Aligning Actions Across Recipe Graphs

Lucia Donatelli, Theresa Schmidt, Debanjali Biswas,
Arne Köhn, Fangzhou Zhai, Alexander Koller
Department of Language Science and Technology
Saarland Informatics Campus
Saarland University
{donatelli,theresas,dbiswas,koehn,fzhai,koller}
@coli.uni-saarland.de

Abstract

Recipe texts are an idiosyncratic form of instructional language that pose unique challenges for automatic understanding. One challenge is that a cooking step in one recipe can be explained in another recipe in different words, at a different level of abstraction, or not at all. Previous work has annotated correspondences between recipe instructions at the sentence level, often glossing over important correspondences between cooking steps across recipes. We present a novel and fully-parsed English recipe corpus, ARA (Aligned Recipe Actions), which annotates correspondences between individual actions across similar recipes with the goal of capturing information implicit for accurate recipe understanding. We represent this information in the form of recipe graphs, and we train a neural model for predicting correspondences on ARA. We find that substantial gains in accuracy can be obtained by taking fine-grained structural information about the recipes into account.

1 Introduction

Cooking recipes are a type of instructional text that many people interact with in their everyday lives. A recipe explains step by step how to cook a certain dish, describing the actions a chef needs to perform as well as the ingredients and intermediate products of the cooking process. However, recipes for the same dish often differ in which cooking actions they describe explicitly, how they describe them, and in which order.[1] For instance, in the three recipes in Fig. 1, the overall process of assembling a batter and making waffles is explained with different levels of detail: recipe (a) explains the process with three distinct cooking actions (in bold); recipe (b) with eight distinct actions; and recipe (c) with nineteen actions.

[1]For now, we understand an *action* as synonymous to a semantic *event*. We use the former term for consistency with other recipe work.

(a) **Beat** eggs. **Mix** in remaining ingredients. **Cook** on hot waffle iron. [2]

(b) **Preheat** your waffle iron. In a large bowl, **mix** together the flour, salt, baking powder, and sugar. In another bowl, **beat** the eggs. **Add** the milk, butter, and vanilla to the eggs. **Pour** the liquid into the flour mixture and **beat** until blended. **Ladle** the batter into the waffle iron and **cook** until crisp and golden. [3]

(c) **Sift** together in a large mixing bowl flour, baking powder, salt, and sugar. In a jug, **measure** out milk. **Separate** eggs, **placing** egg whites in the bowl of standing mixer. **Add** yolks and vanilla essence to milk and **whisk** together. **Pour** over the flour mixture and very gently **stir** until about **combined**. **Stir** in the **melted** butter and **continue mixing** very gently until combined. **Beat** egg whites until stiff and slowly **fold** into batter. **Spoon** the batter into **preheated** waffle iron in batches and **cook** according to its directions. **Remove** immediately and **serve** with maple syrup and fruits. [4]

Figure 1: Three recipe texts for making waffles with cooking actions in bold. The recipes are ordered from least amount of detail (a) to most (c).

The set of recipes in Fig. 1 provides more general information about how to make waffles than each individual recipe can. To our knowledge, only one previous work focuses on alignment of instructions across recipes to facilitate recipe interpretation on a dish level: Lin et al. (2020) (henceforth referred to as L'20) present the Microsoft Research Multimodal Aligned Recipe Corpus to align English recipe text instructions to each other and to video sequences. L'20 focus on alignment of instructions as sentences. Yet, as sentences can be quite long and contain multiple actions, defining instructions at the sentence level often glosses over the relationship between individual actions and excludes the

[2]http://myhappytribe.com/recipes/waffles/
[3]http://www.tastygardener.com/waffles/
[4]http://bakesbychichi.com/2014/03/waffles.html

complex event structure that makes recipe interpretation challenging and compelling. For example, in L'20, *"**Ladle** the batter into the waffle iron and **cook** [...]"* in Recipe (b) is aligned to *"**Spoon** the batter into **preheated** waffle iron in batches and **cook** [..]"* in Recipe (c). If aligned at the sentence level only, the action correspondence of (b)'s recipe-initial *"Preheat"* and (c)'s *"preheated"* much later in the recipe is not (and cannot be) accounted for. The relationship between the individual actions ((b) *"Ladle,"* (c) *"Spoon"*) as well as between both instances of *"cook"* is additionally obscured by coarse-grained sentence alignment.

Aligning recipes at the action level (i.e. the bold items in Fig. 1) instead of the sentence level is practically useful to aggregate detailed information about how to cook a dish in different ways. This alignment would additionally offer greater insight into how recipe actions implicitly contain information about ingredients, tools, cooking processes, and other semantic information necessary for accurate recipe interpretation.

In this paper, we make two contributions towards the goal of complete and accurate alignment of actions across similar recipes. First, we collect a novel **corpus that aligns cooking actions across recipes**, Aligned Recipe Actions (ARA), (Section 4), by crowdsourcing alignment annotations on top of the L'20 corpus for a select number of dishes. In order to determine the annotation possibilities, we develop a neural recipe parser to identify descriptions of actions and substances and arrange them in a recipe graph (Section 3). Second, we present an **alignment model** which identifies correspondences between actions across recipe graphs based on our parsed corpus (Section 5). Error analysis of both human and machine performance illustrates that, though complex, the task of aligning recipe actions is achievable with our methodology and can inform future work on aligning sets of instructions. Our corpus and code are publicly available.[5]

2 Background And Related Work

Recipe text. The recipes in Fig. 1 illustrate some of the idiosyncracies of recipe text: all texts are written entirely in imperative mood ((a) "*cook*"; (b) "*preheat*"; (c) "*sift*"); definite noun phrases frequently drop their determiners ((a) "*eggs*"; (c) "*milk*," "*egg whites*"); many arguments are elided or left implicit ((b) "*beat* ∅"; (c) "*pour* ∅ *over*"); bare adjectives are used to describe desired end states ((b) "*until crisp and golden*"); and many anaphoric expressions refer to entities which were not explicitly introduced before ((b) "*the liquid*"; (c) "*the flour mixture*"). Accurate interpretation of single recipe instructions requires familiarity with situated food ingredients, knowledge of verb semantics to identify how each cooking step relates to the others, and general commonsense about the cooking environment and instructional syntax.

Recipe graphs and corpora. Representing recipes as graphs is the dominant choice (Mori et al., 2014; Jermsurawong and Habash, 2015; Kiddon et al., 2015; Yamakata et al., 2016; Chen, 2017; Chang et al., 2018; Özgen, 2019). Of relevance to this paper is the recipe corpus of Yamakata et al. (2020) (Y'20), which consists of 300 English recipes annotated with graphs as in Fig. 2. We train a recipe parser on this corpus (Section 3) and use the trained parser to identify actions in the L'20 corpus. As noted earlier, Lin et al. (2020) (L'20) created the Microsoft Research Multimodal Aligned Recipe Corpus of roughly 50k text recipes and 77k recipe videos across 4000 dishes in English. The recipes and video transcriptions were segmented into sentences (but not parsed), and 200 recipe pairs were manually annotated for correspondences between recipe sentences. L'20 cluster dishes based on exact match of recipe title; they perform sentence alignments on pairs of recipes using the unsupervised algorithm of Naim et al. (2014). We use L'20's dataset as a basis for our alignment task in Sections 4 and 5.

Recipe parsing. Parsing recipes into graphs is usually comprised of two steps: (i) tagging mentions of substances and cooking steps, and (ii) linking these mentions with input and output edges. Recent work on English recipes has achieved F-Scores above 90 for identifying mentions (Chen, 2017) and F-Scores above 80 for adding the edges (Jermsurawong and Habash, 2015; Chen, 2017; Özgen, 2019). Most of this work uses supervised learning based on hand-annotated recipe datasets. Using unsupervised methods, Kiddon et al. (2015) train a generative neural model on a large corpus of unannotated recipe texts and achieve an F-Score of 80 on predicting edges given gold information about the nodes; the output graphs are less detailed than ours. Özgen (2019) achieves an F-Score of 75 on the same task and presents a subtask of creating action graphs similar to ours in Section 3.4.

[5]https://github.com/coli-saar/ara

Event alignment. Our work shares an interest in modelling procedural knowledge with the detection and alignment of script events. Chambers and Jurafsky (2008, 2009) identified event types from text according to their predicate-argument structures and behavior in event chains via count-based statistics. We capture similar information in a crowdsourcing task reminiscent of Wanzare et al. (2016, 2017) to automatically align actions without all surface text (Regneri et al., 2010).

3 Parsing Recipes into Graphs

The main contribution of this paper is a corpus of *action* alignments between action *graphs* of cooking recipes. Basing our corpus on unannotated recipe texts from L'20, we are dependent on an accurate tagger and parser for pre-processing. The tagger identifies the alignable actions in a recipe, and the parser structures recipes into graph representations. For both tasks, we train neural models on the data of Yamakata et al. (2020) (Y'20); we set a new state of the art on this dataset. Finally, we distill the Y'20-style recipe graphs into more focused *action graphs* (Section 3.4) which the alignment model (Section 5) takes as input.

3.1 Recipe Graphs

The recipe graphs in the Y'20 dataset are directed acyclic graphs with node and edge labels (Fig. 2). Nodes represent entities of ten different types, such as ingredients, cooking tools, and various types of actions. Edges represent actions' predicate-argument structure, as well as part-whole relations and temporal information. The *full graph* shows the states of ingredients and tools throughout the cooking process; it can be read from top to bottom as inputs transformed into consecutive outputs.

3.2 Tagging Recipes

We split the parsing task into two steps. In the first step, we tag the tokens of a recipe with their respective node types. We implement the sequence tagger as a neural network (NN) with a two-layered BiLSTM encoder generating predictions in a CRF output layer:

$$\vec{y} = CRF(BiLSTM^{(2)}(BiLSTM^{(1)}(\vec{x}))),$$

where $\vec{y}$ are the predicted tags over the input sequence with the embedding $\vec{x}$. For comparison, Y'20 employ BERT-NER[6] for their 'recipe named entity (r-NE)' tagging task.

Contrary to common expectation, we find that this tagger performs better with ELMo embeddings (Peters et al., 2018) than with BERT embeddings (Devlin et al., 2019) (Table 1). Trained and evaluated on Y'20's 300-r corpus, our tagger performs two points better than Y'20's tagger and reaches Y'20's inter-annotator agreement.

3.3 Parsing Recipes

In the second step, we predict the edges of the graph. We use the biaffine dependency parser by Dozat and Manning (2017), implemented by Gardner et al. (2018). The model consists of a biaffine classifier upon a three-layered BiLSTM with multilingual BERT embeddings. The model takes as input the tagged recipe text generated by the tagger and generates a dependency tree over the recipe. We use the parser to generate connected recipe graphs of full recipes, so we parse the entire recipe

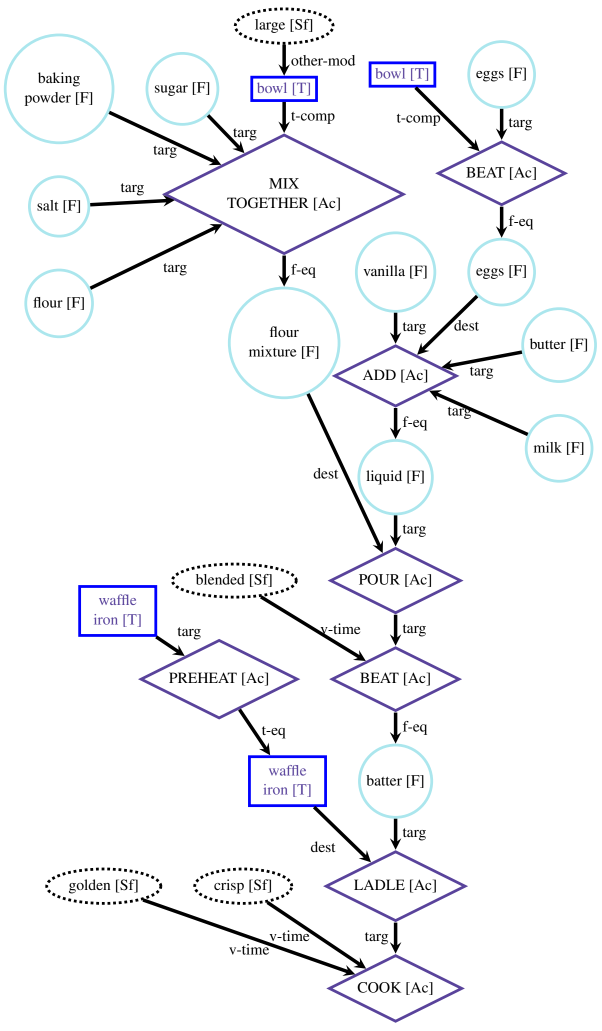


Figure 2: *Full graph* for recipe (b) (Fig. 1) in the style of Y'20. Actions are displayed as diamonds, foods as circles, tools as rectangles, and all else as ellipses.

[6] https://github.com/kyzhouhzau/BERT-NER

Model	Corpus	Embedder	Precision	Recall	F-Score
IAA	100-r by Y'20		89.9	92.2	90.5
Y'20	300-r by Y'20		86.5	88.8	87.6
Our tagger	300-r by Y'20	English ELMo	**89.9** $\pm$ **0.5**	**89.2** $\pm$ **0.4**	**89.6** $\pm$ **0.3**
Our tagger	300-r by Y'20	multilingual BERT	88.7 $\pm$ 0.4	88.4 $\pm$ 0.1	88.5 $\pm$ 0.2

Table 1: Recipe tagging performance compared to Y'20's performance and inter-annotator agreement (IAA).

as a single "sentence".

As this is a dependency *tree* parser, we remove edges from the recipe graphs in the training data to make them into trees: we preserve the edge that is listed first in the raw annotation data and ignore all other edges. This results in dependency edges that point from the inputs to the actions and from the actions to the outputs, such that the final dish only has incoming edges. We still evaluate against the complete recipe graphs in the test set.

Our parsing results are presented in Table 2. We train the parser on the English 300-r corpus by Y'20. Our parser sets a new state of the art on this corpus, with an F-Score of 78.2 on gold-tagged evaluation data. Moreover, the combined tagger and parser achieve an F-Score of 72.3 on unannotated recipe text, compared to 43.3 of Y'20's own parser on automatically tagged recipe text. See Supplementary Material A for model hyperparameters and label-specific performance on actions.

3.4 Action Graphs for Recipe Alignment

To automatically align actions across recipe graphs, we abstract the output of our parser to *action graphs*, which only retain the action nodes from the full graphs: the full graph in Fig. 2 is transformed into the action graph in Fig. 3. We accomplish this by removing all non-action nodes. The paths between action nodes in the full graph become edges in the action graph. Similar to full recipe graphs, action graphs can be read from top to bottom as a temporal and sometimes causal sequence of actions. Each interior node has parent action nodes it is dependent on, as well as child nodes conditional upon it. We utilize this information in our automatic alignment model (Section 5).

In creating our action graphs, we unify all Y'20 action types into a single type *action*: actions by chef, both continuous (e.g. "***chop***," "***add***") and discontinuous ("***sift***$_{a_i}$ *ingredients* ***together***$_{a_{i+1}}$"); actions by food ("*until the butter* ***melts***$_{a_j}$"); and actions by tool ("*until skewer* ***comes out***$_{a_k}$ *clean*").[7]

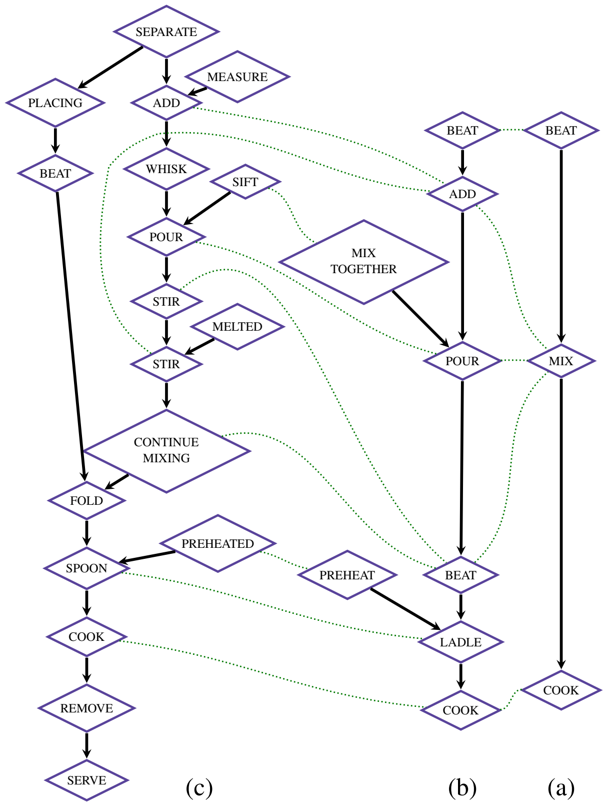


Figure 3: An example of how actions align across action graphs for the recipes in Fig. 1 (expert annotation).

While most actions have surface realizations of exactly one verb token, actions can span up to four consecutive tokens (see Table 3).

4 A Corpus of Recipe Action Alignments

We collect a corpus of crowdsourced annotations of action alignments between recipes for a subset of L'20's corpus. Our Aligned Recipe Action (ARA) corpus makes two important contributions: (i) manual alignments, as opposed to the majority of L'20's automatically aligned corpus; (ii) alignments are at the action level, as opposed to the more coarse-grained sentence level.

4.1 Data Preparation

We draw the recipes for the crowdsourcing from the training data of L'20. We manually chose 10

[7] On the specific task of action tagging, we also outperform Y'20 on gold labels (89.6 F-Score); see Table 5, Supplementary Materials.

Model	Corpus	Tag source	Precision	Recall	F-Score
IAA	100-r by Y'20	gold tags	84.4	80.4	82.3
Y'20	300-r by Y'20	gold tags	73.7	68.6	71.1
Our parser	300-r by Y'20	gold tags	80.4 ± 0.0	76.1 ± 0.0	**78.2 ± 0.0**
Y'20	300-r by Y'20	Y'20 tagger	51.1	37.7	43.3
Our parser	300-r by Y'20	our ELMo tagger	74.4 ± 0.5	70.4 ± 1.0	**72.3 ± 0.8**

Table 2: Recipe parsing performance compared to Y'20 and inter-annotator agreement (IAA).

	count	mean
dishes	10	
recipes	110	
recipes per dish	11	
total action pairs annotated	1592	
annotations per source action	at least 3	
actions per recipe	3-37	15.1
actions per source recipe	4-37	15.9
actions per sentence	0-9	1.7
tokens per action	1-4	1.2

Table 3: Makeup of our ARA annotated corpus.

dishes from the subset of dishes with exactly 11 pairwise-matched recipes.[8] We chose the 10 dishes and corresponding recipes for their variation and to ensure our work generalizes to new cuisines and recipes: they span different cuisines (Italian, Chinese, Indian, American, German) and dish types (appetizer, side, main, dessert). A full list of dishes we annotate is in Supplementary Material B. In all recipes, we detect actions with our tagger (Section 3.3) using ELMo embeddings and trained on the English data of Y'20.

For each dish, we define ten pairs of recipes such that one recipe is the *source recipe* of an alignment to the next shorter *target recipe* of the same dish. We select these pairings by measuring the length of recipes in number of actions. Using this methodology, all recipes except for the longest and shortest recipes of each dish are annotated once as source recipe and once as target recipe, respectively. The pairing procedure is motivated by two rationales: **1. Long-to-Short.** In order to limit annotator disagreement from 1:n alignments in our data, we always present the longer recipe as the source recipe as we expect 1:n alignments to be not as common if the source recipe is longer than the target recipe. **2. Transitive closure.** An alignment from a source recipe to a *secondary target recipe* can be approximated by the alignments between (i) the source recipe and the target recipe and (ii) the alignments between the target recipe and the secondary target recipe.

[8]Dishes in L'20 have between 3 and 100 recipes.

4.2 Data Collection

We obtain the alignments in the corpus in a multi-step process: For every source action a, we initially ask three crowd-workers to vote for the correct target action. If we find a unique plurality vote for a target action b, a is aligned to b. If there is no agreement between the crowd-workers, we iteratively ask more crowd-workers to select a target action, until we have a unique plurality vote for the target. This approach optimizes resource spending as it takes into account that some alignments are very easy (needing few votes) whereas others are harder, resulting in more noise in the votes and therefore needing more votes to obtain a reliable annotation. In extreme cases, where we did not obtain a plurality even after five or six votes, we deferred to an expert annotator: 190 out of 1592 source actions did not receive a unique plurality target; these were adjudicated by two of the authors.

Crowd-sourcing setup. We implemented our crowdsourcing experiment with *Lingoturk* (Pusse et al., 2016). Participants were hired through Prolific[9]. In total we hired 250 participants, each of whom answered on average 32 questions. Each participant was paid 1.47 GBP; the average hourly payment was 8.82 GBP.

A participant's task was to align the set of actions in a source recipe to its target recipe. Importantly, this task is not a simple verb matching task. Participants were instructed to read both recipes in their entirety and were then presented with two source actions at a time in the order in which they appear in the recipe.[10] For each source action, the participant was asked to choose one action from the target recipe that it *"best corresponds to"*, or *"None of these"*; the entire set of actions in the target recipe was available for selection. Both recipes were dis-

[9]https://www.prolific.co/; as part of screening for the platform, all participants were native English speakers and had normal vision (in particular, they distinguish colors correctly).

[10]If a source recipe has an odd number of actions, the last set of questions for the recipe pair has only one question.

played at all times with all actions bolded and in unique colors for ease of identification. Each participant did this for two recipe pairs of differing length and for different dishes, such that the total length of each experiment was roughly comparable. See Supplementary Material B for more details on the annotation setup.

Annotator agreement. To quantitatively assess agreement between our annotators, we compute how often the votes by the participants agreed with the alignment we obtained (i.e., the probability of a participant's vote to align with the plurality). A distribution of the support behind the plurality vote is seen in Fig. 4. While some questions are easy and receive 100% agreement, many of them only receive a plurality from 50% of the answers. For some questions, disagreement between annotators was so high that the most-chosen answer was only chosen by 20% of the annotators. In such cases, we collected a high number of votes from different participants to obtain a plurality.

Overall, 69.3% of the target selections by our annotators agreed with the annotation in the dataset (selected by plurality vote). This measure is not an upper bound for system performance because incorrect annotations by an annotator are only reflected in the dataset if by chance several annotators chose the same incorrect target. Otherwise, the incorrect decision by an annotator will be remedied by the requirement of having a plurality vote from at least three different annotators. It is also not an upper bound for human performance because some annotators were more reliable than others and we report the average over all annotators.

IAA measures. Due to the data collection design (a large group of people answering question with variable answer sets and a skewed but unknown prior distribution over the answers), common inter-annotator metrics such as Krippendorff's α or Cohen's κ are not applicable: These measures require either a fixed set of possible answers for all questions or knowledge of the prior distribution of the answers, or a fixed number of annotators (or a combination of these requirements). Thus, computing such a reliability metric would require making (incorrect) assumptions of the data and render the resulting number uninterpretable.

Human performance and baseline. To obtain a human accuracy score, a subset of the authors manually annotated ten recipe pairs and obtained a 79% accuracy with respect to the gold standard. The majority baseline for the annotations is 31.3% (always choosing "None").

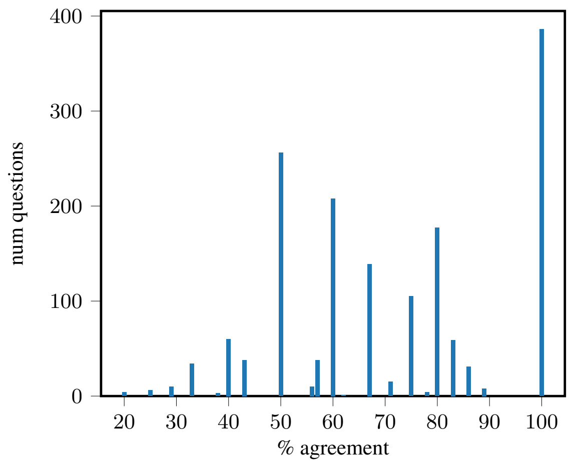


Figure 4: For each question we computed how many answers agreed with the plurality vote. The plot shows the distribution of questions over this agreement score.

Corpus statistics. Details about ARA, our full annotated corpus, are in Table 3. On average, 16 action pairs are collected for each recipe pair. Notably, this average number of actions per recipe (15.1) is almost double the average number of 8 *sentences* per dish in L'20, further motivating our task of annotating fine-grained actions to collect more detailed recipe information. Of particular interest for our task, 70% of the recipes have re-occurring action words (e.g. "*add*") that do not signal re-occurring actions based on the surrounding recipe context. There are 366 unique actions distributed over 1659 action instances. The majority of recipes have two repeated verbs; the highest repetition is 5 identical verbs in one recipe, which occurs in two recipes.

4.3 Disagreement Analysis

Qualitative analysis reveals that inter-annotator disagreement has several sources (actions bolded). We expect 10% of actions to be mistagged (Table 1); typos in the original L'20 corpus are a special case of this ("*from*" misspelled as "***form***"). Several recipes consist of more than 20 actions and contain multiple identical actions in their surface form (see paragraph *Corpus statistics*). Though these actions appear in different colors on the interface, it is easy to forget which color corresponds to which action and subsequently misalign it. This can also cause annotators to simply choose the most similar superficial action without considering context: for example, "***mix***" aligned to "***mash***" regardless of surrounding ingredients or stage of recipe.

Even given the full recipe context, linguistic peculiarities of recipe text make deciding whether actions between two recipes correspond difficult. We discuss several cases of this and quantify their frequency based on 100 of the questions that initially received no majority (percentages in parentheses). Notably, some of these categories overlap.[11]

One-to-many alignments (6%). Our Long-to-Short principle (Section 4.1) pays off, although we still find cases where one source action can align to two distinct target actions. We see this result with infinitive actions: "***set aside to cool***$_{a_i}$," as one action, can be aligned to either action in "***transfer***$_{a_j}$ *to* ***cool***$_{a_{j+1}}$." We also see this with prepositional result phrases: "***stir***$_{a_k}$ *until* ***combined***$_{a_{k+1}}$." Additional manual analysis shows missed cases are minimal enough to not impact downstream performance.

Many-to-one alignments (26%). The reverse case of one-to-many alignments causes disagreement in whether individual actions that comprise a more complex action should be aligned. This happens frequently in baking recipes, where recipes often differ in when and how they combine wet and dry ingredients. For example in Fig. 3, "***Add***" adds the *milk*, *butter*, and *vanilla* to eggs in (b), while the same action in (c) adds egg *yolks* and *vanilla essence* to milk. Though these high-level actions may sequentially align across recipes, the different ingredients that act as arguments can impact whether they are judged to correspond. Conflated actions contribute to this phenomenon: "***Add*** *dry ingredients* $\underline{\textit{alternately}}$ *with wet ingredients*"; "***Let cool*** *in pan* $\underline{\textit{or}}$ *on wire rack.*"

Implicit actions (24%). Implicit actions come in many forms in recipe texts and may cause confusion as to whether an action should be aligned to the implicit step or not aligned at all. We see this with nouns that imply actions: "***return*** *to* ***continue cooking***" versus "***return*** *to cooker.*" We also see actions that must be inferred from their surrounding actions: "***Spoon*** *onto baking tray.* ***Take out*** *after 8 minutes.*" to connote "***Bake.***" Finally, ingredients themselves can imply actions for the chef ("***crushed*** *garlic*"), or not ("***canned*** *tomatoes*").

Let and light verbs (22%). Light causative verbs such as *let* and *allow* are frequent sources of disagreement. Part of this disagreement stems from our action tagger: while sequences such as "***let rest***$_{a_i}$" and "***allow to stand***$_{a_j}$" are tagged as one action, "***allow***$_{a_k}$ *to* ***return***$_{a_{k+1}}$" is tagged as two. If a noun intervenes, such as "***let***$_{a_l}$ *dough* ***rest***$_{a_{l+1}}$," one action can become two.[12] Disagreement can then arise as to which of the two actions carries the semantic weight and should be aligned. By-phrases also introduce confusing subevent structure: "***create*** *layers by* ***putting*** *pasta...*" versus "***layer*** *pasta.*"

Negative and conditional events (2%). Some recipes include negative events: "***Avoid*** *over* ***mixing***" or "*without* ***kneading***." Conditional events may also cause disagreement based on whether an annotator judges them as real or not: "*If* ***refrigerated****...*"; "*When you are ready to* ***bake****...*"

No best alignment (17%). Though some actions have no best alignment, the crowdsourcing task may bias participants to choose an answer.

The sources of disagreement we find in ARA illustrate the need for analyzing recipe text at the fine-grained level of actions. In refining the sentence-level alignments of L'20 to the cooking action level, we find that judging when and how specific actions "correspond" is a complex task that requires intricate knowledge of the cooking domain.

5 Automatic Alignment of Actions

We develop two automatic alignment models that mimic our crowdsourcing task; these models align action nodes between our parsed action graphs (Section 3.4). We evaluate both models on the crowdsourced alignment dataset (Section 4): one using features solely from the actions to be aligned, and one incorporating parent and child action nodes into the alignment decision.

5.1 Alignment Model Structure

We treat alignment as an assignment problem: for each action in the source recipe R_1, we independently decide which action in the target recipe R_2 it should align to. Alternatively, a source action may have no adequate alignment and be unaligned. We

[11]For example, 'let' and light verbs can cause 1:n alignments. The 3% we do not report in the text are due to typos (2) and a noun misclassified by our tagger as an action (1).

[12]This is peculiarity of English verb constructions. In pilot annotation on a German corpus, this disagreement does not occur, as all objects stand alone from complex verb constructions and action sequences are not interrupted. Yamakata et al. (2020) notice this difficulty with English, too, and distinguish 'actions by chef' and 'actions by food,' such that "***let*** *dough* ***rest***" consists of one action by chef and one action by food (Table 6). The authors still notice inconsistency in how annotators evaluate these sequences: while Ac is annotated with 92.7% accuracy, other actions show accuracy between 17% and 46%.

use a two-block architecture for this classification task.

Encoder The Encoder generates an encoding vector $enc(i)$ for each action $a(i)$ of a recipe. We re-tokenize each recipe with the BERT tokenizer and obtain token embeddings $emb(j)$ from BERT. For each action $a(i)$ we track the list of tokens $t(a(i))$ that correspond to this action. We then obtain $enc(i)$ for the two versions of our alignment model in the following way.

In the *base* model, we run an LSTM over the embeddings to generate the representations for each action, such as:

$$enc\_b(i) = LSTM^{seq}([emb(j) \,|\, j \in t(a(i))])$$
$$enc(i) = enc\_b(i)$$

The *extended* model incorporates structural information about the recipe graph. We extract the child and parent action nodes for each source action. We then combine the *base* encoding of each action with the *base* encodings of its children and parents. As each action can have multiple parents and children, we run an $LSTM^p$ over the parent base encodings for all parents $p(a(i))$ for an action $a(i)$ and an $LSTM^c$ over the child base encodings for all child nodes $c(a(i))$. We obtain $enc\_ext(i)$ by concatenating $enc\_b(i)$ with the outputs of these two LSTMs:

$$enc\_p(i) = LSTM^p([enc\_b(p) \,|\, p \in p(a(i))])$$
$$enc\_c(i) = LSTM^c([enc\_b(c) \,|\, c \in c(a(i))])$$
$$enc\_ext(i) = [enc\_b(i); enc\_c(i); enc\_p(i)]$$
$$enc(i) = enc\_ext(i)$$

Whenever the parent or child list is empty, we replace the LSTM output with a trained embedding representing the empty child / parent list.

Scorer The Scorer predicts the alignment target for an action using one-versus-all classification. Given a source action $a_1(i) \in R_1$, we compute scores $s(enc(a_1(i)), enc(a_2(j)))$ for the alignment to every target action (including the "None" target) $a_2(j) \in R_2 \cup none$. For both the *base* and the *extended* model, the encoding of the "None" alignment target is a trained vector of the same size as the action encoding. We compute $s(enc(a_1(i)), enc(a_2(j)))$ using a multi-layer perceptron with two hidden layers and the element-wise product $enc(a_1(i)) \odot enc(a_2(j))$ as input.

Model Name	Accuracy
Human Upper Bound	79.0
Sequential Order	16.5
Cosine Similarity	41.5
Common Action Pairs	52.1
Our Alignment Model (*base*)	66.3
Our Alignment Model (*extended*)	**72.4**

Table 4: Performance comparison of the three baseline models, alignment model (*base*), and alignment model (*extended*) on the Action Alignment Corpus.

Training and evaluation We train both models using cross-entropy loss, with updates only on incorrect predictions to avoid overfitting. We use 10-fold cross validation on the ten different dishes, with one dish serving as test dish, such that the aligner is always evaluated on an unknown domain.

5.2 Experiment Results

We implement three baselines for comparison: (i) sequential ordering of alignments, such that $a_1(i)$ from recipe R_1 will be aligned to $a_2(i)$ from recipe R_2; (ii) cosine similarity scores between the BERT embeddings of action pairs as the scorer; and (iii) common action pair frequencies with 10-fold cross-validation (Table 4). We further compare our model to the human upper bound for alignment accuracy as discussed in Section 4.2. Our *base* alignment model outperforms all the baselines, but we observe a substantial gain in accuracy in the *extended* alignment model, illustrating the importance of structural information about the recipe in aligning actions. The extended model still does not reach human performance, illustrating the difficulty of the task.

The *extended* alignment model achieves an accuracy of 69.8% on aligning actions of the longer recipe to actions of the shorter recipe. This suggests that our Long-to-Short approach to data collection yields valid data in both directions.

As a point of comparison, L'20 achieve an F-Score of 54.6 for text-to-text alignments and 70.3 for text-to-video alignment at the sentence level, suggesting that alignments of actions may be harder to annotate than alignments of entire sentences, but easier to align automatically.

6 Conclusion & Future Work

In this paper, we have made two contributions: (i) ARA, a novel corpus of human-annotated alignments between corresponding cooking actions, and (ii) a first neural model to align actions across recipe graphs. We find that incorporating structural information about recipes improves the accuracy of the neural model, highlighting the usefulness of recipe graphs and of recipe parsers.

Compared to previous work, our corpus and model represent alignments at the level of individual actions and not of entire sentences. In refining the sentence-level alignments of L'20 to the cooking action level, we find that judging when and how specific actions "correspond" is a complex task that requires intricate knowledge of the cooking domain. Alternatively, the complexity of recipe interpretation can be framed as a matter of recognizing nuances in how meaning is construed in recipe text given the genre and its preferred syntactic constructions (Langacker, 1993; Trott et al., 2020).

Looking ahead, our work lays a foundation for research which automatically aggregates multiple recipe graphs for the same dish, identifying common and distinct parts of the different recipes. This opens up a variety of applications in the cooking domain, including dialogue systems which can explain a recipe at different levels of abstraction.

Acknowledgments

Partially funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – Project-ID 232722074 – SFB 1102.

References

Nathanael Chambers and Dan Jurafsky. 2008. Unsupervised learning of narrative event chains. In *Proceedings of ACL-08: HLT*, pages 789–797, Columbus, Ohio. Association for Computational Linguistics.

Nathanael Chambers and Dan Jurafsky. 2009. Unsupervised learning of narrative schemas and their participants. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 602–610, Suntec, Singapore. Association for Computational Linguistics.

Minsuk Chang, Leonore V. Guillain, Hyeungshik Jung, Vivian M. Hare, Juho Kim, and Maneesh Agrawala. 2018. Recipescape: An interactive tool for analyzing cooking instructions at scale. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems, CHI 2018, Montreal, QC, Canada, April 21-26, 2018*, page 451. ACM.

Yuzhe Chen. 2017. A statistical machine learning approach to generating graph structures from food recipes. Master's thesis, Brandeis University.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Timothy Dozat and Christopher D. Manning. 2017. Deep biaffine attention for neural dependency parsing. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew Peters, Michael Schmitz, and Luke Zettlemoyer. 2018. AllenNLP: A deep semantic natural language processing platform. In *Proceedings of Workshop for NLP Open Source Software (NLP-OSS)*, pages 1–6, Melbourne, Australia. Association for Computational Linguistics.

Jermsak Jermsurawong and Nizar Habash. 2015. Predicting the structure of cooking recipes. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 781–786, Lisbon, Portugal. Association for Computational Linguistics.

Chloe Kiddon, Ganesa Thandavam Ponnuraj, Luke Zettlemoyer, and Yejin Choi. 2015. Mise en place: Unsupervised interpretation of instructional recipes. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Ronald W. Langacker. 1993. Universals of construal. In *Annual Meeting of the Berkeley Linguistics Society*, volume 19, pages 447–463.

Angela Lin, Sudha Rao, Asli Celikyilmaz, Elnaz Nouri, Chris Brockett, Debadeepta Dey, and Bill Dolan. 2020. A recipe for creating multimodal aligned datasets for sequential tasks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4871–4884, Online. Association for Computational Linguistics.

Shinsuke Mori, Hirokuni Maeta, Tetsuro Sasada, Koichiro Yoshino, Atsushi Hashimoto, Takuya Funatomi, and Yoko Yamakata. 2014. Flowgraph2text: Automatic sentence skeleton compilation for procedural text generation. In *Proceedings of the 8th International Conference on Natural Language Generation (INLG)*.

Iftekhar Naim, Young Chol Song, Qiguang Liu, Henry A. Kautz, Jiebo Luo, and Daniel Gildea. 2014. Unsupervised alignment of natural language instructions with video segments. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27 -31, 2014, Québec City, Québec, Canada*, pages 1558–1564. AAAI Press.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Florian Pusse, Asad Sayeed, and Vera Demberg. 2016. LingoTurk: managing crowdsourced tasks for psycholinguistics. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*, pages 57–61, San Diego, California. Association for Computational Linguistics.

Michaela Regneri, Alexander Koller, and Manfred Pinkal. 2010. Learning script knowledge with web experiments. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 979–988, Uppsala, Sweden. Association for Computational Linguistics.

Sean Trott, Tiago Timponi Torrent, Nancy Chang, and Nathan Schneider. 2020. (re)construing meaning in NLP. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5170–5184, Online. Association for Computational Linguistics.

Lilian Wanzare, Alessandra Zarcone, Stefan Thater, and Manfred Pinkal. 2017. Inducing script structure from crowdsourced event descriptions via semi-supervised clustering. In *Proceedings of the 2nd Workshop on Linking Models of Lexical, Sentential and Discourse-level Semantics*, pages 1–11, Valencia, Spain. Association for Computational Linguistics.

Lilian D. A. Wanzare, Alessandra Zarcone, Stefan Thater, and Manfred Pinkal. 2016. A crowdsourced database of event sequence descriptions for the acquisition of high-quality script knowledge. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 3494–3501, Portorož, Slovenia. European Language Resources Association (ELRA).

Yoko Yamakata, Shinji Imahori, Hirokuni Maeta, and Shinsuke Mori. 2016. A method for extracting major workflow composed of ingredients, tools, and actions from cooking procedural text. In *2016 IEEE International Conference on Multimedia & Expo Workshops (ICMEW)*.

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.

Mehmet Özgen. 2019. Tagging and action graph generation for recipes. Msc thesis, Hacettepe University.

A Tagger and Parser Evaluation

A.1 Methodology

All our measures are computed as averages of the mean values of four individually trained instances of our model and are reported together with the respective standard deviations. For the evaluation of parsing automatically tagged recipe text (Table 2 in the paper), the recipes are tagged by four individually trained instances of our tagger (with ELMo embeddings) before they are parsed by four individually trained instances of the parser.

While Y'20 cross-validate over the whole 300-r corpus, we train on a subset of 240 recipes and evaluate on a subset of 30 recipes. Each subset is randomly chosen from the full corpus such that the proportions of 100-r to 200-r are preserved as 1:2 .

A.2 Tagger

For hyper-parameters, see Table 5. The hyper-parameters were fine-tuned separately for the pre-trained embeddings (ELMo and BERT, respectively).

Hyper-parameter	ELMo	BERT
Hidden size BiLSTM	50	200
Layers BiLSTM	2	2
Dropout BiLSTM	0.5	0.5
Dropout CRF	0.5	0.5
Regularization method	L2 ($\alpha = 0.5$)	L2 ($\alpha = 0.1$)
Optimization method	Adam	Adam
Learning rate	0.0075	0.001
Gradient norm	10.0	10.0
Training epochs	50	50

Table 5: Hyper-parameters for the tagger after fine-tuning on the German data set. The number of epochs is an observation.

For reference, we display an overview over the labels defined by Y'20 in Table 6. The label-specific performance for action sequences is itemized in detail in Table 7.

A.3 Parser

We did not perform any fine-tuning on the parser. The hyper-parameters are reported in Table 8.

Hyper-parameter	Value
Tag embedding dim	100
Hidden size BiLSTM	400
Layers BiLSTM	3
Input dropout	0.3
BiLSTM dropout	0.3
Classifier dropout	0.3
Optimization method	Dense Sparse Adam (betas (0.9, 0.9))
Gradient norm	5
Training epochs	20 ± 10

Table 8: Hyper-parameters for the parser - no fine-tuning performed. The number of training epochs is an observation.

Label	Meaning	Explanation
F	Food	Eatable; also intermediate products
T	Tool	Knife, container, etc.
D	Duration	Duration of cooking
Q	Quantity	Quantity of food
Ac	Action by chef	Verb representing a chef's action
Ac2	Discontinuous Ac (English only)	Second, non-contiguous part of a single action by chef
Af	Action by food	Verb representing action of a food
At	Action by tool (English only)	Verb representing a tool's action
Sf	Food state	Food's initial or intermediate state
St	Tool state	Tool's initial or intermediate state

Table 6: Y'20 labels for the tagging task.

Model	Data	Label	# Instances	Precision	Recall	F-Score
Y'20	Y'20	{*Ac, Ac2, Af, At*}		88.7	89.3	89.0
Our model ('21)	Y'20	{*Ac, Ac2, Af, At*}		92.0 ± 1.6	88.4 ± 1.8	90.1 ± 1.7
Y'20	Y'20	*Ac*	4977	92.3	93.1	92.7
Our model ('21)	Y'20	*Ac*	483	94.6 ± 1.1	91.7 ± 1.6	93.1 ± 1.2
Y'20	Y'20	*Ac2*	178	43.8	46.8	45.3
Our model ('21)	Y'20	*Ac2*	16	69.2 ± 9.0	68.8 ± 10.2	68.1 ± 3.2
Y'20	Y'20	*Af*	255	51.4	50.4	50.9
Our model ('21)	Y'20	*Af*	32	64.8 ±9.9	47.7 ± 7.8	54.9 ± 8.7
Y'20	Y'20	*At*	15	60.0	10.0	17.1
Our model ('21)	Y'20	*At*	2	100 ± 0.0	100 ± 0.0	100 ± 0.0

Table 7: Label-specific performance for action sequences; comparison of our model and that of Y'20. The values in lines 1 and 2 are micro-averaged over the four individual action labels.

The set of edges labels as determined by Y'20 are displayed in Table 9.

B Crowdsourcing

B.1 Dishes Annotated

(1) Baked Ziti; (2) Blueberry Banana Bread; (3) Cauliflower Mash; (4) Chewy Chocolate Chip Cookies; (5) Garam Masala; (6) Homemade Pizza Dough; (7) Orange Chicken; (8) Pumpkin Chocolate Chip Bread; (9) Slow Cooker Chicken Tortilla Soup; (10) Waffles.

B.2 Experiment Platform

Figure 5 gives a screenshot of the instruction page, and Figure 6 gives a screenshot of the question page.

C Alignment Model Training Details

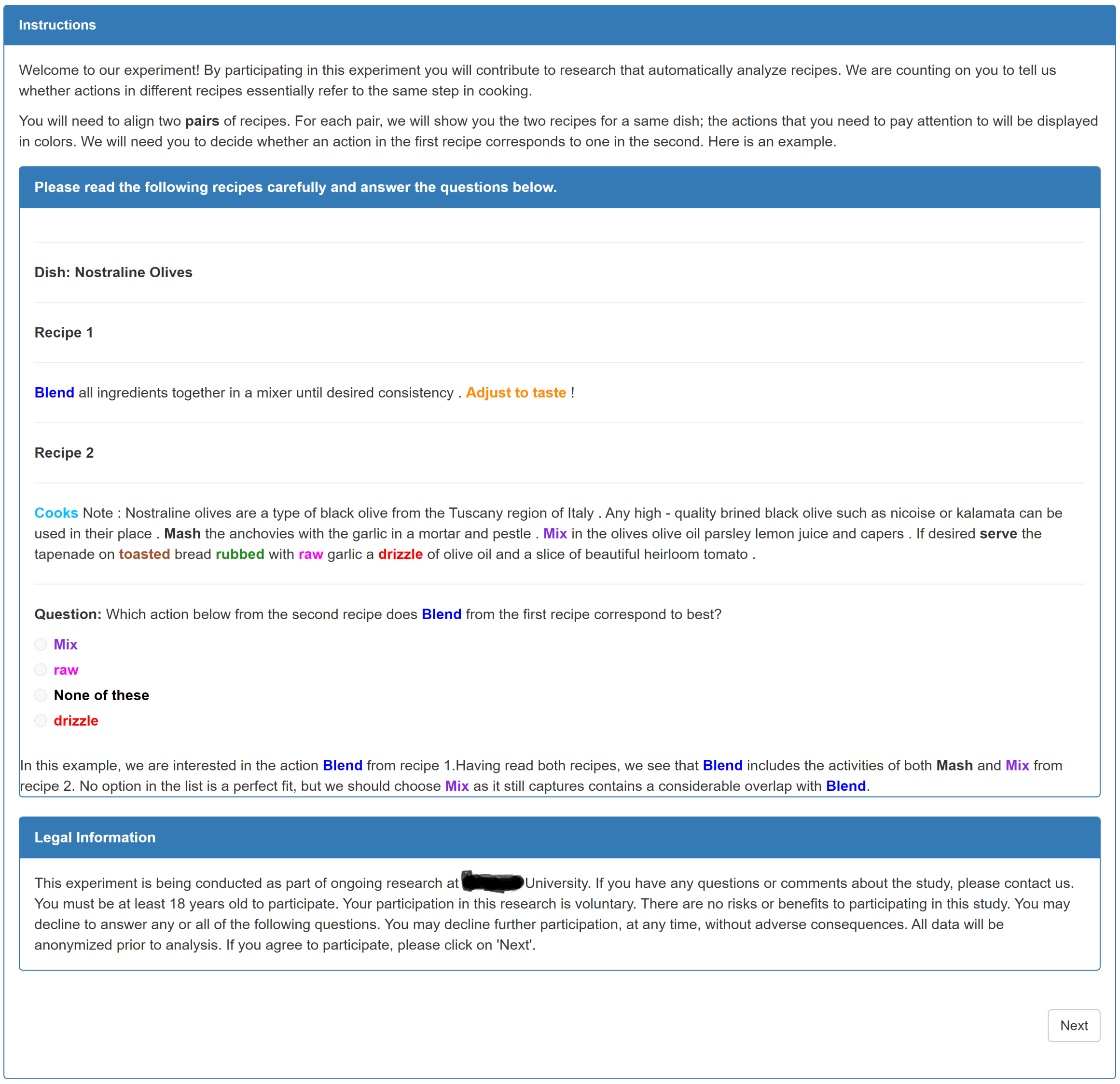
In order to re-tokenize and generate token embeddings for the recipes, we utilized the *bert-uncased-base* BERT model with embedding dimension of 768. We trained both alignment models (*base* and *extended*) for 10-folds with 1 dish in the test set and 9 dishes for training. In each fold, the model runs for 40 epochs with a train/dev split of 8/1 dishes respectively. We use Adam as the optimizer with a 0.0001 learning rate and Cross Entropy loss as the loss function during training. For aligner's hyper-parameters, see Table 10.

Hyper-parameter	Value
BERT embedding dim	768
$LSTM^{(seq)}$ hidden dim	768
$LSTM^{(p)}$ hidden dim	768
$LSTM^{(c)}$ hidden dim	768
MLP layer 1 output dim	128
MLP layer 2 output dim	32
MLP layer 3 output dim	1

Table 10: Hyper-parameters for the Alignment model.

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

Table 9: Y'20 edge labels for the parsing task.



Instructions

Welcome to our experiment! By participating in this experiment you will contribute to research that automatically analyze recipes. We are counting on you to tell us whether actions in different recipes essentially refer to the same step in cooking.

You will need to align two **pairs** of recipes. For each pair, we will show you the two recipes for a same dish; the actions that you need to pay attention to will be displayed in colors. We will need you to decide whether an action in the first recipe corresponds to one in the second. Here is an example.

Please read the following recipes carefully and answer the questions below.

Dish: Nostraline Olives

Recipe 1

Blend all ingredients together in a mixer until desired consistency . **Adjust to taste** !

Recipe 2

Cooks Note : Nostraline olives are a type of black olive from the Tuscany region of Italy . Any high - quality brined black olive such as nicoise or kalamata can be used in their place . **Mash** the anchovies with the garlic in a mortar and pestle . **Mix** in the olives olive oil parsley lemon juice and capers . If desired **serve** the tapenade on **toasted** bread **rubbed** with **raw** garlic a **drizzle** of olive oil and a slice of beautiful heirloom tomato .

Question: Which action below from the second recipe does **Blend** from the first recipe correspond to best?

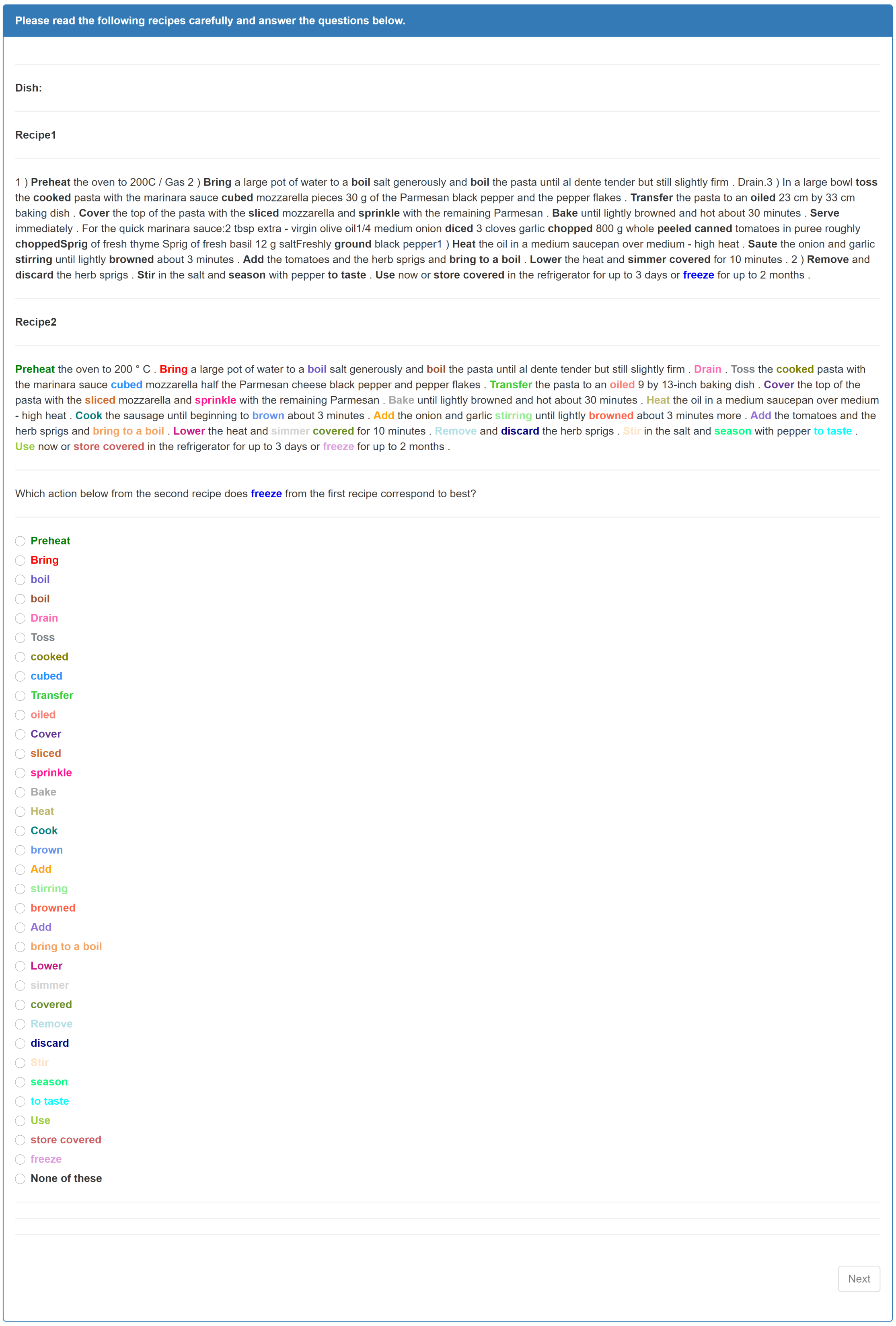
- ○ **Mix**
- ○ **raw**
- ○ **None of these**
- ○ **drizzle**

In this example, we are interested in the action **Blend** from recipe 1.Having read both recipes, we see that **Blend** includes the activities of both **Mash** and **Mix** from recipe 2. No option in the list is a perfect fit, but we should choose **Mix** as it still captures contains a considerable overlap with **Blend**.

Legal Information

This experiment is being conducted as part of ongoing research at [redacted] University. If you have any questions or comments about the study, please contact us. You must be at least 18 years old to participate. Your participation in this research is voluntary. There are no risks or benefits to participating in this study. You may decline to answer any or all of the following questions. You may decline further participation, at any time, without adverse consequences. All data will be anonymized prior to analysis. If you agree to participate, please click on 'Next'.

Figure 5: A screen shot of the instruction page.


Please read the following recipes carefully and answer the questions below.

Dish:

Recipe1

1) **Preheat** the oven to 200C / Gas 2) **Bring** a large pot of water to a **boil** salt generously and **boil** the pasta until al dente tender but still slightly firm . Drain.3) In a large bowl **toss** the **cooked** pasta with the marinara sauce **cubed** mozzarella pieces 30 g of the Parmesan black pepper and the pepper flakes . **Transfer** the pasta to an **oiled** 23 cm by 33 cm baking dish . **Cover** the top of the pasta with the **sliced** mozzarella and **sprinkle** with the remaining Parmesan . **Bake** until lightly browned and hot about 30 minutes . **Serve** immediately . For the quick marinara sauce:2 tbsp extra - virgin olive oil1/4 medium onion **diced** 3 cloves garlic **chopped** 800 g whole **peeled canned** tomatoes in puree roughly **choppedSprig** of fresh thyme Sprig of fresh basil 12 g saltFreshly **ground** black pepper1) **Heat** the oil in a medium saucepan over medium - high heat . **Saute** the onion and garlic **stirring** until lightly **browned** about 3 minutes . **Add** the tomatoes and the herb sprigs and **bring to a boil** . **Lower** the heat and **simmer covered** for 10 minutes . 2) **Remove** and **discard** the herb sprigs . **Stir** in the salt and **season** with pepper **to taste** . **Use** now or **store covered** in the refrigerator for up to 3 days or **freeze** for up to 2 months .

Recipe2

Preheat the oven to 200 ° C . **Bring** a large pot of water to a **boil** salt generously and **boil** the pasta until al dente tender but still slightly firm . **Drain** . **Toss** the **cooked** pasta with the marinara sauce **cubed** mozzarella half the Parmesan cheese black pepper and pepper flakes . **Transfer** the pasta to an **oiled** 9 by 13-inch baking dish . **Cover** the top of the pasta with the **sliced** mozzarella and **sprinkle** with the remaining Parmesan . **Bake** until lightly browned and hot about 30 minutes . **Heat** the oil in a medium saucepan over medium - high heat . **Cook** the sausage until beginning to **brown** about 3 minutes . **Add** the onion and garlic **stirring** until lightly **browned** about 3 minutes more . **Add** the tomatoes and the herb sprigs and **bring to a boil** . **Lower** the heat and **simmer** **covered** for 10 minutes . **Remove** and **discard** the herb sprigs . **Stir** in the salt and **season** with pepper **to taste** . **Use** now or **store covered** in the refrigerator for up to 3 days or **freeze** for up to 2 months .

Which action below from the second recipe does **freeze** from the first recipe correspond to best?

- ○ **Preheat**
- ○ **Bring**
- ○ **boil**
- ○ **boil**
- ○ **Drain**
- ○ **Toss**
- ○ **cooked**
- ○ **cubed**
- ○ **Transfer**
- ○ **oiled**
- ○ **Cover**
- ○ **sliced**
- ○ **sprinkle**
- ○ **Bake**
- ○ **Heat**
- ○ **Cook**
- ○ **brown**
- ○ **Add**
- ○ **stirring**
- ○ **browned**
- ○ **Add**
- ○ **bring to a boil**
- ○ **Lower**
- ○ **simmer**
- ○ **covered**
- ○ **Remove**
- ○ **discard**
- ○ **Stir**
- ○ **season**
- ○ **to taste**
- ○ **Use**
- ○ **store covered**
- ○ **freeze**
- ○ **None of these**

Figure 6: A screen shot of the experiment interface.