Understanding the Cooking Process with English Recipe Text

Yi Fan
Heidelberg Institute for Theoretical Studies
Heidelberg, Germany
yi.fan@h-its.org

Anthony Hunter
Department of Computer Science
University London College
London, United Kingdom
anthony.hunter@ucl.ac.uk

Abstract

Translating procedural text, like recipes, into a graphical representation can be important for visualizing the text, and can offer a machine-readable formalism for use in software. There are proposals for translating recipes into a flow graph representation, where each node represents an ingredient, action, location, or equipment, and each arc between the nodes denotes the steps of the recipe. However, these proposals have had performance problems with both named entity recognition and relationship extraction. To address these problems, we propose a novel framework comprising two modules to construct a flow graph from the input recipe. The first module identifies the named entities in the input recipe text using BERT, Bi-LSTM and CRF, and the second module uses BERT to predict the relationships between the entities. We evaluate our framework on the English recipe flow graph corpus. Our framework can predict the edge label and achieve the overall F1 score of 92.2, while the baseline F1 score is 43.3 without the edge label predicted.

1 Introduction

Recipes, one of everyday life's most common procedural texts, explain how to cook a dish using clear step-by-step instructions. With many global challenges directly related to food, nutrition and healthcare (Schulze et al., 2018; Branca et al., 2019), analysing recipe texts is now attracting attention in the field of natural language processing(NLP) and artificial intelligence as a cross-disciplinary research focus (van Erp et al., 2021).

Having demonstrated that flowcharts are reasonable representations of procedural texts like recipes due to their ability to clearly and concisely present workflows or processes (Maeta et al., 2015), much research has focused on how they can be obtained from text (Papadopoulos et al., 2022; Yamakata et al., 2020; Donatelli et al., 2021). Each flowchart node represents a single word or a sequence of words in the input text, and reflecting basic concepts such as tools, actions and materials, and the edges state the relationship between two nodes. An example of a flowchart with its original recipe text is shown in Figure 1, and we refer the reader to see more examples in Appendix D. Generally, the frameworks for this task includes two parts, named entity recognition (NER) and relationship extraction (RE). However, the previous frameworks for English recipes can only detect which nodes are interconnected without identifying the label of the edges in the flowchart, and the overall F1 score for the whole framework is only 43.3 (Yamakata et al., 2020). Although a subsequent proposal by Donatelli et al. (2021) improves the overall F1 score to 72.3, it still does not solve the edge label recognition problem, and there is room for performance improvement. Thus, this research aims to address these issues by proposing a better framework for understanding English recipe texts that can produce a complete flow graph based on the input recipe.

In our framework, we use BERT (Devlin et al., 2018), Bi-LSTM (Huang et al., 2015) and CRF (Lafferty et al., 2001) to recognise the named entities as the nodes of the output flow graph and then extract their relationship by applying BERT to construct the edges. Unlike the previous work, which employed a dependency parser or used manually selected features to produce a spanning arborescence to generate a flowchart, we treat the edge prediction task as a classification problem that can directly predict the relationship of any pair of nodes by taking advantage of the large pre-trained model. We evaluated our framework on the English recipe flow graph corpus (Yamakata et al., 2020), and our framework surpasses the existing methods' performance in this task when the edges' label is predicted. This implies that our framework has potential to provide guidance to cooking robots, detect allergic food sources in recipes, convert recipes into low-calorie versions, etc.

Findings of the Association for Computational Linguistics: ACL 2023, pages 4244–4264
July 9-14, 2023 ©2023 Association for Computational Linguistics

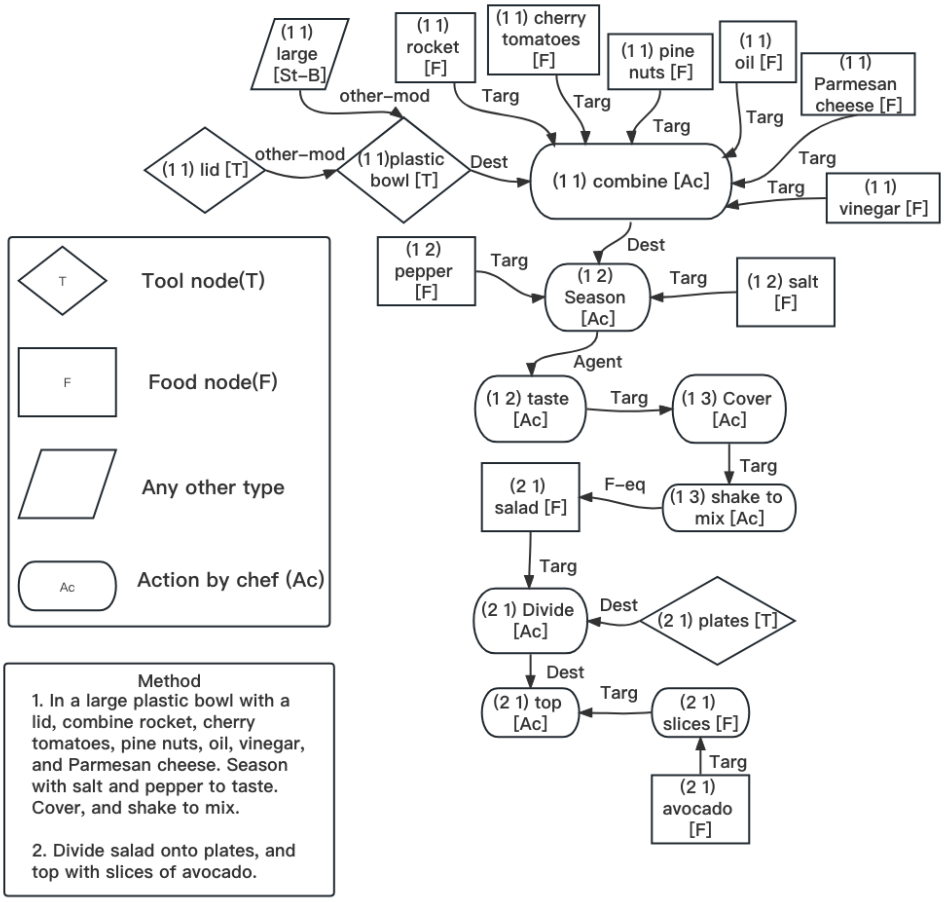

Figure 1: The visualisation of the output flow graph for the recipe "Easy rocket, cherry tomato and pine nut salad."

2 Related work

There has been a growing interest in the analysis of recipes by scholars in NLP and deep learning, for example, studying of recipes from a linguistic perspective (Jurafsky, 2014), detecting allergens in recipes for users to improve recipe recommendation systems (Alemany et al., 2016), finding patterns of spice and food combinations in recipes (Jain et al., 2015), exploring food and recipes to help meet the challenges of sustainable and healthy eating (Willett et al., 2019) and making a robot that can cook (Bollini et al., 2013).

Meanwhile, much research has focused on how to represent the semantics of the input text. Momouchi (1980) proposes using flowcharts to represent recipes, which was the first proposal for procedural text understanding. Representing recipes as graphs allows for analysis, such as measuring the similarity between recipes (Maeta et al., 2015)., which indicates that graph structure can usefully reflect the semantic information in a recipe. Mori et al. (2012) proposes a machine learning approach for processing Japanese recipe texts. The limitation of their study is that it only focuses on entity relations within the same sentence, which means that their work is insufficient to convert recipes into flowcharts. The manually annotated results of Mori et al. (2014) show that directed acyclic graphs are a suitable representation for natural language understanding of the procedural cooking text. Maeta et al. (2015) proposes a framework composed of NER and maximum spanning tree estimation for procedural text understanding. Although the results of their framework tests on the Japanese recipe corpus Mori et al. (2014) are not very encouraging, the design of the whole framework is inspiring for the follow-up work. Yamakata et al. (2017) annotates a corpus of English recipes containing 100 recipes using the method of Mori et al. (2014). They aim to analyse the structural differences between Japanese

recipes and English recipes. They compare the annotated named entities in the Japanese and English corpora, identify lexical, semantic and underlying structural differences between the Japanese and English recipes, and analyse their reasons. One of their most important results is the demonstration that it is feasible to annotate English recipe texts using the Mori et al. (2014) guide, as most previous work has focused on Japanese recipe texts.

Building on the work of Yamakata et al. (2017), Yamakata et al. (2020) add 200 recipe texts in English annotated with named entities and flowchart representations to the original corpus of 100 recipes to produce a new English flow graph corpus (300-r). The label of the named entities is shown in Table 1, and the label for the edges is shown in Table 2. They propose a framework (Y'20) for this corpus to output flowcharts. Firstly, they train BERT-NER for the NER task. Secondly, they treat all entities in a recipe as a sequence and extract the flowchart using a dependency-style maximum spanning tree parser. The result of edge construction for the flow graph is acceptable when the correctly labelled sequences are entered. However, the model's performance drops dramatically when the input is a sequence of named entities extracted from the previous model. Moreover, the edges of the flowchart in their output are unlabelled. Donatelli et al. (2021) applies a two-layered BiLSTM encoder generating predictions in a CRF output layer with ELMo embeddings (Peters et al., 2018) for the NER task. For predicting edges, they use the biaffine dependency parser by Dozat and Manning (2016), implemented by Gardner et al. (2018). Although they achieve state-of-art performance on the 300-r corpus, the labelling of the edges remains unaddressed. To the best of our knowledge, we are the first to design a framework to produce a complete flow graph for understanding English recipes.

3 The whole framework

To convert the input recipe text into flowchart output, we need two steps to process it, namely recognising named entities and identifying relationships between named entities. Therefore, in our complete framework, we have two main modules to deal with those two tasks. The following subsections will introduce why we choose such models and how we use them in detail.

Tag	Meaning
F	Food
T	Tool
D	Duration
Q	Quantity
Ac	Actions by the chef
Ac2	discontinuous action
Af	Action by foods
At	Action by tool
Sf	food state
St	tool state

Table 1: The named entities tags table

Label	Meaning
Agent	Subject
Targ	Direct object
Dest	Indirect object
T-comp	tool complement
F-comp	Food complement
F-eq	Food equality
F-part-of	Food part-of
F-set	Food set
T-eq	Tool equality
T-part-of	Tool parf-of
A-eq	Action equality
V-tm	Head verb for timing, etc.
other-mod	Other relationships

Table 2: The edge label table

3.1 Named entities

In this task, we use a combination of three models, which are Bidirectional Encoder Representations from Transformers (BERT), Bi-directional Long Short-Term Memory (BiLSTM) and Conditional Random Fields (CRF). We will describe why we have chosen these three models.

First, given BERT's powerful information memory and extraction capabilities, we chose to use BERT at the bottom level for extracting textual information considering the context to produce the embeddings. Next, we use the model BiLSTM, which captures semantic dependencies in both directions. In our module, the role of the BiLSTM is to extract useful information from the BERT output and integrate it in both directions. The last model we use is the CRF. After abstracting entity

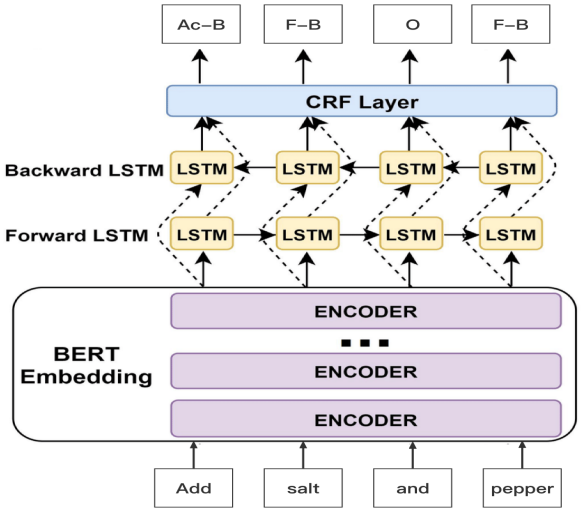

Figure 2: The architecture of the whole module for named entity recognition

recognition into a sequence labelling problem, one problem is that the predictions of labels are independent. Still, the accuracy of entity recognition is calculated by considering the span level rather than the single token level because, in the BIO tagging format, some entities are bound to each other. Therefore, it is necessary to compute the global optimum of the whole labelled sequence, taking into account the consistency of the label predictions within the entity, that is, to transform the problem of solving a classification problem for T number of words with N number of labels into finding the most appropriate prediction sequence from N^T sequences. For example, when we are predicting the entity labels of two words, the first word can not be labelled O (Other), with the second word being labelled Ac-I (Action Inside). This is because if the second word is labelled AC-I, the first must be Ac-B (Action Begin) in the context of only two words. So if the module predicts that the label of the first word is O, the probability that the label of the second word is any label with an -I suffix should be 0. We can see the whole module for named entity recognition in Figure 2.

When we input the entire recipe text into our NER module, it automatically tags each word. Therefore, for the NER task, our module exploits the idea of transfer learning, taking advantage of the pre-trained model, i.e. BERT, which is very capable of acquiring dynamic word embeddings. It also uses BiLSTM for more adequate and accurate extraction of word representations. Finally, CRF solves the consistency problem of intra-entity label prediction.

3.2 Relation extraction

In this task, the model we use is BERT. Unlike the previous works using spanning tree estimation (Yamakata et al., 2020) or the dependency parser to generate the edges for the flow graph (Donatelli et al., 2021), we treat the edge prediction task as a classification problem. Our model inputs are two named entities and the sentences that these two entities are in. Thus we will have two cases where two named entities are in the same sentence and two named entities in different sentences. For convenience, we specify that when predicting the relationship between two named entities, the first occurrence of the named entity in the input text is e_1, and the second occurrence of the named entity is e_2.

In order to have the appropriate input, we need to re-label the data once more for our classification task, which is done automatically by our written script on the training set. The data format in the original corpus is shown in Figure 3. The first three numbers and the last three numbers of each line are used to identify the nodes in the flowchart, which represent the number of the step in which the entity appears, the number of the sentence in the step and the number of the word in the sentence, respectively in the input recipe text. For example, 2 3 1 means that the entity begins in the first word of the third sentence in the second step of the input recipe. The fourth label indicates the label of the edge between the two nodes. Thus, each row represents a departure node pointing to an arrival node via an edge with a label. We specify that $l(e_1, e_2)$

represents the label of the edge pointed to e_2 by node e_1 as l and $l(e_2, e_1)$ means the label of the edge pointed to e_1 by node e_2 as l. And we need to add two new labels as $not\_relate(e_1, e_2)$ and $not\_relate(e_2, e_1)$, because as we explore the relationships of named entities in the text, we need to explore the relationships between all named entities. When the relationship between two named entities is not_relate, it means there is no relationship between them. Doing this allows us to use all the information in the input text more effectively. The essential point is that our model can predict whether two entities are connected and their relationship type. We implemented a method to identify the first entity in a pair that appears in the recipe as e_1 and the second entity as e_2 and extract the sentence they are in. If they are related, we will re-label them. Otherwise, their relationship will be $not\_related$. The new label for the edge prediction task is shown in Appendix A.

```
2 3 1 t 2 4 1
2 4 8 d 2 4 1
2 4 26 d 2 4 1
```

Figure 3: A part of the flow graph data for the recipe "Best Chocolate Chip Cookies".

Our model is BertForSequenceClassification, a pre-trained model of BERT provided by Hugging Face[1]. For the input to the model, we first add the [CLS] token to the input, meaning classification. We then concatenate the words in the named entities e_1 and e_2 with a space to separate them. This is then followed by the [SEP] token, meaning separation. If e_1 and e_2 are in the same sentence, we add the sentence directly after it and then add [SEP] at the end, as shown in Figure 4. If e_1 and e_2 are not in the same sentence, then we first add the [SEP] tag between the sentence where e_1 is and the sentence where e_2 is and then add the [SEP] tag at the end. This is concatenated after the [SEP] following the two words, as shown in Figure 5.

3.3 The framework

First, we construct the flowchart nodes for the input recipe by identifying all the named entities in our input recipe text by our named entity recognition module. The next step is to use our classification model to extract the relationships between all named entities by combining the output of the first module with the input text to construct the labelled edges of the flowchart. Finally, we ignore the entities with not_relate labels from the output of the second model, and we build a complete flowchart directly.

[1] https://huggingface.co/docs/transformers/model_doc/bert

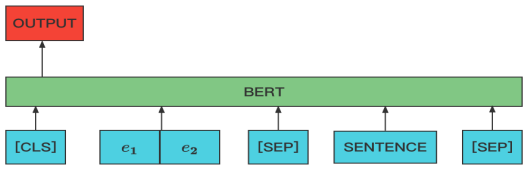

Figure 4: Two named entities in the same sentence

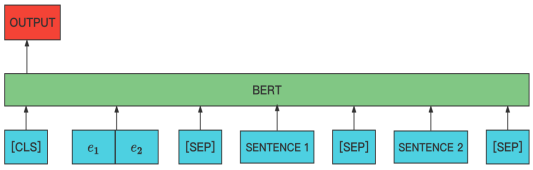

Figure 5: Two named entities not in the same sentence

3.4 Other experiment settings

We conducted our experiments on the 300-r corpus. By applying 10-fold cross-validation, we randomly split the data into 270 recipes for training and 30 for testing. The optimiser for both tasks is Adam (Kingma and Ba, 2014). We evaluate and compare the results by choosing precision, recall, and F1 score. Also, we will consider the accuracy result in the test set.

4 Results

4.1 The named entity recognition task

We first look at the performance of our module in the named entity recognition task, as illustrated in Table 3. Since Y'20 and D'21 show results for the model on all data; for comparison purposes, we also show results for our module run on all data. We can observe from the table that our module is overall superior to theirs. This is a satisfactory result since our dataset contains only 300 recipe texts. The accuracy of our module on the test set was **98.5%**. Y'20 and D'21 do not disclose the performance of their model on the test set. The most crucial point is that our high accuracy results on the named entity task can help tremendously with the next step of the relationship extraction task in

the overall framework since an entity relationship extraction can only perform well when the input is correct. Otherwise, the wrong input entities will propagate the error to influence the building of the whole graph.

Model	Precision	Recall	F1
Y'20	86.5	88.8	87.6
D'21	89.9 ± 0.5	89.2 ± 0.4	89.6 ± 0.3
Ours	**94.4**	**93.9**	**94.1**

Table 3: The comparison of the result for the named entity recognition task between three frameworks on the corpus 300-r.

4.2 The relationship extraction task

In this section, we present the performance of our module when the input is the correct entity and then when the input is the result of our named entity recognition. And we will compare our results with the previous works, shown in Table 4.

Because their work predicts edges without labels, we conducted two experiments to compare, one predicting edges without labels and one predicting edges with labels. From Table 4, we can observe that our module performs much better, regardless of whether our module predicts edges with or without labels for flowcharts. Although the performance drops slightly when the module tries to predict the edges' labels, we can still have a reasonably good overall result.

On the other hand, our module obtained an F1 score of **98.1** for the entire data input without predicting the labels of the edges. In addition, our module obtained an F1 score of **84.3** on the test set when no edge labels were predicted. When predicting the edges with labels, our edge prediction module has an F1 score of **92.9** with all data input. According to Yamakata et al. (2020), this level of accuracy is sufficient to enable tasks including information retrieval of recipes and symbol grounding for cross-modal cooking applications.

When our module does not predict edge labels, the accuracy is **84.2%** on the test set. And when our module predicts edge labels, the accuracy on the test set is **73.0%**. This result shows that although we compare favourably with the results of previous works in general, the module performs slightly worse on the test set when predicting edge labels. The reason may be that there are still too few recipe texts to train, so the module does not generalise well enough and overfits the data. When predicting the edge labels, the module achieves an accuracy of 98.8% on the training set but still does not perform as well on the test set. Another reason could be the imbalance in the type of recipe text between the training and test sets. Our recipe data includes various recipes such as main courses, appetisers, bread and desserts, salads, etc. If, when splitting the training and test sets, most of the recipe types in the training set are main courses and most of the recipe types in the test set are desserts, the module can not predict the labels of the relationships very well. Still, it can determine whether there is a relationship between the named entities from their types and the rest of the information. This situation can be further explored in future work, where we can label and add more recipes of various categories to the data. We could then pay attention to the balance of classes when dividing the training and test sets to see if the module would perform better. Alternatively, we could focus the module on a single type of recipe, such as main dishes, to see if the module has a better text understanding for a particular kind of recipe.

4.3 The result of the overall framework

As seen from Table 4, when we take the output of our named entity recognition module as the input to the relationship extraction module, the framework achieves an overall F1 score of **90.3**, which is when the framework outputs the labels for the edges. Without predicting the labels of edges, our F1 score for the entire framework is **94.5**.

In Yamakata et al. (2020)'s study, they achieved an F1 score of 43.3 when they used the output of their named entity recognition module as input to the relationship extraction module, and this score was when the framework did not output the labels of the edges. There is a massive drop in the performance of their framework when compared to when they use the correct named entities as inputs. Our framework performed satisfactorily overall, although there was a slight drop compared to when the correctly named entities were used as input. The results of the whole framework are mainly credited to the high accuracy of our module in the NER task. Because during the subsequent extraction of relationships between named entities, if the identified entity is incorrect, then our relationship extraction module will not extract the correct relationship for that entity, no matter how well it is

Model	Input	Precision	Recall	F1
Y'20	gold tags	73.7	68.6	71.1
D'21	gold tags	80.4	76.1	78.2
Ours without edge label	gold tags	**99.9**	**96.3**	**98.1**
Ours with edge label	gold tags	**94.7**	**91.1**	**92.9**
Y'20	Y'20 tagger	51.1	37.7	43.3
D'21	ELMo tagger	74.4 ± 0.5	70.4 ± 1.0	72.3 ± 0.8
Ours without edge label	our tagger	**98.2**	**91.0**	**94.5**
Ours with edge label	our tagger	**92.1**	**88.6**	**90.3**

Table 4: The comparison of the result for the edge prediction task between three frameworks on the corpus 300-r. We show our framework's performance in the case of predicting the edge label and not predicting the edge label.

trained. Secondly, in Y'20, if there is an incorrect entity or an edge in the middle of the flowchart that is incorrectly predicted, then their framework's prediction of the edges in the rest of the flowchart will be affected. The novelty of our framework is that even if there is a named entity in the middle of the flowchart that is inherently incorrect or an edge that is incorrectly predicted does not affect our framework's predictions for the rest of the flowchart at all. In terms of results, not only do we have a better result than previous works when we do not predict the edge labels, but our framework also performs well when predicting the edge labels. Also, when comparing the relationship extraction module tested separately, there is a slight drop in the framework's performance. Still, because of our framework design ideas and the high accuracy of the named entity recognition module, there is not a massive drop in the performance of our framework.

5 Discussion

While our framework addresses the remaining issues of low accuracy in identifying named entities and the inability to predict edge labels, we still have some problems. The flowchart we get when we enter "Cook the rest of the pancakes, one at a time but remember to melt a small knob of butter before adding the batter" in the framework is shown in Figure 7, and the correct flowchart is shown in Figure 6. We can observe that our framework identifies the rest of the flowchart correctly, except for the relationship between the named entity "Cook" and the named entity "Melt". In the sentence, "Melt" is the action on the butter and "Cook" is the action on the whole pancake. So the butter should be melted before the pancakes can be cooked, while our framework thinks the pancakes should be cooked before the butter is melted. However, surprisingly, if we change the input to "Remember to melt a small piece of butter before adding the batter to cook the pancakes." Our framework can correctly predict all the relationships. We can see those two inputs express the same meaning, but we obtained a different result. Because of the flexibility of the language, the same recipe may be written in several ways. In this example, organising the input sentence with a clear order and purpose is an improvement for the learning process of the framework. Hence, it illustrates getting the framework to learn the expression of the same sentence in different writing styles is also a key point in natural language understanding. Using data augmentation to change the same sentence into diverse representations with the same named entity information inside would give our framework a more robust text understanding capability.

Besides, we can see that one of the weaknesses of our framework is that we do not have a specific strategy for choosing the root node. We should select "Cook" as the root node in this case. Although our framework's accuracy is reasonable, it could be further improved if there was a suitable way to determine the root node. Maeta et al. (2015) suggests choosing the Ac labelled named entity that was last entered into the framework as the root node. However, in this example, we can observe that due to the language's flexibility, the last Ac labelled named entity could be "adding" or "cook". So this approach does not always work. This study can be further investigated in future work. In addition, as our framework focuses on learning the relationships between individual nodes, not much attention is paid to the completeness of the flowchart. This is the main drawback of our framework. Although a few incomplete graphs were observed during the test, it is still the prior issue we need to address in

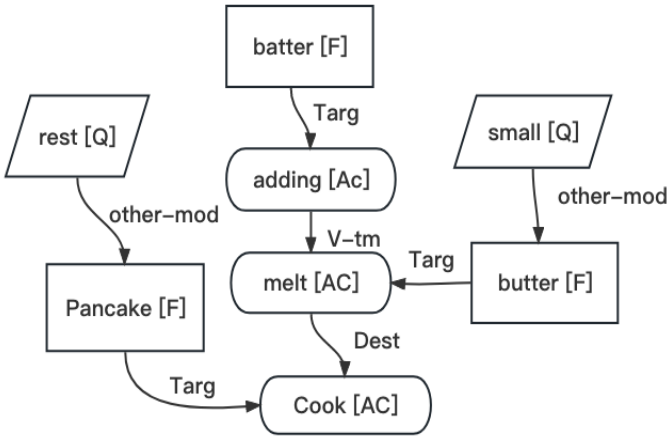

Figure 6: The gold standard from the dataset

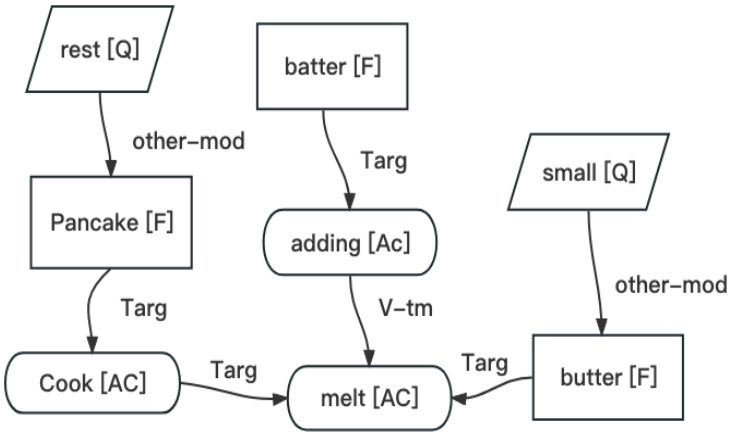

Figure 7: The flowchart produced by our framework

the future.

6 Conclusion

This paper proposes a framework for English recipe understanding by constructing flowcharts. This includes identifying the edge's label, indicating that we can efficiently output a complete flow graph. The overall framework performance achieves a satisfactory result and exceeds all previous work on the r-300 corpus, so it has the potential to be deployed in intelligent software kitchen assistants, cooking robots and diet management, etc.

Despite the good results of our framework, it still has some problems and some improvements that could be made in future work. Firstly, our framework performs better when predicting labels with more significant amounts of data. Therefore, adding more labelled recipes in the future would benefit the training. The second is that the framework can perform better by selecting a root node and forcing all nodes to be linked together. Choosing a root node is a difficult task, and a suitable root node can facilitate the learning process of the framework. However, if the correct root node is not chosen, this can have a significant negative impact. In the future, we could also try to use data augmentation to allow the framework to learn different sentences with the same semantic meaning, giving it a more vital generalisation ability. Also, we would like to investigate if the framework could understand recipe text better by focusing on only one type of recipe, such as dessert or main course. Moreover, exploring other more powerful pre-trained models and combining some linguistic features would be a good direction.

References

José Alemany, Stella Heras, Javier Palanca, and Vicente Julián. 2016. Bargaining agents based system for automatic classification of potential allergens in recipes. *ADCAIJ: Advances in Distributed Computing and Artificial Intelligence Journal*, 5(2):43–51.

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

Francesco Branca, Anna Lartey, Stineke Oenema, Victor Aguayo, Gunhild A Stordalen, Ruth Richardson, Mario Arvelo, and Ashkan Afshin. 2019. Transforming the food system to fight non-communicable diseases. *Bmj*, 364.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Lucia Donatelli, Theresa Schmidt, Debanjali Biswas, Arne Köhn, Fangzhou Zhai, and Alexander Koller. 2021. Aligning actions across recipe graphs. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6930–6942, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Timothy Dozat and Christopher D Manning. 2016. Deep biaffine attention for neural dependency parsing. *arXiv preprint arXiv:1611.01734*.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson Liu, Matthew Peters, Michael Schmitz, and Luke Zettlemoyer. 2018. Allennlp: A deep semantic natural language processing platform. *arXiv preprint arXiv:1803.07640*.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991*.

Anupam Jain, Rakhi NK, and Ganesh Bagler. 2015. Analysis of food pairing in regional cuisines of india. *PloS one*, 10(10):e0139539.

Dan Jurafsky. 2014. *The Language of Food: A Linguist Reads the Menu*. W. W. Norton.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, page 282–289, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

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

Yoshio Momouchi. 1980. Control structures for actions in procedural texts and pt-chart. In *Proceedings of the 8th Conference on Computational Linguistics*, COLING '80, page 108–114, USA. Association for Computational Linguistics.

Shinsuke Mori, Hirokuni Maeta, Yoko Yamakata, and Tetsuro Sasada. 2014. Flow graph corpus from recipe texts. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 2370–2377, Reykjavik, Iceland. European Language Resources Association (ELRA).

Shinsuke Mori, Tetsuro Sasada, Yoko Yamakata, and Koichiro Yoshino. 2012. A machine learning approach to recipe text processing.

Dim P. Papadopoulos, Enrique Mora, Nadiia Chepurko, Kuan Wei Huang, Ferda Ofli, and Antonio Torralba. 2022. Learning program representations for food images and cooking recipes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16559–16569.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Matthias B Schulze, Miguel A Martínez-González, Teresa T Fung, Alice H Lichtenstein, and Nita G Forouhi. 2018. Food based dietary patterns and chronic disease prevention. *bmj*, 361.

Marieke van Erp, Christian Reynolds, Diana Maynard, Alain Starke, Rebeca Ibáñez Martín, Frederic Andres, Maria C. A. Leite, Damien Alvarez de Toledo, Ximena Schmidt Rivera, Christoph Trattner, Steven Brewer, Carla Adriano Martins, Alana Kluczkovski, Angelina Frankowska, Sarah Bridle, Renata Bertazzi Levy, Fernanda Rauber, Jacqueline Tereza da Silva, and Ulbe Bosma. 2021. Using natural language processing and artificial intelligence to explore the nutrition and sustainability of recipes and food. *Frontiers in Artificial Intelligence*, 3.

Walter Willett, Johan Rockström, Brent Loken, Marco Springmann, Tim Lang, Sonja Vermeulen, Tara Garnett, David Tilman, Fabrice DeClerck, Amanda Wood, et al. 2019. Food in the anthropocene: the eat–lancet commission on healthy diets from sustainable food systems. *The Lancet*, 393(10170):447–492.

Yoko Yamakata, John Carroll, and Shinsuke Mori. 2017. A comparison of cooking recipe named entities between japanese and english. In *Proceedings of the 9th Workshop on Multimedia for Cooking and Eating Activities in Conjunction with The 2017 International Joint Conference on Artificial Intelligence*, CEA2017, page 7–12, New York, NY, USA. Association for Computing Machinery.

Yoko Yamakata, Shinsuke Mori, and John Carroll. 2020. English recipe flow graph corpus. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 5187–5194, Marseille, France. European Language Resources Association.

A Datasets

The English recipes consist of the 100 recipes annotated by Yamakata et al. (2017) and 200 recipes annotated in the same manner by Yamakata et al. (2020). All the recipes are sampled from the Allrecipes UK/Ireland website. The named entities have eight types proposed in Mori et al. (2014) and two additional types—namely discontinuous action and action by a tool. For more precise details, please see Yamakata et al. (2020).

The new labels for the edge relationship extraction are shown in Table 5.

B Some other results

Since Y'20 released their result in detail as the baseline for the following research, we list the precise comparison of the result with them. The specific comparison of the result for the NER task between our work and Y'20 is shown in Table 6. The specific comparison of the result for the edge prediction task and the whole framework between our work and Y'20 is shown in Table 7. The comparison of the F1 score for a different type of output when the input is all data is shown in Table 8.

C Code

Our code is available on Github.

D Some visualisation of flow graph

In this section, we will present several flow graph visualisations, some of which are the output of our framework to give the reader a clear understanding of what it means.

Old label	New label
Agent	a(e1,e2)
	a(e2,e1)
A-eq	a-eq(e1,e2)
	a-eq(e2,e1)
Dest	d(e1,e2)
	d(e2,e1)
F-comp	f-comp(e1,e2)
	f-comp(e2,e1)
F-eq	f-eq(e1,e2)
	f-eq(e2,e1)
F-part-of	f-part-of(e1,e2)
	f-part-of(e2,e1)
F-set	f-set(e1,e2)
	f-set(e2,e1)
other-mod	o(e1,e2)
	o(e1,e2)
Targ	t(e1,e2)
	t(e2,e1)
t-comp	t-comp(e1,e2)
	t-comp(e2,e1)
T-eq	t-eq(e1,e2)
	t-eq(e2,e1)
T-part-of	t-part-of(e1,e2)
	t-part-of(e2,e1)
V-tm	v-tm(e1,e2)
	v-tm(e2,e1)
	not-relate(e1, e2)
	not-relate(e2, e1)

Table 5: The comparison of the old labels and new labels for the arc.

Tag	$Precision^*$	$Recall^*$	F_1^*	Precision	Recall	F_1	Total Number of Entities
F	90.5%	93.9%	92.1	96.4%	97.3%	96.9	5007
T	87.7%	90.1%	88.9	96.3%	95.8%	96.0	1904
D	87.4%	90.2%	88.8	98.4%	98.2%	98.3	589
Q	70.9%	76.8%	73.7	91.7%	95.1%	93.3	529
Ac	92.3%	93.1%	92.7	95.5%	94.0%	94.5	4977
Ac2	43.8%	46.8%	45.3	90.8%	91.0%	90.8	178
Af	51.4%	50.4%	50.9	95.3%	89%	92.0	255
At	60.0%	10.0%	17.1	91.8%	88.7%	90.2	15
Sf	66.1%	71.4%	68.6	91.6%	93.1%	92.3	1105
St	81.6%	82.5%	82.0	96.3%	97.1%	96.7	876
Total	86.5%	88.8%	87.6	94.4%	93.9%	94.1	15435

Table 6: The result for the named entity recognition task. The results marked with an asterisk belong to (Yamakata et al., 2020).

Label	$Recall^*$	$Recall^\star$	$Recall^\bullet$	$Recall^\diamond$	$Precision^\diamond$	$F1^\diamond$	Total number of data
Agent	55.1%	96.1%	93.3%	91.0%	92.5%	91.7	674
Targ	82.3%	98.3%	96.9%	94.4%	93.9%	94.1	6210
Dest	79.5%	98.0%	93.4%	90.8%	92.4%	91.6	1983
T-comp	83.0%	97.6%	86.6%	84.0%	91.4%	87.5	650
F-comp	88.2%	97.5%	90.5%	88.9%	90.5%	89.7	286
F-eq	19.9%	94.7%	92.9%	90.3%	91.4%	90.9	1139
F-part-of	12.3%	90.0%	85.8%	83.5%	88.9%	86.1	737
F-set	8.7%	100%	86.9%	84.6%	97.3%	90.5	23
T-eq	13.0%	95.8%	87.3%	84.7%	93.8%	89.0	316
T-part-of	37.5%	92.6%	87.3%	84.5%	84.0%	84.3	190
A-eq	35.7%	97.4%	92.8%	90.0%	94.6%	92.2	237
V-tm	74.1%	97.3%	94.0%	91.2%	96.9%	94.0	640
other-mod	73.2%	96.5%	96.9%	94.3%	90.0%	92.1	2776
Total	68.6%	96.3%	91.1%	88.6%	92.1%	90.3	15861

Table 7: The result of the edge prediction task. The results marked with $*$ belong to Yamakata et al. (2020). For our model in edge prediction task, the results marked with $\star$ are Recall without label, the results marked with $\bullet$ are Recall with label. The result marked with $\diamond$ is the result of our whole framework.

Label	F_1 without label	F_1 with label	Total number of data
Agent	98.0	94.1	674
Targ	99.1	96.7	6210
Dest	99.0	94.1	1983
T-comp	98.8	90.2	650
F-comp	98.7	91.8	286
F-eq	97.2	93.5	1139
F-part-of	94.7	88.6	737
F-set	100	93.0	23
T-eq	97.8	91.6	316
T-part-of	96.1	87.1	190
A-eq	98.7	95.0	237
V-tm	98.6	96.7	640
other-mod	98.2	94.7	2776
Total	98.1	92.9	15861

Table 8: The comparison of the F1 score for the relationship extraction task between predicting edges' labels and not predicting labels of edges.

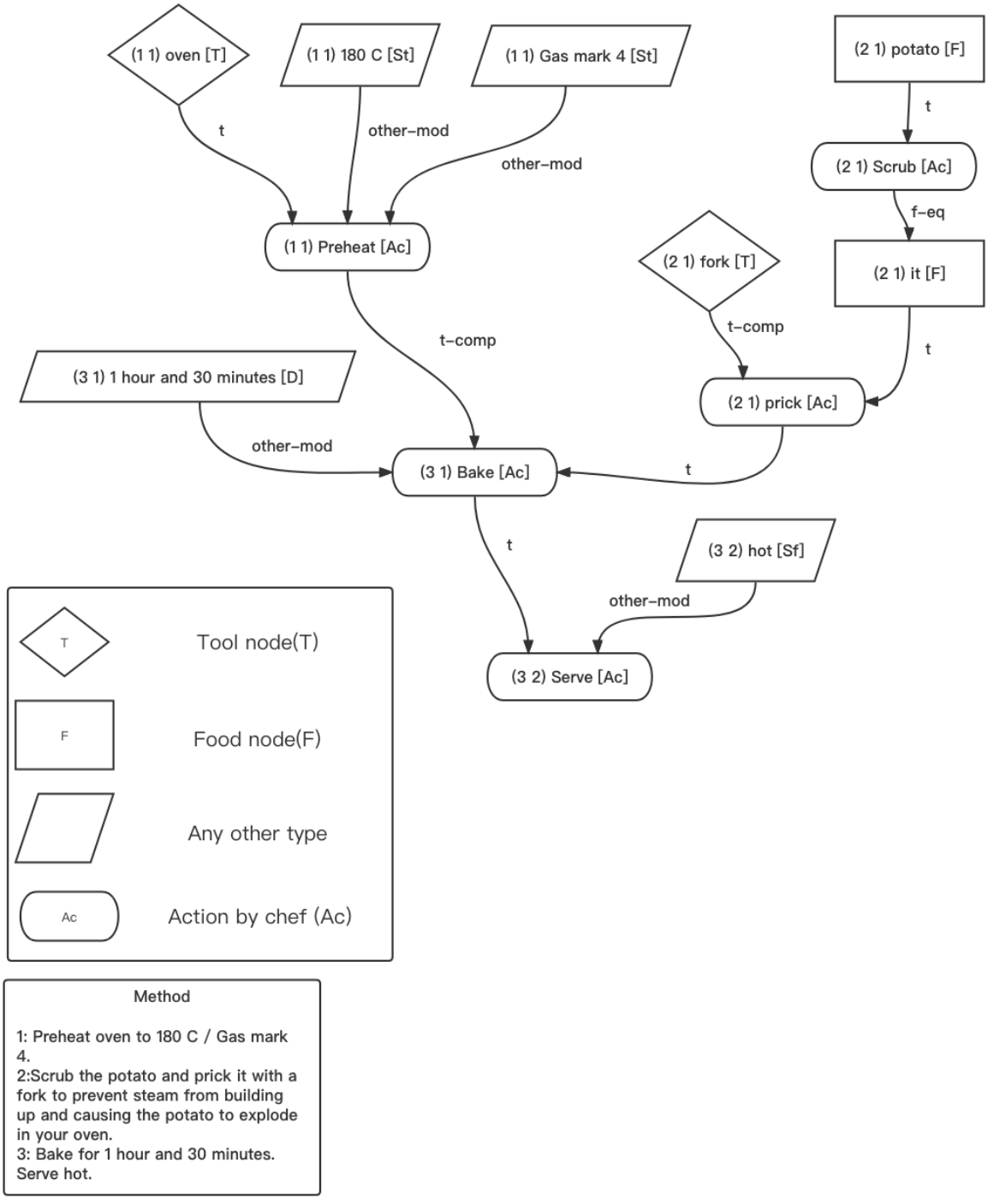

Figure 8: The visualisation of the output flow graph for the recipe “Baked Potato”

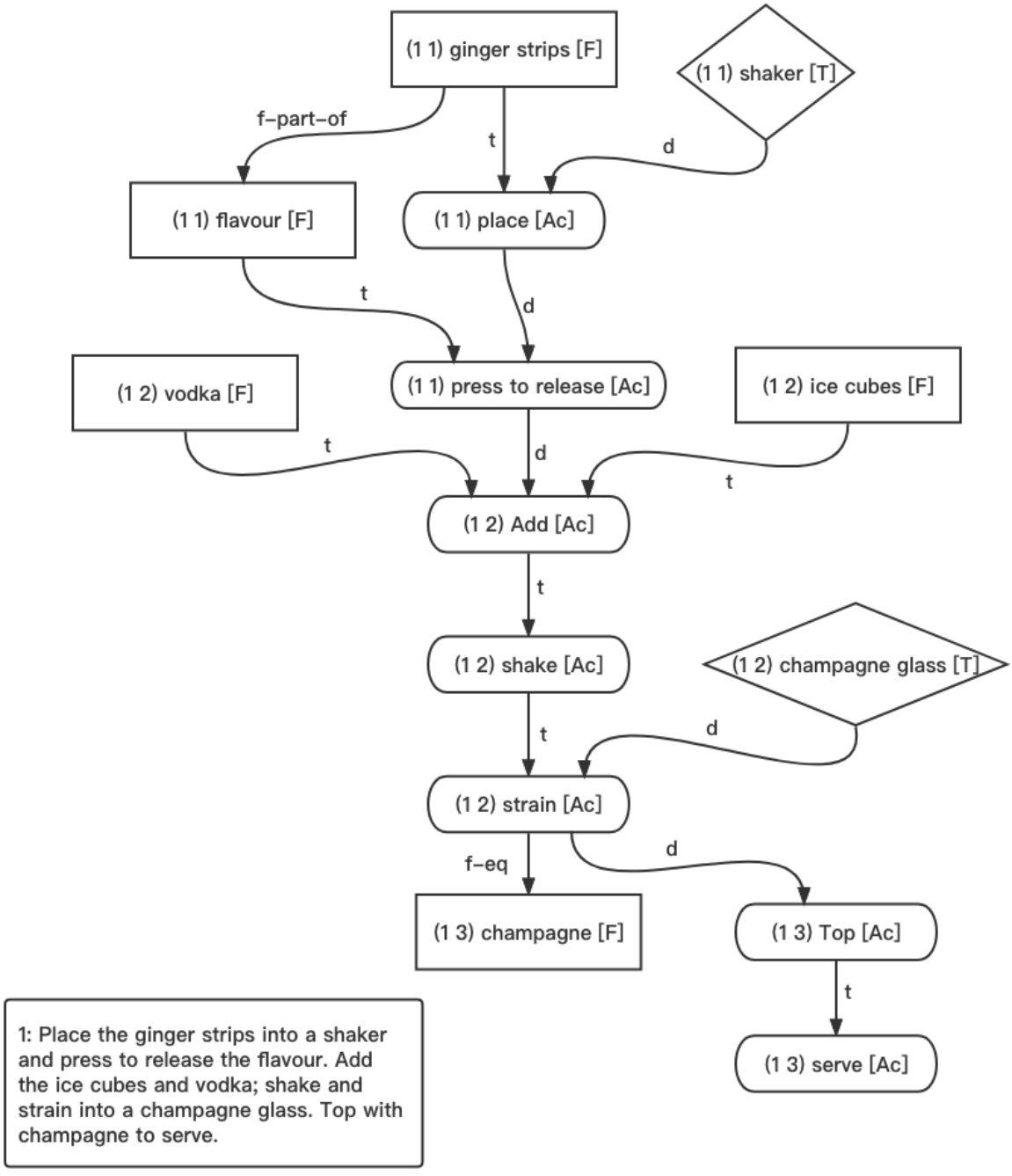

Figure 9: The visualisation of the output flow graph for the recipe “Ginger Champagne Cocktail”

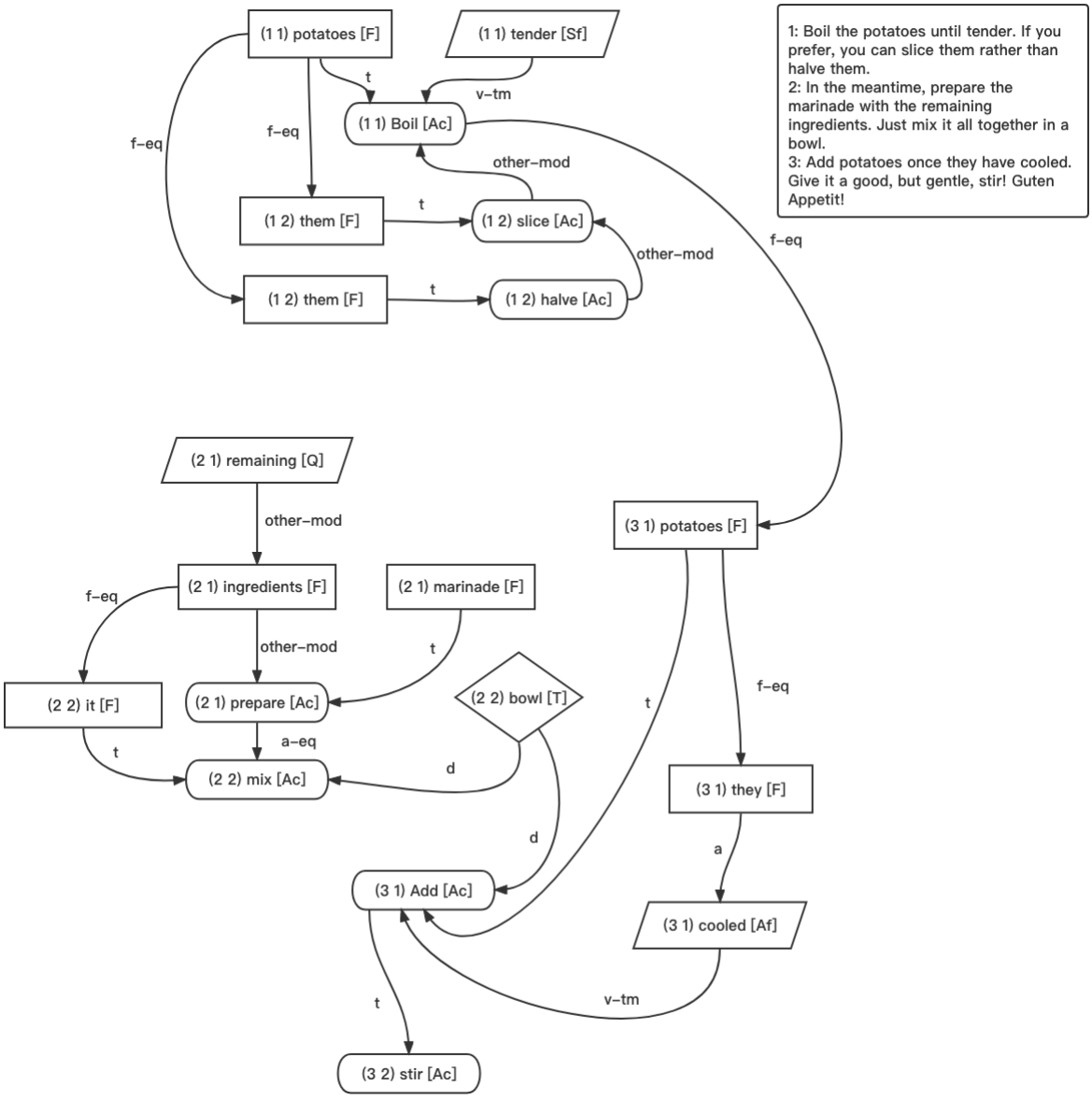

Figure 10: The visualisation of the output flow graph for the recipe “German potato salad”

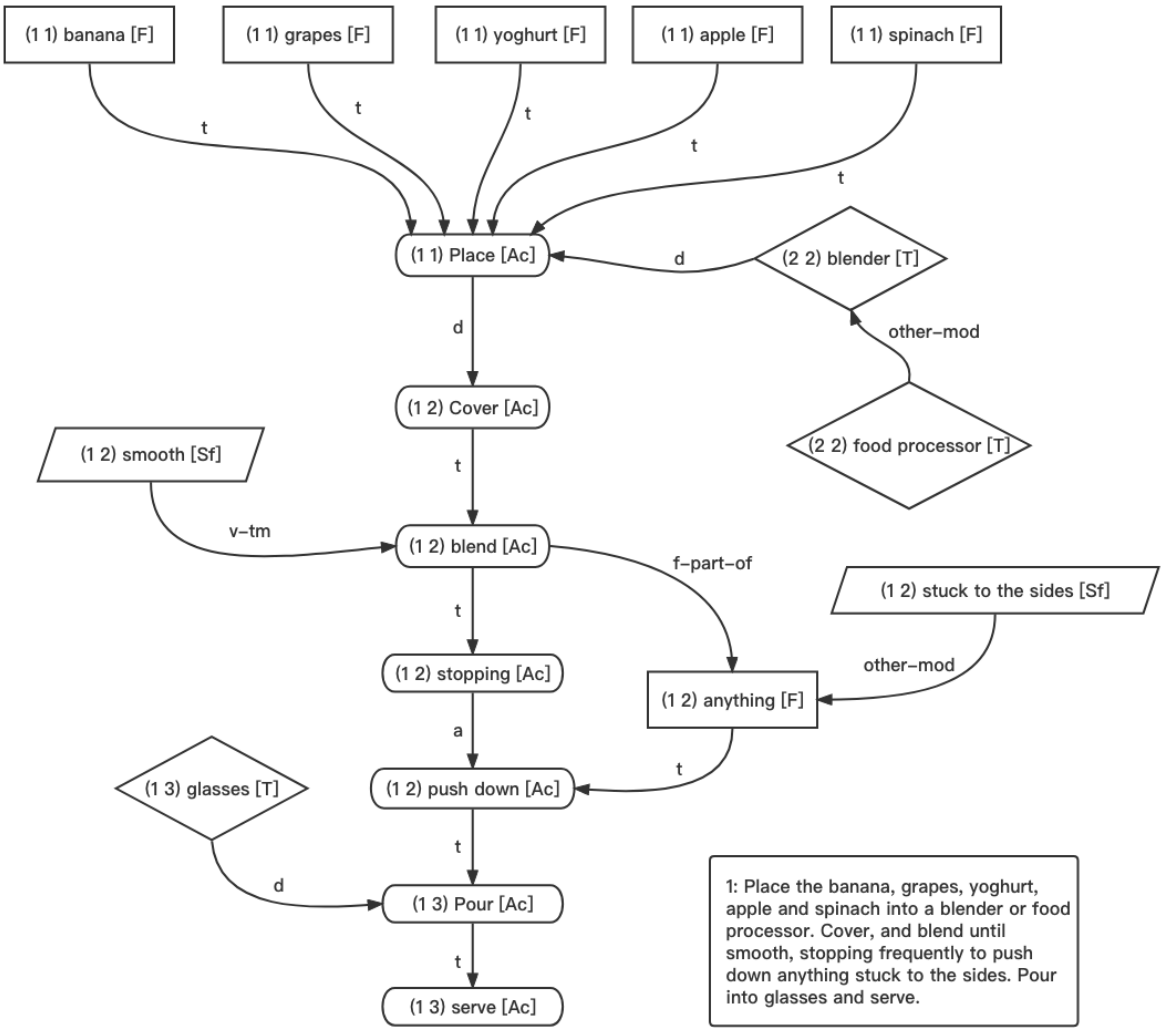

Figure 11: The visualisation of the output flow graph for the recipe “Groovy green smoothie”

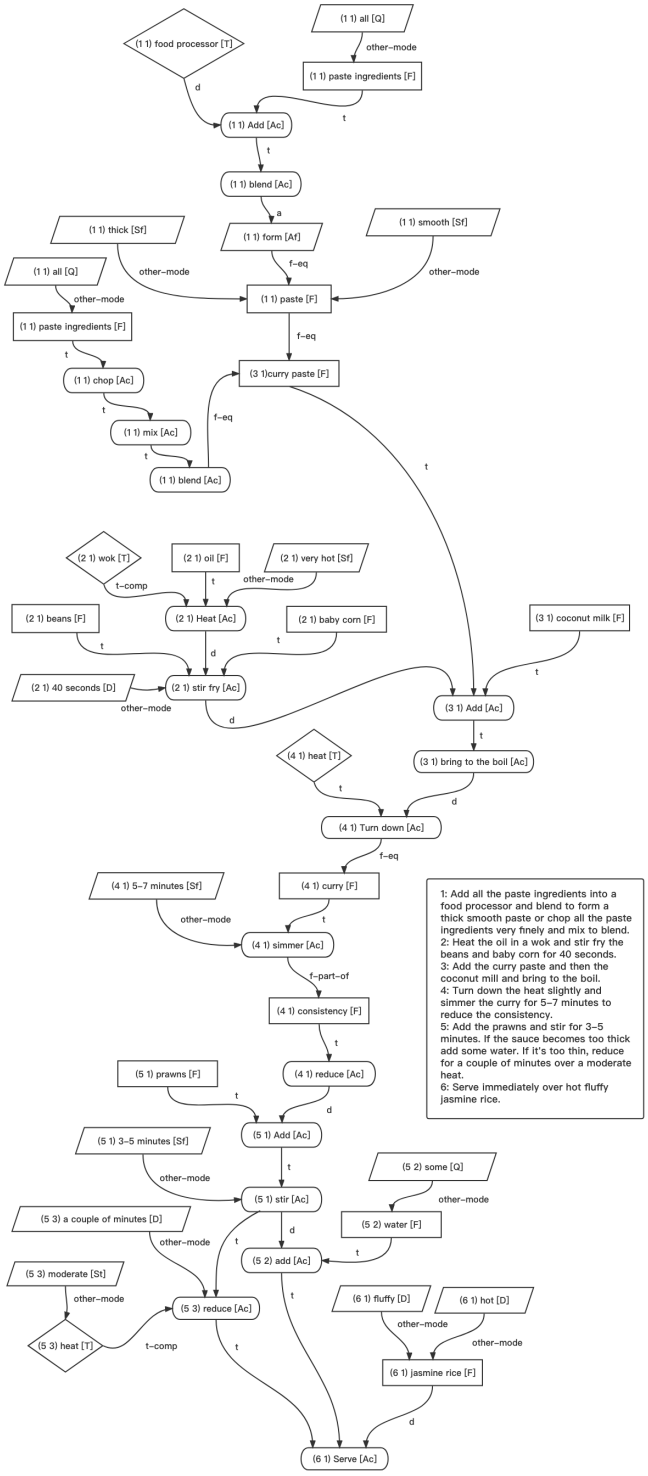

Figure 12: The visualisation of the output flow graph for the recipe “Thai green prawn curry”

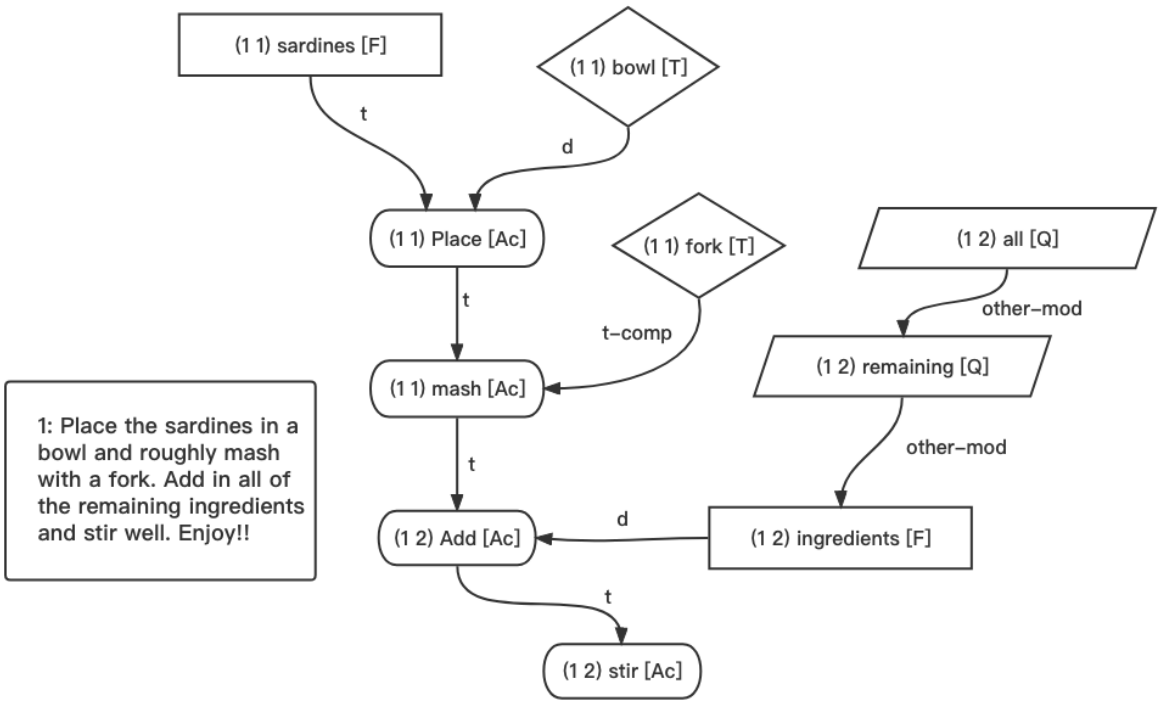

Figure 13: The visualisation of the output flow graph for the recipe “Sardine spread”

ACL 2023 Responsible NLP Checklist

A For every submission:

☑ A1. Did you describe the limitations of your work?
Left blank.

☑ A2. Did you discuss any potential risks of your work?
Left blank.

☑ A3. Do the abstract and introduction summarize the paper's main claims?
Left blank.

☒ A4. Have you used AI writing assistants when working on this paper?
Left blank.

B ☑ Did you use or create scientific artifacts?

1 and 3

☑ B1. Did you cite the creators of artifacts you used?
1 and 3

☐ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
No response.

☐ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
No response.

☐ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
No response.

☐ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
No response.

☐ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
No response.

C ☑ Did you run computational experiments?

3

☒ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
Space is limited. Once the paper has been accepted, I will add the details.

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.

☑ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
3

☑ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
3

☒ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
Space is limited. If the paper is accepted, I will disclose my code.

D ☒ Did you use human annotators (e.g., crowdworkers) or research with human participants?

Left blank.

☐ D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
No response.

☐ D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?
No response.

☐ D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
No response.

☐ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
No response.

☐ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
No response.