool before slicing.
Table 5: Edge estimation accuracy with gold r-NEs and with automatically recognized r-NEs.

<table>
<thead>
<tr>
<th>Label</th>
<th>#Gold</th>
<th>#Est</th>
<th>Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>Agent</td>
<td>674</td>
<td>371</td>
<td>55.1%</td>
</tr>
<tr>
<td>Targ</td>
<td>6210</td>
<td>5109</td>
<td>82.3%</td>
</tr>
<tr>
<td>Dest</td>
<td>1983</td>
<td>1576</td>
<td>79.5%</td>
</tr>
<tr>
<td>T-comp</td>
<td>650</td>
<td>540</td>
<td>83.0%</td>
</tr>
<tr>
<td>F-comp</td>
<td>286</td>
<td>252</td>
<td>88.2%</td>
</tr>
<tr>
<td>F-eq</td>
<td>1139</td>
<td>227</td>
<td>19.9%</td>
</tr>
<tr>
<td>F-part-of</td>
<td>737</td>
<td>91</td>
<td>12.3%</td>
</tr>
<tr>
<td>F-set</td>
<td>23</td>
<td>2</td>
<td>8.7%</td>
</tr>
<tr>
<td>T-eq</td>
<td>316</td>
<td>41</td>
<td>13.0%</td>
</tr>
<tr>
<td>T-part-of</td>
<td>190</td>
<td>71</td>
<td>37.5%</td>
</tr>
<tr>
<td>A-eq</td>
<td>237</td>
<td>85</td>
<td>35.7%</td>
</tr>
<tr>
<td>V-tm</td>
<td>640</td>
<td>474</td>
<td>74.1%</td>
</tr>
<tr>
<td>other-mod</td>
<td>2776</td>
<td>2033</td>
<td>73.2%</td>
</tr>
<tr>
<td>Total</td>
<td>15,861</td>
<td>10,883</td>
<td>68.6%</td>
</tr>
</tbody>
</table>

Table 6: Edge estimation recall.

5. Conclusion

Taking as a starting point research on recipes written in Japanese (Mori et al., 2014), we have presented an annotated corpus of English cooking recipe procedures, and described and evaluated computational methods for learning these annotations. In this paper we have demonstrated that:

- the approach to annotating and analysing recipes transfers well to a typologically different language;
- the annotation scheme supports high inter-annotator agreement; and
- recipe named entities can be identified with good accuracy using a deep neural network tagger.

The paper described how we annotated each recipe in the corpus with recipe named entities (r-NEs); inter-annotator agreement was high. A deep neural network NER tool

---

6To detect prepositional context, we use the RASP grammatical analysis system (Briscoe et al., 2006)—having adapted its lexicon to account for the non-standard writing style of cooking recipes.

7In this setup, we are pipelining two modules; eventually we would like to combine these into a single model by training on both module representations jointly.
trained on this data obtained an overall F_1 of 87.5. This accuracy is comparable to r-NE tagging of Japanese recipes. Starting from the r-NE annotation, inter-annotator agreement for flow-graph annotation was 82.3 F_1. This level of agreement is good, considering the very large space of possible edge annotations. Computing flow graphs used a dependency-style parsing procedure, which achieved an F_1 for edge detection of 71.1. Previous work on analysing Japanese recipes has shown that this level of accuracy is sufficient to support tasks including recipe information retrieval and symbol grounding for cross-modal cooking applications.

In future work, we intend to refine the procedure for extracting a flow graph from the weighted digraph of r-NE nodes. We will experiment with training classifiers for adding labels to edges, and for inserting links that form confluences in the flow graph. Although these are extra processing steps, they would add structure only monotonically to the minimum weight spanning arborescence, so should not have a negative impact on efficiency or on correct structure that is already in the flow graph.

6. Acknowledgements
This work was supported by JSPS KAKENHI Grant Number 18K11425 and ROIS NII Open Collaborative Research 2020 (No. 19S0404).

7. Bibliographical References


