Supplemental Material - EaT-PIM: Substituting Entities in Procedural Instructions using Flow Graphs and Embeddings

The following supplemental material provides details for our work's rule-based linking procedure (Section 1), the preprocessing steps used to perform part of our evaluation (Section 2), and some details about example recipes referenced in our discussion (Section 3). The content detailed in these supplemental material sections can also be found in our source code, which will be made available as open-source upon acceptance of the paper.

1 Rule-Based Verb-Noun Linking

We perform a rule-based method to convert the results of dependency relations and part-of-speech (POS) tags produced by spaCy. To apply these rules, we consider a single sentence at a time to extract all noun-verb pairs that are present – linking together information from different sentences is performed in a subsequent step. Detailed documentation related to these tags can be found at https://spacy.io/api. spaCy's tagging system is similar to dependency relations[1] and POS tags[2] published by Universal Dependencies. In the remainder of this section, we provide visualizations provided by spaCy of its dependency parse result, including annotations of the relation among each word and their POS tags.

The goal of this rule-based parsing is to identify pairs of verbs and nouns that are indicative of some entity and action taking place in the instructions. We also note that for nouns, we extract full noun phrases rather than the individual word. For example, from Figure 1, while "skillet" is the main noun, the rest of the adjectives are grouped together to form "a large deep skillet" as the full noun phrase. All such nouns are also cleaned to remove punctuation and stop words (such as "a" or "the"). The following subsections detail the main rules that we apply to extract such pairs – concrete implementation of these rules is present in within our experimental code's `eatpim/etl/parse_documents.py` script.

1.1 Direct Objects and Subjects

The most straightforward case is when a verb has a "direct object" (denoted `dobj`) relation with a noun. This can be seen in the beginning of Figure 1, where "bacon" is the direct object of the action "place." We thus extract the pair (bacon, place) from this sentence. This case applies to all situations where a noun has a direct dependency relation to a verb, and also applies to relations where

[1] https://universaldependencies.org/en/index.html
[2] https://universaldependencies.org/u/pos/

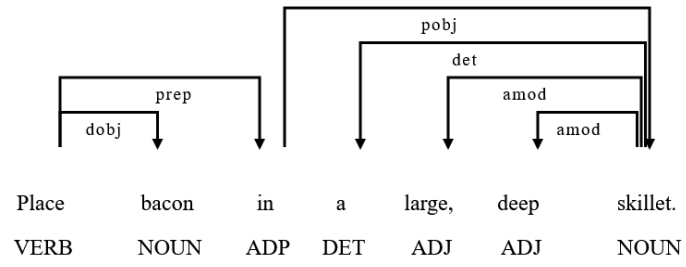

Fig. 1. An example parse including a simple direct object relation and a verb-noun pair connected by a preposition.

the noun has a "subject" relation to the verb (this is less common in procedural instructions, however, because they are typically formatted as having an implied subject – the user – perform the actions specified in each step).

1.2 Prepositions

While we do not wish to capture some of the detail found in prepositions in this work, we do need to handle prepositions to connect verbs with nouns. For example, once again in Figure 1, "in" has a "preposition" relation to the verb "place." It also is connected to "skillet" which is the object of the preposition. Whenever prepositions appear with a connection to a verb with a noun as the `pobj`, we extract it. For this example, the pair (place, large deep skillet) is extracted.

1.3 Phrasal Verb Particle

In some limited cases, we need to modify the verb to accuracy capture the meaning of the word. For example, in the sentence parsed in Figure 2, the action taking place should be to "Mix-in" (rather than cases where "in" is indicating the location where "mix" is taking place). Such situations are identified by the phrasal verb particle relation (denoted `prt`), which we use to modify the verb that we are extracting from the sentence. In this example, the pair (mix-in, tomatoes) would be extracted.

1.4 Conjunctions

Conjunctions such as "and" should indicate that the information is somehow being repeated, such as applying the action to multiple entities or applying multiple actions to the same entity. An example can be seen in Figure 3, where conjunctions are present between both verbs and nouns. From this sentence, we can tell that the actions to "peel" and "cut" should be applied to both "carrots" and "onions." There is a simple `dobj` relation between "cut" and "carrots," but

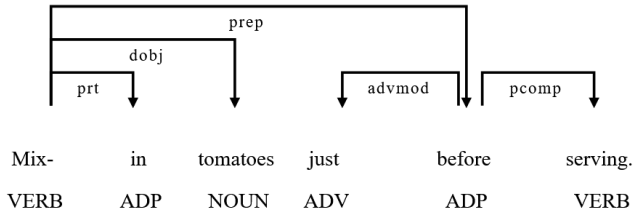

Fig. 2. An example parse of a sentence where the verb is modified by a `prt` relation.

the other two words of interest are only connected by "conjunction" relations. To handle such cases, in any instance where a "conjunction" relation occurs between two words, we apply verb-noun pairing to all connected conjunctions if it any one word has a pairing. Returning to our example, since the pair (cut, carrots) is extracted, the pairing with "cut" is replicated for "onions" to produce the pair (cut, onions). Similarly, since "peel" is connected to "cut," the pairing with "carrots" is repeated to produce (peel, carrot). Lastly, since "carrots" now has a new pair, the pairing is once again repeated to produce (peel, onions) for a total of four verb-noun pairs.

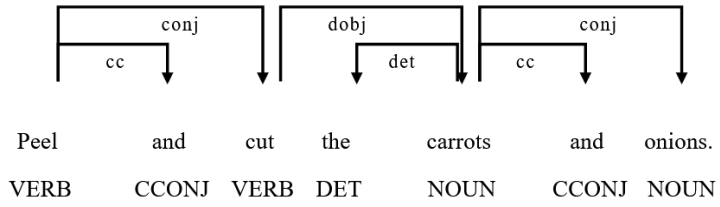

Fig. 3. An example sentence with conjunctions present in the action and the entities involved in the step.

1.5 Clause Complements and Modifiers

Lastly, we want to consider some special cases where a tool is being used to perform an action. Consider the example sentence in Figure 4, where we want to express the idea that we use the "knife" to perform the action "cut." As we can see from the dependency relations, "knife" is connected to the verb "use," which then has a clausal complement relation (denoted `xcomp`) to our action of interest, "cut." Another slight rewording of this same instruction can be seen in Figure 5. Due to the small change in the sentence's structure, in this example "using" has an adverbial clause modifier relation (denoted `advcl`) with the verb "cut."

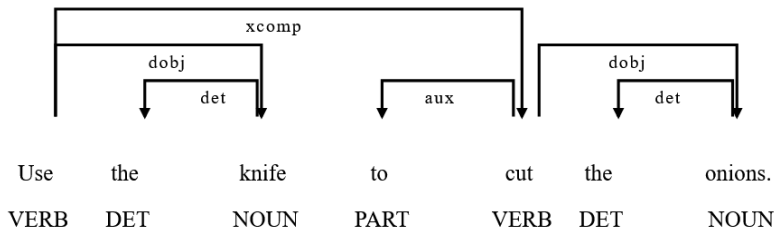

Fig. 4. An example sentence including a clausal complement, to "cut," which is dependent on the action to "use" the knife.

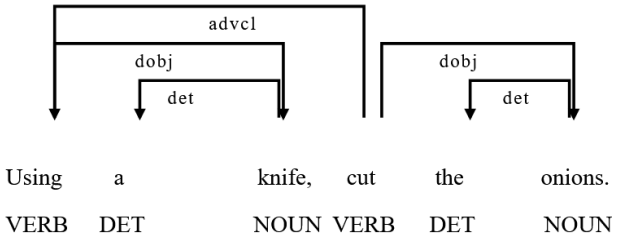

Fig. 5. Another example sentence which expresses a similar instruction as in Figure 4 in a different manner, which makes the dependent clause to "cut" now be identified as an adverbial clause modifier.

When we come across such situations in our instructions, we firstly choose to not focus on the verb "use" as the action to capture in our verb-noun pairs. Similar to how we do not wish to include detail such as prepositions, we instead want to focus on the main action that the tool is being used for – i.e., to "cut" some entity. Thus, whenever we encounter two verbs connected by a `xcomp` or `advcl` relation, if the dependent verb has a `dobj` relation to a noun, a pairing is created between the noun and the `xcomp` or `advcl` verb. For the two examples, both sentences would produce pair (knife, cut) for the noun "knife."

2 Preprocessing for Flow Graph Evaluation

The flow graph data published by Yamakata et al. is not made publicly available for download, but it can be accessed upon request from the authors' web page (https://sites.google.com/view/yy-lab/resource/english-recipe-flowgraph). This page also contains some simple information about the content of the dataset – further details about their tagging system and process used to annotate the flow graphs can be found in their publications referenced on the same page.

Compared to the flow graph data available from Yamakata et al.'s publication, the key differences compared to the flow graphs that we wish to produce are that (1) their flow graphs capture a greater level of detail (such as the duration

of an action and edge-labels indicating whether the role of an entity in an action) than ours, and (2) actions are represented as nodes rather than edges. The published flow graph data includes manual annotations for the part-of-speech, named entity tags, and relations for each word.

We first remove all leaf nodes present in the graph (i.e., nodes with no incoming edges) that are not annotated as Food or a part of a Food named entity. As our current experiments focus only on ingredient entities, we note that details related to equipment in the graphs are discarded. Next, we combine together words that indicate parts of ingredient names (annotated as `Food-Begin` or `Food-Inside`). This enables us to view the full names of ingredients as a single entity, similar to how we use full noun phrases produced by spaCy. For example, the ingredient "baking powder" would have the word "baking" annotated as `Food-Begin` and "powder" is `Food-Inside` – these are combined together as "baking powder" to form the full name of the ingredient.

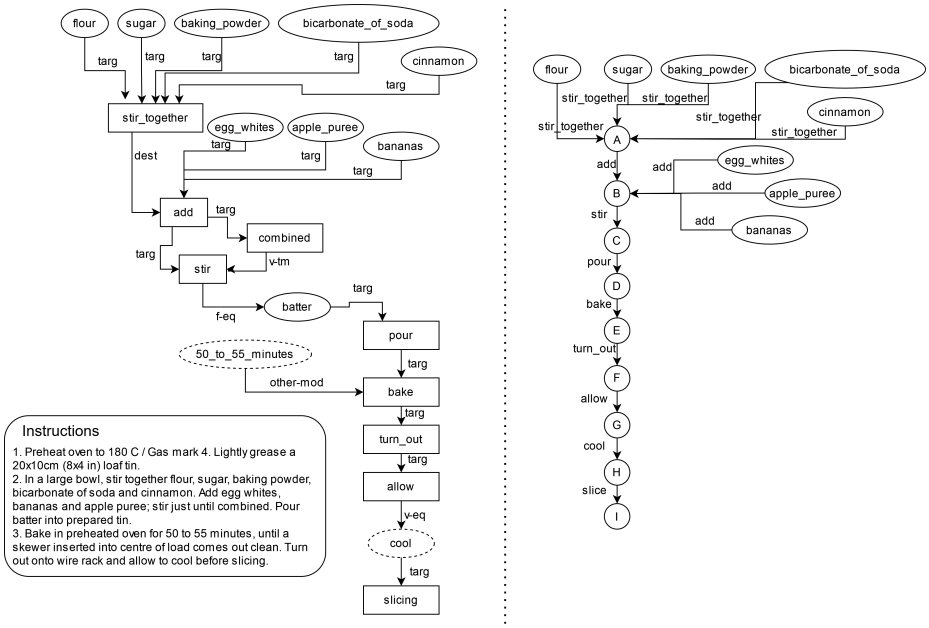

Fig. 6. An example comparing some content of a flow graph from the original dataset (shown on the left) to the preprocessed form used in our evaluation (shown on the right). Note that nodes related to tools and equipment from the original data have been omitted in this figure.

To remove details regarding the state of ingredients (such as to cook "until crispy"), we remove nodes annotated with the Food State or Tool State label. When such nodes are removed, we adjust the connections of the flow graph by connecting all incoming edges of that node to its outgoing edges, allowing us to

retain the connections of the graph. We similarly remove Tool nodes and adjust the edges of nodes connected to them.

Next, we make adjustments to the flow graph representation to ensure ingredients are all leaf nodes while retaining references to previous steps. Here, the `Food Equality` and `Food Part-of` edge labels can be used to indicate that food nodes within the flow graph are the same as an ingredient referenced from an earlier step. The first occurrence of such ingredients tended to be leaf nodes. Thus, when we encountered these relations, we followed the same procedure as the previous step to remove the node while shifting connections of incoming and outgoing edges.

To extract edge labels suitable for our flow graph formulation, we used the dataset's `Action` annotations to identify verbs. From here, we modify the graph so that the actions are captured as edges rather than nodes, and we replace the action node with an intermediate node (which represents the intermediate state of the food after applying the action to all incoming nodes).

An example comparing a flow graph from the original dataset to its preprocessed form can be seen in Figure 6. Notable differences include the removal of the detail to stir "until combined" and to bake for "50 to 55 minutes." The difference in edge labels primarily comes from how we shift over all actions from being nodes to being edges – for example, the node "stir_together" is replaced with an intermediate node, and the incoming edges are now the action "stir_together."

In comparing our results to those of Yamakata et al., we also note that they provide results for two settings – one where they use gold-standard named entity tags for each word in the instruction, and one where they train an automatic classifier. Their reported F1 score when using the gold-standard tags is 0.711, while their F1 score using the automatic entity tags is 0.433. They note that this difference in performance is likely because "an edge will always be wrong if the r-NE at even just one of its two end points is mis-recognized". The results produced by our work is more comparable to the automatic entity tagging setting in their experiments, since our method to generate flow graphs also performs PoS tagging and dependency parsing automatically.

3 Example Recipes

The following subsections provide details about the example recipes that use the ingredient Red Potatoes, included in our discussion section. The recipes were originally marked as **Mashed Potato** (Subsection 3.1), **Potato Gratin** (Subsection 3.2), and **Healthy Soup** (Subsection 3.3) in the main text.

3.1 "New" Mashed Potatoes

Recipe link from Food.com: https://www.food.com/recipe/new-mashed-potatoes-172744

Ingredients:

- Red Potatoes or New Potatoes (annotated only as Red Potatoes in the dataset)
- Salt Water
- Butter
- Red Onion
- Garlic
- Garlic Powder
- Heavy Cream or Creme Fraiche (annotated only as Heavy Cream in the dataset)
- Salt and Pepper

Instructions:

1. Scrub potatoes and cut into 1/2 inch cubes.
2. Place into cold salted water, enough to cover, and boil until tender.
3. Mince onions and garlic while potatoes are cooking.
4. Drain well.
5. In a large bowl, put drained potatoes, and 1 stick of butter mash lightly.
6. Add onions, garlic, garlic powder and salt and pepper.
7. Using a hand mixer, blend well (I leave some chunks).
8. Add remaining butter and enough cream to make smooth.
9. These reheat great and even better the next day when flavors have blended. I used them today to make a casserole, put into a buttered casserole dish and added some grated swiss and broiled about 5 minutes until cheese was browned a bit and bubbly.

3.2 (Potatoes with garlic and cheese) Pommes de terre a l'ail

Recipe link from Food.com: https://www.food.com/recipe/potatoes-with-garlic-and-cheese-pommes-de-terre-a-lail-15202

Ingredients:

- Bacon
- Olive Oil or Con Oil (annotated only as Olive Oil in the dataset)
- Red Potatoes
- Salt and Black Pepper
- Gruyere Cheese
- Garlic
- Parsley
- Double Cream

Instructions:

1. Wipe out a heavy cast iron deep frypan (cocotte) with an oiled paper.
2. Cut the bacon into 1/2 inch wide strips, and over a mediium heat cook until crisp and golden coloured, then add the oil and a first layer of potatoes.
3. Season well with salt and pepper and cover with a layer of grated cheese.

4. Continue filling the dish with alternate layers of seasoned potatoes and cheese until the dish is full.
5. Sprinkle with finely chopped garlic and parsley.
6. Cover the dish and place over a low heat.
7. Cook for 30 minutes shaking the dish occasionally but do not stir the contents.
8. When the potatoes are tender when pierced with a pointed knife, pour the cream over the surface, replace the lid and cook for a further 3 to 4 minutes.
9. Serve the gratin immediately either in the dish or turned out on to a heated serving dish, crusted side uppermost.

3.3 1-2-3 Healthy Soup

Recipe link from Food.com: https://www.food.com/recipe/1-2-3-healthy-soup-266081

Ingredients:

- Soy Beans
- Chicken Stock
- Broccoli
- Red Potatoes
- Green Cabbage
- Salt and Pepper

Instructions:

1. Cook the soy beans as directed on package in the chicken stock.
2. Once beans are cooked add all other ingredients to the borth and cook 15-20 minutes or until potatoes are cooked through.
3. Once soup has cooled to room temperature puree the broth in a blender.
4. Bon Apetite.