Building Flow Graphs for English Recipes

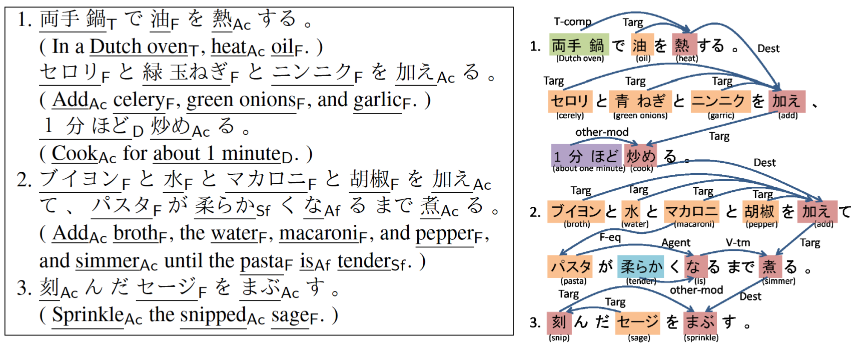
Leila Khalili (lkhalil), Mark Trawick (mtrawic)

September 2020

1 Background

1.1 Idea

We are interested in transforming recipes from their original text to a more computationally useful representation. We experimented with creating flow graphs, computationally tractable representations with steps linking actions to entities and the relationships between steps. (1; 2). The root of thee graph is the final dish created by following the recipe instructions. The nodes in the graph are the recipe domain specific concepts while labelled edges between nodes represent the relationships connecting these nodes. A flow graph can help to find similar procedures even though on the surface the linguistic expression is different. For example two instructions " add sliced banana to the batter" and "slice the banana and add it to the batter" are different textually but they belong to the same flow graph (3). Figure 2 shows an example of a flow graph in a Japanese recipe (2).



1. 両手 鍋$_{\mathrm{T}}$ で 油$_{\mathrm{F}}$ を 熱$_{\mathrm{Ac}}$ する 。
   (In a Dutch oven$_{\mathrm{T}}$, heat$_{\mathrm{Ac}}$ oil$_{\mathrm{F}}$.)
   セロリ$_{\mathrm{F}}$ と 緑 玉ねぎ$_{\mathrm{F}}$ と ニンニク$_{\mathrm{F}}$ を 加え$_{\mathrm{Ac}}$ る 。
   (Add$_{\mathrm{Ac}}$ celery$_{\mathrm{F}}$, green onions$_{\mathrm{F}}$, and garlic$_{\mathrm{F}}$.)
   1 分 ほど$_{\mathrm{D}}$ 炒め$_{\mathrm{Ac}}$ る 。
   (Cook$_{\mathrm{Ac}}$ for about 1 minute$_{\mathrm{D}}$.)
2. ブイヨン$_{\mathrm{F}}$ と 水$_{\mathrm{F}}$ と マカロニ$_{\mathrm{F}}$ と 胡椒$_{\mathrm{F}}$ を 加え$_{\mathrm{Ac}}$ て 、 パスタ$_{\mathrm{F}}$ が 柔らか$_{\mathrm{Sf}}$ く な$_{\mathrm{Af}}$ る まで 煮$_{\mathrm{Ac}}$ る 。
   (Add$_{\mathrm{Ac}}$ broth$_{\mathrm{F}}$, the water$_{\mathrm{F}}$, macaroni$_{\mathrm{F}}$, and pepper$_{\mathrm{F}}$, and simmer$_{\mathrm{Ac}}$ until the pasta$_{\mathrm{F}}$ is$_{\mathrm{Af}}$ tender$_{\mathrm{Sf}}$.)
3. 刻$_{\mathrm{Ac}}$ ん だ セージ$_{\mathrm{F}}$ を まぶ$_{\mathrm{Ac}}$ す 。
   (Sprinkle$_{\mathrm{Ac}}$ the snipped$_{\mathrm{Ac}}$ sage$_{\mathrm{F}}$.)

T-comp
Targ
Dest
other-mod
F-eq
Agent
V-tm
1. 両手 鍋 で 油 を 熱 する 。
(Dutch oven) (oil) (heat)
セロリ と 青 ねぎ と ニンニク を 加え 、
(cerely) (green onions) (garric) (add)
1 分 ほど 炒め る 。
(about one minute) (cook)
2. ブイヨン と 水 と マカロニ と 胡椒 を 加え て 、
(broth) (water) (macaroni) (pepper) (add)
パスタ が 柔らか く な る まで 煮 る 。
(pasta) (tender) (is) (simmer)
3. 刻 ん だ セージ を まぶ す 。
(snip) (sage) (sprinkle)

Figure 1: An example of a flow graph constructed for recipe instructions (2).

1.2 Application and Novelty

There are many useful applications of flow graphs. Since a flow graph represents the meaning of a sequence of procedures, it can be used to develop more intelligent search engines and improve machine translations of recipes between different languages. (1; 2). Additionally a flow graph may be used as input to other deep learning models. However, flow graphs are not designed as a diagram for a chef to follow while cooking.

There have been few studies on building flow graphs for English recipes. Maeta et al (1) derived flow graphs for Japanese recipes. The method they used is based on heuristic rules. In another study Kiddon et al (4), applied an unsupervised learning method to map instructional recipes to action graphs. As far as we reviewed there has been little work to construct flow graphs for recipes in English automatically. We hypothesize that we can automatically construct flow graphs for recipe texts in English. Moreover, we are using new feature vectors in our model.

2 Challenges

- Making a useful flow graph requires understanding relationships between words within a sentence and also relationships of words in different sentences. Finding the intra-sentential relationships can be easily found using the syntactic parsing. However, for inter-sentential relationships we sometimes need to identify anaphora, co-references and ellipsis, which is not straight forward. For example consider the instructions "Preheat oven to 180 C" and later in the recipe an instruction "Bake in preheated oven". The oven in both instructions are referring to the same oven but this fact is missed when only considering relationships within sentences.

  For parsing and finding the co-references we tried "The Stanford CoreNLP natural language processing toolkit" (5) to do the dependency parsing and finding the co-references. We realized that this toolkit can not find the co-references properly. For example let's consider a part of a recipe "Preheat oven to 180 C / Gas mark 4. Bake in preheated oven for 50 to 55 minutes, until a skewer inserted into centre of loaf comes out clean.". In this recipe there is a coreference relationship between "oven" and "preheated oven". The toolkit recognizes this relationship if the second sentence changes to "Bake in **the** preheated oven for ...". There are couple of problems of why this package can not identify these relationships which are explained in the following points. The main reason is that the written language of recipes is different from other types of written language.

- Cooking texts are written in a style different from most other text. This causes many challenges. They are often written as a list of commands, instead of normal sentences, which lack an explicit subject. The instruction "Boil pasta. Drain." has an implied subject of you the reader. Furthermore frequently sentences such as "Bake for 50 minutes" do not explicitly mention what to bake or where. Because cooking texts are different many standard NLP tools need to be adapted or trained on cooking texts explicitly. The Stanford NLP toolkit is trained on a domain which is different than than the domain we are interested in which contributes to it not solving our problem.

- Additionally, interpreting a recipe involves reasoning about how physical objects are changed while performing actions. For example to realize what the phrase "the wet mixture" refers to in a baking recipe requires a common sense knowledge about how objects in the world work interact together.

  So it was impossible for us to derive the flow graph with this method. Therefore, we found an alternative method, which is explained in section 3.

3 Implementation

To construct a flow graph there are two main steps. First identify recipe named entities (`r-NEs`) which serve as the nodes. Second, estimate the arcs to connect the nodes.

3.1 Recipe Named Entities

Each vertex in the flow graph represents a concept represented by a word sequence. In order to find these concepts we took the annotated recipes. The type of annotations are summarized in the table 3. Then entities consisting of multiple words are combined into one. For example the first instruction in the recipe starts with "Preheat the oven to 180 C/ Gas mark 4.". The annotated file for this sentence is as follows where the first three numbers show the number of the step, the number of the sentence in the step, and the number of word in the sentence. These three numbers are also used as node ID in the flow graph file. The fourth token shows the corresponding word, the fifth token shows part-of-speech tag automatically labeled with the parsing system RASP, and the last token shows manually labeled r-NE tag.

Table 1: Recipe named entity (r-NE) tags (3).

Tag	Meaning
F	Food
T	Tool
D	Duration
Q	Quantity
Ac	Action by chef
Ac2	Discontinuous Ac (English only)
Af	Action by food
At	Action by tool (English only)
Sf	Food state
St	Tool state

```
1 1 1 Preheat VV0 Ac-B
```

```
1 1 9 oven NN1 T-B
1 1 14 to II O
1 1 17 180 MC St-B
1 1 21 C ZZ1 St-I
1 1 23 / CC O
1 1 25 Gas NN1 St-B
1 1 29 mark NN1 St-I
1 1 34 4 MC St-I
```

where B, O, I stands for begin, outside and inside of the sentence. For getting the same concept we combined the words that are annotated to same r-NE and are just different in B, I, O. For instance "180 C" and "Gas mark 4". There are fewer nodes in the flow graph than words in the recipe text because after the `r-NE` identifying process any word that falls into the "Other" category is removed.

3.2 Drawing Arcs

In the second and final step the nodes are connected with arcs. Each arc has a label representing a certain relationship between the nodes. Table 2 lists the flow graph arc labels.

The direction of the arc generally represents order of completing the steps. To draw the arcs we constructed a maximum spanning tree connecting all the nodes (1). The weight of each arc, $s(u, v, l)$ is a probability defined by the function given in equation 1.

$$s(u, v, l) = \frac{\exp(\mathbf{\Theta} \cdot \mathbf{f}(u, v, l))}{\sum_{(x,r)\in(V\setminus\{u\}\times L)} \exp(\mathbf{\Theta} \cdot \mathbf{f}(u, v, l))} \tag{1}$$

Here L is the arc label set from Table 2, $\mathbf{\Theta}$ is a vector of weight parameters and $\mathbf{f}(u, v, l)$ is a function that takes edges u, v, and label l and returns a feature vector (explained in section 3.3). Intuitively s computes the probability of making a labeled from u to v with label l considering various features like word embedding, distance apart in the recipe text, surrounding words. To compute the weight parameter $\mathbf{\Theta}$ of the arcs we used a log linear model (6). Let $(V_t, u_t, v_t, l_t)_{t=1}^{T}$ denote a set of T training instances, where V_t is a set of the vertices and (u_t, v_t, l_t) is a ground truth arc having label l in the t-th training instance. With these training examples, the weight parameters are estimated by minimizing the equation 2. We used arc annotations from (3)'s dataset for our training data.

Table 2: Flow graph edge labels (3)

Label	Meaning
Agent	Subject
Targ	Direct object
Dest	indirect object (container)
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 part-of
A-eq	Action equality
V-tm	Head verb for timing, etc.
other-mod	other relationships

$$-\sum_{t=1}^{T} \log \frac{\exp(\mathbf{\Theta} \cdot \mathbf{f}(u, v, l))}{\sum_{(x,r)\in(V\setminus\{u\}\times L)} \exp(\mathbf{\Theta} \cdot \mathbf{f}(u, v, l))} + \frac{1}{2}||\mathbf{\Theta}||^2 \tag{2}$$

In order to minimize equation 2 we used the `scipy.optimize` package.

This method considers all possible arcs and arc labels between the nodes. After maximizing equation 2 using training data gives us $\mathbf{\Theta}$. With $\mathbf{\Theta}$ equation 1 gives us the weight for each arc. To decide which arc to include in the flow graph we construct a Minimum Spanning Tree. We used networkX python package (7) to construct the tree. The root of the tree was chosen the last word in the input with name entity of "Ac".

3.3 Extracting feature vectors

As it mentioned in the previous section (3.2) we need to derive feature vectors $\mathbf{f}(u, v, l)$ to train our model. Each pair in the training data set is defined as a concatenation of the following feature vectors. The features we considered are as follows:

1. The Google's Universal Sentence Embedding (8) of u and v. For example if u is the word "preheat" and v is "oven" then the feature is the embedding of "preheat oven".

2. A one hot encoding of whether u is in the same, subsequent, or previous sentence as v.

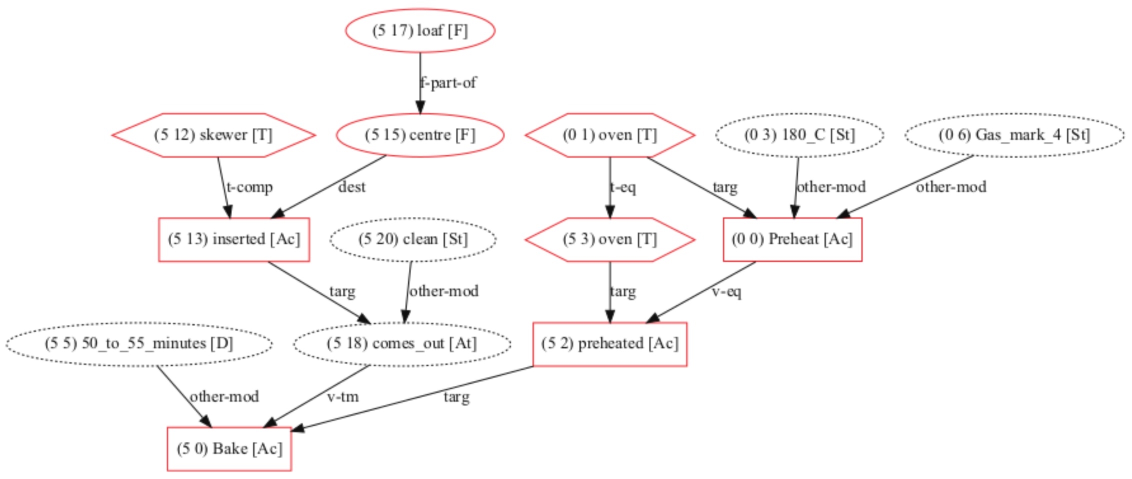


Figure 2: The flow graph of the example recipe which is manually created (3)

3. The distance in words between u and v in the recipe text divided by length of recipe text. If v occurs before u then the value is negative.
4. Whether or not u and v are in the same sentence and there is a preposition in between. We used NLTK POS tagger to determine prepositions.
5. A one hot encoding of the arc label

4 Results

The training data set we trained equation 2 consisted of 67 recipes. First we analyzed the edge labels of the recipes.

As you can see a lot of edges are ”Targ” or ”other-mode”. We tested our model on two pieces of recipes. Figure 2 shows the manually created flow graph of a piece of recipe ”Preheat oven to 180 C / Gas mark 4. Bake in preheated oven for 50 to 55 minutes, until a skewer inserted into centre of loaf comes out clean”.

Example 1: The first recipe is a single sentence of ”Preheat oven to 180 C / Gas mark 4”. The flow graph of this example is shown in figure 3. Figure 3a shows the output of the networkx. In figure 3b we plotted 3a. Comparing this flow graph with the ground truth flow graph, we can realize that the an edge between ”Preheat” and ”170_C” is missing. Since the weighted graph we got from networkx is a minimum spanning arborescence, we need to extend this to a directed acyclic graph by adding further edges. These new edges need to have weights smaller than a threshold. The red arrow shows the edge added after applying the latter procedure.

Example 2: The second example is applying our model to get the flow graph for the part of the recipe ”Preheat the oven to 170 C/ Gas mark 3. Bake in preheated oven for 50 to 55 minutes, until a skewer inserted into centre of loaf comes out clean”. The manually created flow graph is shown in figure 2. We have not applied the directed acyclic graph algorithm for a general case. This part is left for the future work.

Table 3: Number of different kind of edges in the recipes we used.

Label	Number
Agent	340
Targ	2239
Dest	658
t-comp	328
F-comp	96
F-eq	467
F-part-of	248
F-set	0
T-eq	136
T-part-of	41
A-eq	106
V-tm	223
other-mod	1003

By investigating the edge labels in flow graphs of figure 3 and figure 4a we realized that most of the labels are either ”Targ” or ”other-mod”. We think that since our training data was limited, the model was not trained enough on the labels other than the two mentioned labels. Therefore, we need to train the model on a bigger corpus in order to get more accurate results for the edge labels. We also tried our model on a training data that does not consider the edge labels 4b.

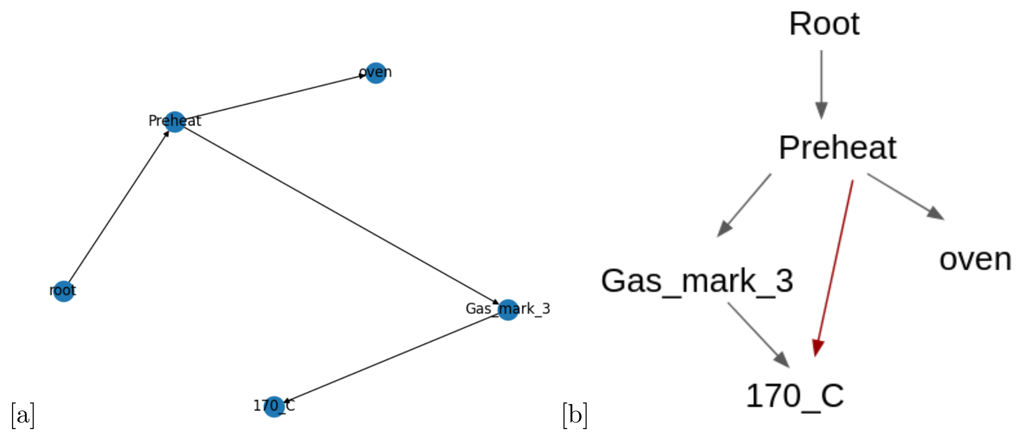


Figure 3: The flow graph we got for the sentence "Preheat the oven to 170 C/ Gas mark 3". a) shows the output of networkx. b) The same graph of part a after drawing edges smaller than a threshold.

As you can see in figure 4b this results are not accurate either. However, it can predict the corefernce between "preheated oven" and "preheat". Here, in order to distinguish between the words "oven" in first and second sentence we named them "*oven_1*" (in first sentence) and "*oven_2*" (in second sentence).

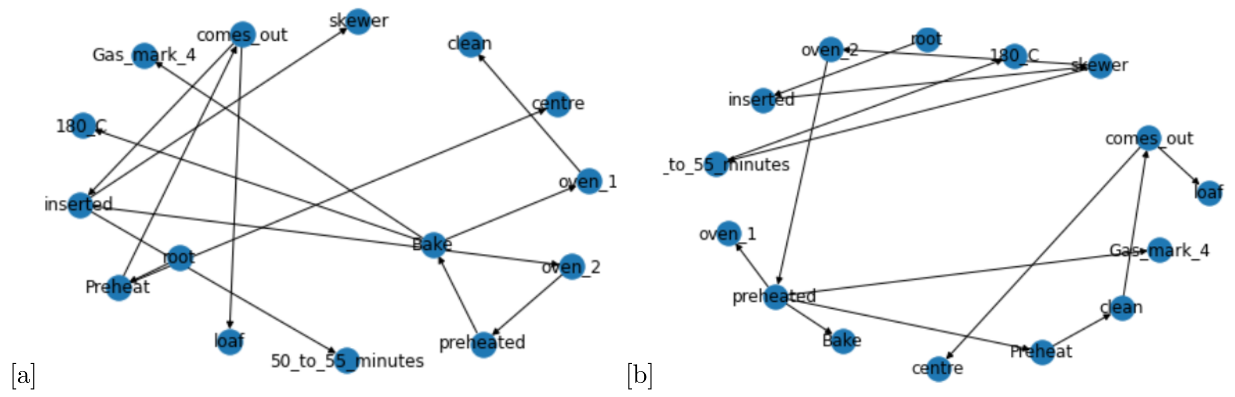


Figure 4: The flow graph we got for the sentence "Preheat the oven to 170 C/ Gas mark 3. Bake in preheated oven for 50 to 55 minutes, until a skewer inserted into centre of loaf comes out clean", a) considering the edge labels b) without considering the edge labels.

5 Technical Challenges

Among the challenges we faced during our work implementing the loss function and maximizing it proved the most difficult. Understanding the math and intuition behind it took many discussions and several re writings of the code after getting clearly wrong results. Finally, when we did achieve descent results the whole process of training and testing took many hours. For example training the model on 2/3's of our data took around 18 hours. With such a long time between making a change and seeing results we were not able to experiment with different features like we had hoped. Debugging was challenging as well. Another pitfall we encountered was using feature 3, the distance between words in the recipe. It was recommended by (3) without suggesting the very important detail of standardizing or normalizing the value. At first we were seeing graphs where the first item in the recipe was pointing to almost everything else due to a warped sense of importance of appearing well before other words.

6 Conclusion

In this project we intended to apply a framework consisted of two steps for procedural text understanding in the domain of English cooking recipes. We summarized what we learned from this work as the following items:

- Parsing an entire recipe text into a meaningful representation remains a **hard task**!
- Flow graphs are representations with steps linking actions to entities and the relationships between steps. These graphs can have lots of applications like intelligent search engines, input for deep learning models or machine translations of recipes between different languages.
- The written language of recipes are different than most of other texts. Also, for understanding recipes we need domain knowledge. These are some of the challenges in applying procedural text understanding in the domain of cooking.
- Applying a two step procedure including sentence-by-sentence semantic parsing and then finding inter-sentential relationships (eg. anaphora, co-references and ellipsis) failed. The reason is that the CoreNLP toolkit (5) that we used didn't include the cooking domain knowledge. Moreover, it is trained on texts which are not recipes.
- Deriving flow graphs is computationally very expensive. We trained our model on a small corpus. However, we believe training our framework on a bigger corpus will make the results a lot better.
- The framework we used gives reasonable flow graph for a single sentence. In the case of a bigger part of recipe it lacks some edges, however, it can find coreferences.
- The feature vector of each pair of vertices in the flow graph plays an important role in the accuracy of the flow graph.
- The statistics of the manually annotated edge labels shows that our data is very unbalanced. Our framework can not handle this unbalanced data and it returns the dominant labels most for the labels of most of the edges. We need to modify the framework to overcome this problem.

Our results show that in order to get useful flow graphs we need train our model on a bigger corpus. Moreover we need to modify our model to handle unbalanced data. We need to examine our model for defects and try different features.

References

[1] Maeta, H., Sasada, T. & Mori, S. A framework for recipe text interpretation. In *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct Publication*, 553–558 (2014).

[2] Mori, S., Maeta, H., Yamakata, Y. & Sasada, T. Flow graph corpus from recipe texts.

[3] Yamakata, Y., Mori, S. & Carroll, J. A. English recipe flow graph corpus. In *Proceedings of The 12th Language Resources and Evaluation Conference*, 5187–5194 (2020).

[4] Kiddon, C., Ponnuraj, G. T., Zettlemoyer, L. & Choi, Y. Mise en place: Unsupervised interpretation of instructional recipes. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 982–992 (2015).

[5] Manning, C. D. *et al.* The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*, 55–60 (2014).

[6] "Berger, A. L., Della Pietra, S. A. & Della Pietra, V. J. "a maximum entropy approach to natural language processing". In *Computational Linguistics*, 39–71 (1996). URL `https://www.aclweb.org/anthology/J96-1002`.

[7] Hagberg, A., Swart, P. & S Chult, D. Exploring network structure, dynamics, and function using networkx. Tech. Rep., Los Alamos National Lab.(LANL), Los Alamos, NM (United States) (2008).

[8] Cer, D. *et al.* Universal sentence encoder. *arXiv preprint arXiv:1803.11175* (2018).