Mise en Place: Unsupervised Interpretation of Instructional Recipes

Chloé Kiddon[†], **Ganesa Thandavam Ponnuraj**[‡], **Luke Zettlemoyer**[†], **and Yejin Choi**[†]

[†] Computer Science & Engineering, University of Washington, Seattle, WA
{chloe, lsz, yejin}@cs.washington.edu

[‡] Department of Computer Science, Stony Brook University, Stony Brook, NY
gthandavam@gmail.com

Abstract

We present an unsupervised hard EM approach to automatically mapping instructional recipes to action graphs, which define what actions should be performed on which objects and in what order. Recovering such structures can be challenging, due to unique properties of procedural language where, for example, verbal arguments are commonly elided when they can be inferred from context and disambiguation often requires world knowledge. Our probabilistic model incorporates aspects of procedural semantics and world knowledge, such as likely locations and selectional preferences for different actions. Experiments with cooking recipes demonstrate the ability to recover high quality action graphs, outperforming a strong sequential baseline by 8 points in F1, while also discovering general-purpose knowledge about cooking.

1 Introduction

Instructional language describes how to achieve a wide variety of goals, from traveling successfully to a desired location to cooking a particular dish for dinner. Despite the fact that such language is important to our everyday lives, there has been relatively little effort to design algorithms that can automatically convert it into an actionable form. Existing methods typically assume labeled training data (Lau et al., 2009; Maeta et al., 2015) or access to a physical simulator that can be used to test understanding of the instructions (Branavan et al., 2009; Chen and Mooney, 2011; Bollini et al., 2013). In this paper, we present the first approach for unsupervised learning to interpret instructional recipes using text alone, with application to cooking recipes.

Given a recipe, our task is to segment it into text spans that describe individual actions and construct an *action graph* whose nodes represent actions and edges represent the flow of arguments across actions, for example as seen in Fig. 1. This task poses unique challenges for semantic analysis. First, null arguments and ellipses are extremely common (Zwicky, 1988). For example, sentences such as "Bake for 50 minutes" do not explicitly mention what to bake or where. Second, we must reason about how properties of the physical objects are changed by the described actions, for example to correctly resolve what the phrase "the wet mixture" refers to in a baking recipe. Although linguistic context is important to resolving both of these challenges, more crucial is common sense knowledge about how the world works, including what types of things are typically baked or what ingredients could be referred to as "wet."[1]

These challenges seemingly present a chicken and egg problem — if we had a high quality semantic analyzer for instructions we could learn common sense knowledge simply by reading large bodies of text. However, correctly understanding instructions requires reasoning with exactly this desired knowledge. We show that this conflict can be resolved with an unsupervised learning approach, where we design models to learn various aspects of procedural knowledge and then fit them to unannotated instructional text. Cooking recipes are an ideal domain to study these two challenges simultaneously, as vast amounts of recipes are available online today, with significant redundancy in their coverage that can help bootstrap the overall learning process. For example, there are over 400 variations on "macaroni and cheese" recipes on allrecipes.com, from "chipotle

[1]The goal of representing common sense world knowledge about actions and objects also drives theories of frame semantics (Fillmore, 1982) and script knowledge (Schank and Abelson, 1977). However, our focus is on inducing this style of knowledge automatically from procedural texts.

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 982–992,
Lisbon, Portugal, 17-21 September 2015.

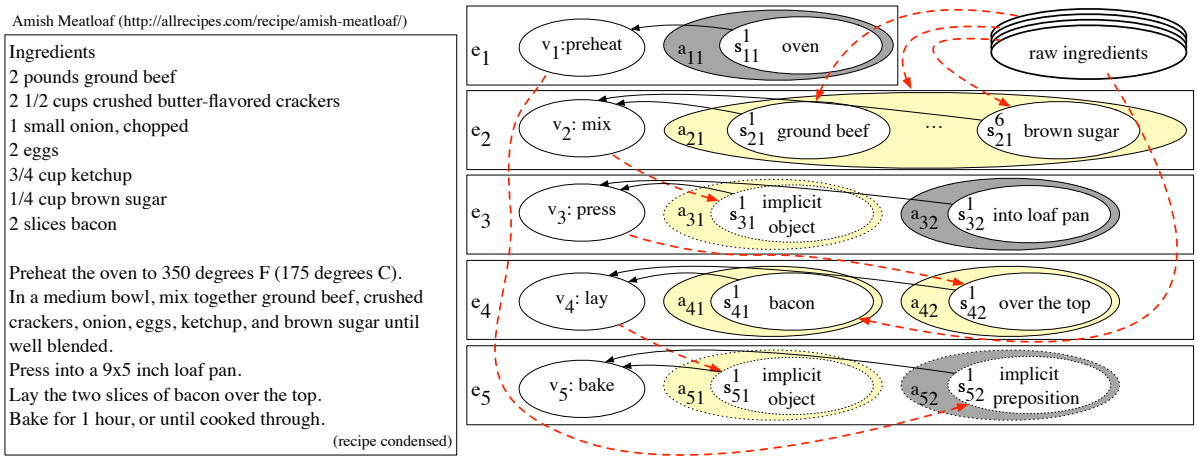

Figure 1: An input recipe (left) and a partial corresponding output action graph (right). Each rectangle (e_i) represents an action. The leftmost oval (v_i) in each action is the action's verb and the following ovals (a_{ij}) represents the verb's arguments. The yellow ovals represent foods; the grey ovals represent locations. Argument ovals with dotted boundaries are implicit, i.e., not present in text. The inner white ovals (s_{ij}^k) are string spans. The red dashed lines represent connections to string spans from their originating verb or raw ingredient. The string spans also connect to their associated verb in the action diagram to model the flow of ingredients. For example, there is a directed path from each raw ingredient to the implicit object of bake, representing that the object being baked is composed of all of the raw ingredients.

macaroni and cheese," to "cheesy salsa mac."

We present two models that are learned with hard EM algorithms: (1) a segmentation model to extract the actions from the recipe text, and (2) a graph model that defines a distribution over the connections between the extracted actions. The common sense knowledge is encoded in the second model which can, for example, prefer graphs that model implicit verb arguments when they better match the learned selectional preferences. The final action graph is constructed with a local search algorithm, that allows for global reasoning about ingredients as they flow through the recipe.

Experiments demonstrate the ability to recover high quality action graphs, gaining up to 8 points in F1 over a strong baseline where the ingredients flow sequentially through the verbs. The learned models are also highly interpretable, specifying for example that "dough" likely contains "flour" and that "add" generally requires two food arguments, even if only one is mentioned in the sentence.

2 Task Definition

Procedural text such as a recipe defines a set of actions, i.e. *predicates*, applied to a set of objects, i.e. *arguments*. A unique challenge in procedural text understanding is to recover how different arguments flow through a chain of actions; the results of intermediate actions (e.g., "Boil the pasta until al dente.") provide the inputs for future actions (e.g., "Drain."). We represent these correspondences with an *action graph*. In this section, we first describe our structured representation of recipe text, then we define how components of the recipe connect. Finally, we will show how given a recipe and a set of connections we can construct an action graph that models the flow of ingredients through the recipe. Fig. 1 provides a detailed running example for the section.

2.1 Recipe R

A recipe R is a piece of text that describes a list of instructions and a (possibly-empty) set of raw ingredients that are required to perform the instructions. The first step is to segment the text into a list of verb-argument tuples, called *actions*, $E_R = \{e_1 = (v_1, \mathbf{a}_1), \ldots, e_n = (v_n, \mathbf{a}_n)\}$. Sec. 6 will describe an unsupervised approach for learning to segment recipes. Each action e_i pairs a verb v_i with a list of arguments $\mathbf{a}_i$, where a_{ij} is the j^{th} argument of verb v_i. In Fig. 1, each row contains an action with a verb in the white oval and its arguments in the yellow and gray ovals.

Each argument is a tuple $a_{ij} = (t_{ij}^{syn}, t_{ij}^{sem}, S_{ij})$ with a syntactic type $t^{syn}(a) \in T^{syn} = \{DOBJ, PP\}$, a semantic type $t^{sem}(a) \in$

$T^{sem} = \{food, location, other\}$, and a list of text string spans $S_{ij} = \{s_{ij}^1, \ldots, s_{ij}^{|S_{ij}|}\}$, where s_{ij}^k is the k^{th} span in the j^{th} argument of verb v_i. In Fig. 1, the spans of each argument are represented by the white ovals inside of the argument ovals. For example, a_{21} contains a span for each raw ingredient being mixed in the second action (e.g., s_{21}^1 ="ground beef," s_{21}^6 ="brown sugar"). The syntactic type determines whether the argument is the direct object or a prepositional phrase argument of the verb in the recipe text. All other syntactic constructs are ignored and left for future work. The semantic types include food, location, and other. In Fig. 1, yellow ovals represent foods and gray ovals represent locations. Arguments of other semantic types are marked as $other$ (e.g., "Mash **using a fork**").

We also augment the set of arguments for each verb to include *implicit* arguments with empty string spans. This allows making connections to arguments that the author does not mention explicitly (e.g., the elided direct object of "bake" in e_5). Every verb is assigned one implicit PP argument, and, if a verb has no argument with syntactic type $DOBJ$, an implicit direct object. These arguments have indeterminate semantic types, which are to be determined based on how they connect to other actions. For example, in Fig. 1, when the implicit object of "bake" is connected to the output of the "lay" action, it is inferred to be of type $food$ since that is what is created by the "lay" action. However, when the implicit PP argument of "bake" is connected to the output of the "preheat" action, it is inferred to be a location since "preheat" does not generate a food.

2.2 Connections C

Given a segmented recipe, we can build graph connections. A *connection* identifies the origin of a given string span as either the output of a previous action or as a new ingredient or entity being introduced into the recipe. A connection is a six-tuple $(o, i, j, k, t^{syn}, t^{sem})$ indicating that there is a connection from the output of v_o to the argument span s_{ij}^k with syntactic type $t^{syn} \in T^{syn}$ and semantic type $t^{sem} \in T^{sem}$. We call o the *origin index* and i the *destination index*. For example, in Fig. 1, the connection from the output of the "press" verb (e_3) to "over the top" (s_{42}^1) would be $(3, 4, 2, 1, PP, food)$. If a span introduces raw ingredient or new location into the recipe, then $o = 0$; in Fig. 1, this occurs for each of the spans that represent raw ingredients as well as "oven" and "into loaf pan."

Given a recipe R, a set of connections C is valid for R if there is a one-to-one correspondence between spans in R and connections in C, and if the origin indexes of connections in C are 0 or valid verb indexes in R, $\forall(o, i, j, k, t^{syn}, t^{sem}) \in C,\ o \in \{\mathbb{Z} \mid 0 \leq o \leq |E_R|\}$.

2.3 Action graph G

A recipe R and a set of connections C define an *action graph*, which is a directed graph $G = (V, E)$. Each raw ingredient, verb, and argument span is represented by a vertex in V. Each argument span vertex is connected to its associated verb vertex, and each connection $c = (o, i, j, k, t^{syn}, t^{sem})$ adds a corresponding edge to E. Edges from connections with semantic type $food$ propagate ingredients through the recipe; edges from connections with semantic type $location$ propagate a location. Fig. 1 shows an action graph. By following the edges, we can tell that the implicit food entity that is being baked in the final action has been formed from the set of ingredients in the mixing action and the bacon from e_4 and that the baking action occurs inside the oven preheated in e_1.

3 Probabilistic connection model

Our goal is, given a segmented recipe R, to determine the most likely set of connections, and thus the most likely action graph. We model (1) a prior probability over C, $P(C)$ (Sec. 3.1), and (2) the probability of seeing a segmented recipe R given a set of connections C, $P(R|C)$ (Sec. 3.2). The most likely set of connections will maximize the joint probability: $P(R|C)P(C)$. A summary of this model is presented in Fig. 2, and the details are described in the this section.

3.1 Connections prior model

The probability of a set of connections C depends on features of the incoming set of connections for each action. Let a *destination subset* $\mathbf{d}_i \subseteq C$ be the subset of C that contains all connections that have i as the destination index. In Fig. 1, $\mathbf{d}_3$ contains the connection from v_2 to the implicit object as well as a connection to "into loaf pan" with an origin index of 0. Using the chain rule, the probability of C is equal to the product of the probability

- **Input:** A set of connections C and a recipe R segmented (**Sec. 6**) into its actions $\{e_1 = (v_1, \mathbf{a}_1), \ldots, e_n = (v_n, \mathbf{a}_n)\}$
- The joint probability of C and R is $P(C, R) = P(C)P(R|C)$, each defined below:

1. **Connections Prior (Sec. 3.1)**: $P(C) = \prod_i P(\mathbf{d}_i|\mathbf{d}_1, \ldots, \mathbf{d}_{i-1})$
   Define $\mathbf{d}_i$ as the list of connections with destination index i. Let $c_p = (o, i, j, k, t^{syn}, t^{sem}) \in \mathbf{d}_i$. Then,
   - $P(\mathbf{d}_i|\mathbf{d}_1, \ldots, \mathbf{d}_{i-1}) = P(vs(\mathbf{d_i})) \prod_{c_p \in \mathbf{d_i}} P(\mathbb{1}(o \rightarrow s^k_{ij})|vs(\mathbf{d_i}), \mathbf{d}_1, \ldots, \mathbf{d}_{i-1}, c_1, \ldots, c_{p-1})$
     (a) $P(vs(\mathbf{d_i}))$: multinomial verb signature model (**Sec. 3.1.1**)
     (b) $P(\mathbb{1}(o \rightarrow s^k_{ij})|vs(\mathbf{d_i}), \mathbf{d}_1, \ldots, \mathbf{d}_{i-1}, c_1, \ldots, c_{p-1})$: multinomial connection origin model, conditioned on the verb signature of $\mathbf{d}_i$ and all previous connections (**Sec. 3.1.2**)
2. **Recipe Model (Sec. 3.2)**: $P(R|C) = \prod_i P(e_i|C, e_1, \ldots, e_{i-1})$
   For brevity, define $\mathbf{h}_i = (e_1, \ldots, e_{i-1})$.
   - $P(e_i|C, \mathbf{h}_i) = P(v_i|C, \mathbf{h}_i) \prod_j P(a_{ij}|C, \mathbf{h}_i)$ (**Sec. 3.2**)
     Define argument a_{ij} by its types and spans, $a_{ij} = (t^{syn}_{ij}, t^{sem}_{ij}, S_{ij})$.
     (a) $P(v_i|C, \mathbf{h}_i) = P(v_i|g_i)$: multinomial verb distribution conditioned on verb signature (**Sec. 3.2**)
     (b) $P(a_{ij}|C, \mathbf{h}_i) = P(t^{syn}_{ij}, t^{sem}_{ij}|C, \mathbf{h}_i) \prod_{s^k_{ij} \in S_{ij}} P(s^k_{ij}|t^{syn}_{ij}, t^{sem}_{ij}, C, \mathbf{h}_i)$
        i. $P(t^{syn}_{ij}, t^{sem}_{ij}|C, \mathbf{h}_i)$: deterministic argument types model given connections (**Sec. 3.2.1**)
        ii. $P(s^k_{ij}|t^{syn}_{ij}, t^{sem}_{ij}, C, \mathbf{h}_i)$: string span model computed by case (**Sec. 3.2.2**):
           A. $t^{sem}_{ij} = food$ and $origin(s^k_{ij}) \neq 0$: IBM Model 1 generating composites (**Part-composite model**)
           B. $t^{sem}_{ij} = food$ and $origin(s^k_{ij}) = 0$: naïve Bayes model generating raw food references (**Raw food model**)
           C. $t^{sem}_{ij} = location$: model for generating location referring expressions (**Location model**)

Figure 2: Summary of the joint probabilistic model $P(C, R)$ over connection set C and recipe R.

of each of the destination subsets:

$$P(C) = \prod_i P(\mathbf{d}_i|\mathbf{d}_1, \ldots, \mathbf{d}_{i-1}).$$

The probability of each destination subset decomposes into two distributions, a verb signature model and a connection origin model:

$$P(\mathbf{d}_i|\mathbf{d}_1, \ldots, \mathbf{d}_{i-1}) = P(vs(\mathbf{d_i})) \times \prod_{c_p \in \mathbf{d_i}} P(\mathbb{1}(o \rightarrow s^k_{ij})|vs(\mathbf{d_i}), \mathbf{d}_1^{i-1}, c_1^{p-1}).$$

We define each of these distributions below.

3.1.1 Verb signature model

A destination subset $\mathbf{d}_i$ deterministically defines a *verb signature* g_i for verb v_i based on the syntactic and semantic types of the connections in $\mathbf{d}_i$ as well as whether or not each connection has a non-zero origin index. If the origin index is 0 for all connections in $\mathbf{d}_i$, we call v_i a *leaf*. (In Fig, 1, v_1 (preheat) and v_2 (mix) are leafs.) The formal definition of a verb signature is as follows:

Definition 1 The *verb signature* g_i for a verb v_i given a destination set $\mathbf{d}_i$ consists of two parts:

1. **type:** $\{t^{syn} \mid \exists (o, i, j, k, t^{syn}, food) \in \mathbf{d}_i\}$
2. **leaf:** true iff $(o, i, j, k, t^{syn}, t^{sem}) \in \mathbf{d}_i \Rightarrow o = 0$

For example, in Fig. 1, the signature for the "mix" action is $g_2 = (\{DOBJ\}, true)$ and the signature for the "lay" action is $g_4 = (\{DOBJ, PP\}, false)$. Given that there are two syntactic types (i.e., $DOBJ$ and PP) and each verb signature can either be labeled as a leaf or not, there are eight possible verb signatures.

We define a deterministic function that returns the verb signature of a destination subset: $vs(\mathbf{d}_i) = g_i$. $P(vs(\mathbf{d}_i))$ is a multinomial distribution over the possible verb signatures.

3.1.2 Connection origin model

We define $\mathbb{1}(o \rightarrow s^k_{ij})$ as an indicator function that is 1 if there is a connection from the action with index o to the span s^k_{ij}. The probability that a string span has a particular origin depends on (1) the verb signature of the span's corresponding verb, and (2) the previous connections. If, for example, g_i has **leaf**$= true$, then the origin of s^k_{ij} must be 0. If an origin has been used in a previous connection, it is much less likely to be used again.[2]

We assume that a destination subset is a list of connections: if $c_p \in \mathbf{d}_i$, we define c_1^{p-1} as the connections that are prior to c_p in the list. Similarly, $\mathbf{d}_1^{i-1}$ is the set of destination sets $(\mathbf{d}_1, \ldots, \mathbf{d}_{i-1})$. The connection origin model is a multinomial distribution that defines the probability of an origin for a span conditioned on the verb signature and all previous connections:

$$P(\mathbb{1}(o \rightarrow s^k_{ij})|vs(\mathbf{d_i}), \mathbf{d}_1^{i-1}, c_1^{p-1}),$$

[2] A counterexample in the cooking domain is separating egg yolks from egg whites to be used in separate components, only to be incorporated again in a later action.

where $c_p = (o, i, j, k, t^{syn}, t^{sem})$.

3.2 Recipe model

Given a set of connections C for a recipe R, we can determine how the actions of the recipe interact and we can calculate the probability of generating a set of recipe actions $E_R = \{e_1 = (v_1, \mathbf{a}_1), \ldots, e_n = (v_n, \mathbf{a}_n)\}$. Intuitively, R is more likely given C if the destinations of the connections are good text representations of the origins. For example, a string span "oven" is much more likely to refer to the output of the action "Preheat the oven" than "Mix flour and sugar."

We define the history $\mathbf{h}_i$ of an action to be the set of all previous actions: $\mathbf{h}_i = (e_1, \ldots, e_{i-1})$. The probability of a recipe R given a set of connections C can be factored by the chain rule:

$$P(R|C) = \prod_i P(e_i|C, \mathbf{h}_i).$$

Given C and a history $\mathbf{h}_i$, we assume the verb and arguments of an action are independent:

$$P(e_i|C, \mathbf{h}_i) = P(v_i|C, \mathbf{h}_i) \prod_j P(a_{ij}|C, \mathbf{h}_i).$$

Since the set of connections deterministically defines a verb signature g_i for a verb v_i, we can simplify $P(v_i|C, \mathbf{h}_i)$ to the multinomial distribution $P(v_i|g_i)$. For example, if g_i defines the verb to have an ingredient direct object, then the probability of "preheat" given that signature will be lower than the probability of "mix."

The probability of an argument $a_{ij} = (t^{syn}_{ij}, t^{sem}_{ij}, S_{ij})$ given the connections and history decomposes as follows:

$$\begin{aligned} P(a_{ij}|C, \mathbf{h}_i) &= P(t^{syn}_{ij}, t^{sem}_{ij}|C, \mathbf{h}_i) \\ &\times P(S_{ij}|t^{syn}_{ij}, t^{sem}_{ij}, C, \mathbf{h}_i). \end{aligned}$$

3.2.1 Argument types model

The first distribution, $P(t^{syn}_{ij}, t^{sem}_{ij}|C, \mathbf{h}_i)$, ensures that the syntactic and semantic types of the argument match the syntactic and semantic type of the incoming connections to spans of that argument. The probability is 1 if all the types match, 0 otherwise. For example, in Fig. 1, this distribution would prevent a connection from the "preheat" action to the food argument a_{42}, i.e., "over the top," since the semantic types would not match.

3.2.2 String span models

The second distribution, $P(S_{ij}|t^{syn}_{ij}, t^{sem}_{ij}, C, \mathbf{h}_i)$, models how likely it is to generate a particular string span given the types of its encompassing argument, the connections, and history. We assume the probability of each span is independent:

$$P(S_{ij}|t^{syn}_{ij}, t^{sem}_{ij}, C, \mathbf{h}_i) = \prod_{s^k_{ij} \in S_{ij}} P(s^k_{ij}|t^{syn}_{ij}, t^{sem}_{ij}, C, \mathbf{h}_i).$$

We break this distribution into three cases. To help describe the separate cases we define the function $origin(s, C)$ to determine the origin index of the connection in C to the span s. That is, $origin(s^k_{ij}, C) = o \Leftrightarrow \exists (o, i, j, k, t^{syn}, t^{sem}) \in C$.

Part-composite model When the encompassing argument is a food and the origin is a previous verb (i.e., $P(s^k_{ij}|t^{syn}_{ij}, t^{sem}_{ij} = food, origin(s^k_{ij}) \neq 0, C, \mathbf{h}_i)$), then the probability of the span depends on the ingredients that the span represents given the connections in C. For example, "dressing" is more likely given ingredients "oil" and "vinegar" than given "chicken" and "noodles". We use IBM Model 1 (Brown et al., 1993) to model the probability of a composite destination phrase given a set of origin food tokens. Let $food(s^k_{ij}, C)$ be the set of spans in food arguments such that there is a directed path from those arguments to s^k_{ij}. IBM Model 1 defines the probability of a span given the propagated food spans, $P(s^k_{ij}|food(s^k_{ij}, C))$.[3]

Raw food model When the encompassing argument is a food but the origin index is 0 (i.e., $P(s^k_{ij}|t^{syn}_{ij}, t^{sem}_{ij} = food, origin(s^k_{ij}) = 0, C, \mathbf{h}_i)$), then there is no flow of ingredients into the span. A span that represents a newly introduced raw ingredient (e.g., "bacon" in e_4 of Fig. 1) should have a high probability. However, spans that denote the output of actions (e.g, 'batter," "banana mixture") should have low probability. We use a naïve Bayes model over the tokens in the span $P(s|\text{is raw}) = \prod_\ell P(w_\ell|\text{is raw})$ where w_ℓ is the ℓ^{th} token in s (e.g., "mixture" would have a very low probability but "flour" would be likely).

Location model When the encompassing argument is a location (i.e., $t^{sem}_{ij} = location$),

[3] IBM Model 1 cannot handle implicit arguments. In this case, we model the probability of having an implicit food argument given the length of the connection (i.e., implicit food arguments nearly deterministically connect to the previous action). The probability of non-empty string spans is scaled accordingly to ensure a valid probability distribution.

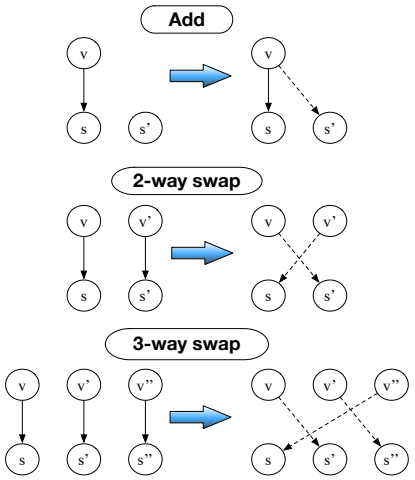

Figure 3: The three types of search operators. For swaps, one of the origins can be 0.

$P(S_{ij}|t_{ij}^{syn}, t_{ij}^{sem}, C, \mathbf{h})$ models the appropriateness of the origin action's location for the destination. If the string span is not implicit, the model deterministically relies on string match between the span and the location argument of the verb at the origin index. For example, the probability of "the preheated oven" conditioned on an origin with location "oven" is 1, but 0 for an origin with location "bowl." If the span s_{ij}^k is empty, we use a multinomial model $P(loc(origin(s_{ij}^k, C))|v_i)$ that determines how likely it is that an action v_i occurs in the location of the origin verb. For example, baking generally happens in an oven and grilling on a grill, but not vice versa. For example, in Fig. 1, the probability of the location span of "bake" is determined by P("oven" | "bake").

4 Local Search

Connections among actions and arguments identify which ingredients are being used by which action. For example, in Fig. 1, we know that we are baking something that contains all the ingredients introduced in e_2 and e_4 because there is a path of connections from the introduction of the raw ingredients to the implicit object of "bake". We cannot make decisions about the origins of arguments independently; the likelihood of each edge depends on the other edges. Identifying the most likely set of connections is, therefore, intractable.

We adopt a local search approach to infer the best set of connections.[4] We initialize the set of connections using a sequential algorithm that connects the output of each event to an argument of the following event, which is a strong baseline as shown in Sec. 8. We score potential local search operators that can be applied to the current set of connections C and make a greedy selection that improves $P(C, R)$ the most until no search operator can improve the probability. We constrain the search so all verbs have a direct object (i.e., implicit direct objects connect to a previous action).

We employ three types of search operators (see Fig. 3 for details). **OP_ADD** changes the origin index of a connection in C from 0 to the index of an event. **OP_2SWAP** swaps the origin indexes of two connections. This works even if one of the origin indexes is 0. **OP_3SWAP** rotates the origin indexes of three connections. This works even if one of the origin indexes is 0. For efficiency reasons, we only allow 3-way swaps with destination indexes within 4 events of each other.

Algorithm 1 Pseudocode for learning $P(C, R)$

Input: Initialized $P(C, R)$, recipe dataset $\mathcal{R}$
Repeat until convergence:
  E-step: Update $C \longleftarrow \arg\max_C P(C, R)$ for each $R \in \mathcal{R}$ using local search (Sec. 4)
  M-step: Update parameters of $P(C, R)$ using action graphs generated in E-step
Return $P(C, R)$

5 Learning

We use hard EM to learn the probabilistic models. Pseudocode is given in Alg. 1. At each iteration, we use our local search algorithm and the current probabilistic models to annotate each recipe in the data set with its most likely set of connections C (Sec. 4). Then, we re-estimate the parameters of the probabilistic models using the recipe-connections pairs as training data. A small (33 recipes) development set was used to determine when to stop the iterations. Experimental details and model initialization are described in Sec. 7.

6 Segmentation

Our inference and learning algorithms assume as input a recipe segmented into a set of events $E_R = \{(v_1, \mathbf{a}_1), \ldots, (v_n, \mathbf{a}_n)\}$. We designed a segmentation system that could be trained on our unannotated data set of mostly imperative sentences.

[4] Similar local search methods have been shown to work well for other NLP tasks, including recent work on dependency parsing (Zhang et al., 2014).

Our system achieves an F1 score of 95.6% on the task of identifying the correct verbs in the test set.[5]

Segmentation model We define a generative model for recipes as:

$$P(R) = P(n) \prod_{i}^{n} P(v_i) P(m \mid v_i) \prod_{j=1}^{m} P(a_{ij}).$$

We first select a number of verbs n in the recipe from a geometric distribution. Given the number of verbs, we select a set of verbs $\mathbf{V} = \{v_1, \ldots, v_n\}$ using a multinomial distribution. For each verb v_i, we select a number of arguments m from a separate multinomial distribution that has the probability of 0, 1, 2, or 3+ arguments given the verb, $P(m \mid v_i)$. For each argument, we generate a string using a bigram model, $P(a_{ij}) = \prod_{\ell} P(w_\ell | w_{\ell-1})$, where w_ℓ is the ℓ^{th} word of a_{ij}.

Inference Given tokenized sentence $T = (w_1, \ldots, w_k)$, we enumerate all possible segmentations and choose the one with the highest probability. To keep this efficient, we use a closed set of possible verbs and assume a closed set of words (e.g., prepositions, adverbs) can only follow the start token in the argument bigram model. Thus, annotating the verbs in a sentence determines a unique set of argument strings. Despite scoring the segmentations for all possible sets of verbs, we found the process to be very efficient in practice.

Learning For unsupervised learning, we again employ a hard EM approach. We initialize our models, segment all of the training data, re-estimate the parameters, and iterate these steps until performance on a development set converges.

We estimate the initial verb multinomial model using counts from the first word of each sentence in the dataset, which are normally verbs in imperative sentences, and filter out any words that have no verb synsets in WordNet (Miller, 1995). All other models are initialized to be uniform.

7 Experimental Setup

Data Set We collected 2456 recipes (with over 23,000 sentences) from allrecipes.com by searching for 20 dish names (e.g., including "banana muffins", and "deviled eggs"). We randomly sampled, removed, and hand labeled 33 recipes for a development set and 100 recipes for test. All models were trained on the unannotated recipes; the dev set was used to determine the stopping point for training. Each recipe in the test set has 13 actions on average.

Recipe pre-processing To pre-process each recipe, we first use the segmentation system described in Sec. 6. Then, we use a string classification model to determine the semantic type (e.g., $food$, $location$, or $other$) of an argument based on its spans. We identify spans as raw ingredients based on string match heuristics (e.g., in Fig. 1, the span "crushed crackers" represents the ingredients "crushed butter-flavored crackers"). We stem all words and ignore function words.

Sequential Baseline Because most connections are sequential – i.e., argument spans are most often connected to the output of the previous verb – sequential connections make a strong baseline; we connect the output of each predicate to the first available argument span of the following predicate. If no argument exists, an implicit argument is created. We run this baseline with and without first identifying raw ingredients in the recipe; if raw ingredient spans are identified, the baseline will not connect the previous event to those spans. Performance suffers significantly if the raw ingredients are not identified beforehand.

Evaluation metrics We report F-measure by comparing the predicted connections from actions to spans (i.e., where the origin index > 0) against gold standard annotations. We don't evaluate connections to raw ingredients as we create those connections during pre-processing (see Sec. 7).

Model initialization The verb signature model (Sec. 3.2) is initialized by first identifying $food$ arguments using string overlap with the ingredient list. All other arguments' types are considered unknown, and partial counts were awarded to all verb signatures consistent with the partial information. The first verb in each recipe was assumed to be the only leaf. The string classification model for the pre-processing step was initialized by using the initialized verb signature model to identify the types of $DOBJ$ arguments. The string classification model was estimated using the argument tokens given the types. We initialized the part-composite model (Sec. 3.2.2) so that exact string matches between ingredients and spans are given

[5] Early efforts using a state-of-the-art parser could only achieve an F1 score of 73.6% for identifying verbs, likely due to a lack of imperative sentences in the training data. This result motivated us to develop our segmentation system.

Algorithm	Prec	Rec	F1
Automatic segmentations			
Sequential baseline	55.7	52.7	54.1
Sequential baseline w/ ingredients	60.4	57.2	58.8
Our model before EM	65.8	62.7	64.2
Our model after EM	**68.7**	**65.0**	**66.8**
Oracle segmentations			
Sequential baseline	67.8	65.2	66.5
Sequential baseline w/ ingredients	73.5	70.7	72.0
Our model before EM	77.1	74.8	75.9
Our model after EM	**81.6**	**78.5**	**80.0**

Table 1: Performance of our algorithm against the sequential baselines.

Verb	Top location tokens	
bake	oven - 55.4%	min - 0.7%
mix	bowl - 32.6%	hand - 0.9%
press	pan - 24.7%	dish - 6.5%
stir	bowl - 5.5%	skillet - 2.0%
fry	heat - 11.9%	skillet - 10.2%
cool	rack - 10.5%	pan - 3.8%
boil	water - 15.5%	saucepan - 5.2%

Table 2: The top scoring location token for example verbs. The percentage is the percent of times the verb has that as a visible location token.

high probabilities and those without are given low probabilities. Given the initialized string classification model, the raw food model (Sec. 3.2.2) is initialized counting whether or not tokens in food arguments occur in the ingredient list. The probability of an implicit location (Sec. 3.2.2) is initialized to a hand-tuned value using the dev set.

8 Results

Quantitative Results We trained our model for four iterations of hard EM until performance converged on the development set. Table 1 presents our results on the test set. We compare our model to the sequential baselines using both the output of our segmentation system and oracle segmentations. We perform significantly better than the sequential baselines, with an increase in F1 of 8 points over the more competitive baseline using our segmentation system and an increase of 8 points using the oracle segmentations.

Qualitative Results We find that the learned models demonstrate interpretable cooking knowledge. Table 3 shows the top composite tokens for different ingredients as learned by the part-composite model (Sec. 3.2.2). The composite tokens show parts of the ingredient (e.g., after "eggs" can be split into "whites" or "yolks") or composites that are likely to contain an ingredient (e.g., "flour" is generally found in "batter" and "dough"). Unsurprisingly, the word "mixture" is one of the top words to describe a combination of ingredients, regardless of the ingredient. The model also learns modifiers that describe key properties of ingredients (e.g., flour is "dry" but bananas are "wet") which is important when evaluating connections for sentences such as "Fold the wet mixture into the dry ingredients."

Verb	Top verb signature	(%)
add	$\{DOBJ, PP\}$	58%
	$\{DOBJ\}$	27%
combine	$\{DOBJ\}$:leaf	68%
	$\{DOBJ\}$	17%
bake	$\{DOBJ\}$	95%
grease	{}:leaf	75%
pour	$\{DOBJ, PP\}$	68%
	$\{DOBJ\}$	27%
reduce	$\{PP\}$	90%
	$\{DOBJ\}$	8%

Table 4: The top verb signatures for example verbs. The syntactic types identify which arguments of the verb are foods and "leaf" means no arguments of the verb connect to previous actions.

Table 2 shows the location preferences of verbs learned by the location model (Sec. 3.2.2). Some verbs show strong preferences on locations (e.g., "bake" occurs in an oven, "mix" in a bowl). The top location for a "boil" action is in "water," but in other recipes "water" is an ingredient.

Finally, Table 4 shows learned verb signatures. For example, "add" tends to be a non-leaf action, and can take one or two food arguments (e.g., one food argument: "Heat the pan. Add onions." vs. two food arguments: "Add the wet mixture to the dry mixture.") We learn that the most likely verb signature for "add" has two food arguments; since over 74% of the occurrences of "add" in the dataset only have one visible argument, the segmentation alone is not enough to determine the signature.

Errors Finally, we performed an error analysis on the development set. 24% of the errors were due to missing or incorrect actions caused by segmentation errors. Among the actions that were segmented correctly, 82% of the outgoing connections were sequential. Of those, our system missed 17.6% of the sequential connections and 18.3% of the non-sequential connections.

Ingredient	Top composite tokens
eggs	egg, yolk, mixture, noodles, whites, cook, top, potato, cold, fill
beef	beef, mixture, grease, meat, excess, cook, top, loaf, sauce, ground
flour	flour, mixture, dough, batter, top, crust, ingredients, sauce, dry, pie
noodles	noodles, cook, mixture, egg, sauce, top, meat, drain, pasta, layer
chicken	chicken, mixture, salad, cook, dressing, pasta, soup, breast, vegetables, noodles
pumpkin	pumpkin, mixture, pie, filling, temperature, seeds, mash, oven, crust, dough
bananas	banana, mixture, batter, muffin, bread, egg, wet, cup, ingredients, slice

Table 3: Examples of ingredients with their top inferred composite words.

9 Related work

Our work relates to a substantial body of research that transforms natural language instructions into actionable plans (Artzi and Zettlemoyer, 2013, Chen and Mooney, 2011, Branavan et al., 2011, Branavan et al., 2009, McMahon et al., 2006). Most of these approaches do interactive learning in virtual environments or simulations, while we learn from the redundancy seen in the text of different instances of similar recipes.

There is also significant related work on supervised learning for instructions. A recent series of studies have explored parsing of cooking recipes (Mori et al., 2012; Mori et al., 2014; Maeta et al., 2015). However, they assume annotated data, study Japanese recipes, and make edge connections independently without taking into account the flow of ingredients. Tasse and Smith (2008) develops annotation for English recipes, but do not mark connections from implicit roles, and only studied segmentation models. Lau et al. (2009) develop models to interpret how-to instructions, but also assume supervision, and do not make connections between different actions.

Data-driven extraction of cooking knowledge has been explored in the context of building a cooking ontology (Gaillard et al., 2012; Nanba et al., 2014). In contrast, our work induces probabilistic cooking knowledge as part of unsupervised learning process for understanding recipes. Cooking knowledge is also closely related to script knowledge, but most prior work focus on newswire and children's books rather than procedural language (Fujiki et al., 2003; Chambers and Jurafsky, 2009; Pichotta and Mooney, 2014; Balasubramanian et al., 2013) or rely on crowdsourced descriptions to learn procedural knowledge (Regneri et al., 2010; Regneri et al., 2011; Frermann et al., 2014). There is work on related, but distinct, tasks that use recipes, including identifying actionable refinements from online recipe reviews (Druck and Pang, 2012) and extracting structured information from ingredient lists (Greene, 2015)

Cooking recipes have also been studied in the context of grounded language learning, e.g., to build robots that can cook (e.g., Bollini et al., 2013, Beetz et al., 2011), or to align cooking videos to natural language descriptions of actions (Regneri et al., 2013) or recipe texts (Malmaud et al., 2014; Malmaud et al., 2015). Our work complements these efforts by recovering fine-grained procedural semantics from text alone.

Finally, detection and resolution of implicit arguments is an instance of zero anaphora detection and resolution (Silberer and Anette, 2012, Tetreault 2002, Whittemore et al., 1991, Palmer et al., 1986). We present an empirical approach for understanding these phenomena in instructions.

10 Conclusion

We presented unsupervised methods for segmenting and identifying latent connections among actions in recipe text. Our model outperformed a strong linear baseline, while learning a variety of domain knowledge, such as verb signatures and probable ingredient components for different composites. Future work includes learning a more comprehensive model of locations (e.g., identifying nested locations such as an oven and a pan in the oven), enriching action graphs with greater semantic coverage (e.g., durations, tools, amounts), and training and evaluating on larger datasets. We also plan to use our techniques to support related tasks, such as instructional recipe generation.

Acknowledgments

We thank the anonymous reviewers, Mike Lewis, Dan Weld, Yoav Artzi, Antoine Bosselut, Kenton Lee, Luheng He, Mark Yatskar, and Gagan Bansal for helpful comments, and Polina Kuznetsova for the preliminary work. This research was supported in part by the Intel Science and Technology Center for Pervasive Computing (ISTC-PC) and the NSF (IIS-1252835 and IIS-1524371).

References

Yoav Artzi and Luke Zettlemoyer. 2013. Weakly supervised learning of semantic parsers for mapping instructions to actions. *Transactions of the Association for Computational Linguistics*, 1(1):49–62.

Niranjan Balasubramanian, Stephen Soderland, Mausam, and Oren Etzioni. 2013. Generating coherent event schemas at scale. In *Proceedings of the 2013 Conference on Empirical Methods on Natural Language Processing*, pages 1721–1731.

Michael Beetz, Ulrich Klank, Ingo Kresse, Alexis Maldonado, Lorenz Mosenlechner, Dejan Pangercic, Thomas Ruhr, and Moritz Tenorth. 2011. Robotic roommates making pancakes. In *Proceedings of the 11th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pages 529–536.

Mario Bollini, Stefanie Tellex, Tyler Thompson, Nicholas Roy, and Daniela Rus. 2013. Interpreting and executing recipes with a cooking robot. *Experimental Robotics*, 88:481–495.

S.R.K. Branavan, Harr Chen, Luke Zettlemoyer, and Regina Barzilay. 2009. Reinforcement learning for mapping instructions to actions. 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: Volume 1 - Volume 1*, pages 82–90.

S.R.K. Branavan, David Silver, and Regina Barzilay. 2011. Non-linear monte-carlo search in civilization ii. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence*, pages 2404–2410.

Peter F. Brown, Vincent J. Della Pietra, Stephen A. Della Pietra, and Robert L. Mercer. 1993. The mathematics of statistical machine translation: parameter estimation. *Computational Linguistics*, 19:263–311.

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 Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing*, pages 602–610.

David L. Chen and Raymond J. Mooney. 2011. Learning to interpret natural language navigation instructions from observations. In *Proceedings of the 25th AAAI Conference on Artificial Intelligence (AAAI-2011)*, pages 859–865.

Gregory Druck and Bo Pang. 2012. Spice it up? Mining refinements to online instructions from user generated content. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*, pages 545–553.

Charles J. Fillmore, 1982. *Frame semantics*, pages 111–137. Hanshin Publishing Co., Seoul, South Korea.

Lea Frermann, Ivan Titov, and Manfred Pinkal. 2014. A hierarchical bayesian model for unsupervised induction of script knowledge. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 49–57.

Toshiaki Fujiki, Hidetsugu Nanba, and Manabu Okumura. 2003. Automatic acquisition of script knowledge from a text collection. In *Proceedings of the Tenth Conference on European Chapter of the Association for Computational Linguistics - Volume 2*, pages 91–94.

Emmanuelle Gaillard, Emmanuel Nauer, Marie Lefevre, and Amélie Cordier. 2012. Extracting generic cooking adaptation knowledge for the TAAABLE case-based reasoning system. In *Proceedings of the 1st Workshop on Cooking with Computers (CwC)*.

Erica Greene. 2015. Extracting structured data from recipes using conditional random fields. The New York Times Open Blog.

TA Lau, Clemens Drews, and Jeffrey Nichols. 2009. Interpreting written how-to instructions. In *Proceedings of the Twenty-First International Joint Conference on Artificial Intelligence*, pages 1433–1438.

Matt MacMahon, Brian Stankiewicz, and Benjamin Kuipers. 2006. Walk the talk: Connecting language, knowledge, and action in route instructions. In *Proceedings of the 21st National Conference on Artificial Intelligence - Volume 2*, AAAI'06, pages 1475–1482. AAAI Press.

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.

Jon Malmaud, Earl J. Wagner, Nancy Chang, and Kevin Murphy. 2014. Cooking with semantics. In *Proceedings of the ACL 2014 Workshop on Semantic Parsing*, pages 33–38.

Jonathan Malmaud, Jonathan Huang, Vivek Rathod, Nick Johnston, Andrew Rabinovich, and Kevin Murphy. 2015. What's cookin'? Interpreting cooking videos using text, speech and vision. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 143–152.

George A. Miller. 1995. WordNet: A lexical database for english. *Communications of the ACM*, 38(11):39–41.

Shinsuke Mori, Tetsuro Sasada, Yoko Yamakata, and Koichiro Yoshino. 2012. A machine learning approach to recipe text processing. In *Proceedings of the 1st Workshop on Cooking with Computers (CwC)*.

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 26–31.

Hidetsugu Nanba, Yoko Doi, Miho Tsujita, Toshiyuki Takezawa, and Kazutoshi Sumiya. 2014. Construction of a cooking ontology from cooking recipes and patents. In *Proceedings the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct Publication*, pages 507–516.

Martha S. Palmer, Deborah A. Dahl, Rebecca J. Schiffman, Lynette Hirschman, Marcia Linebarger, and John Dowding. 1986. Recovering implicit information. In *Proceedings of the 24th Annual Meeting on Association for Computational Linguistics*, pages 10–19.

Karl Pichotta and Raymond Mooney. 2014. Statistical script learning with multi-argument events. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 220–229.

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.

Michaela Regneri, Alexander Koller, Josef Ruppenhofer, and Manfred Pinkal. 2011. Learning script participants from unlabeled data. In *Proceedings of the Conference on Recent Advances in Natural Language Processing*, pages 463–470.

Michaela Regneri, Marcus Rohrbach, Dominikus Wetzel, Stefan Thater, Bernt Schiele, and Manfred Pinkal. 2013. Grounding action descriptions in videos. *Transactions of the Association for Computational Linguistics (TACL), Volume 1.*, 1:25–36.

Roger Carl Schank and Robert P. Abelson. 1977. *Scripts, plans, goals and understanding : an inquiry into human knowledge structures*. The Artificial intelligence series. L. Erlbaum, Hillsdale, N.J.

Carina Silberer and Anette Frank. 2012. Casting implicit role linking as an anaphora resolution task. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics - Volume 1: Proceedings of the Main Conference and the Shared Task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation*, pages 1–10.

Dan Tasse and Noah A. Smith. 2008. SOUR CREAM: Toward semantic processing of recipes. Technical Report CMU-LTI-08-005, Carnegie Mellon University, Pittsburgh, PA.

Joel R. Tetreault. 2002. Implicit role reference. In *Proceedings of the International Symposium on Reference Resolution for Natural Language Processing*, pages 109–115.

Greg Whittemore, Melissa Macpherson, and Greg Carlson. 1991. Event-building through role-filling and anaphora resolution. In *Proceedings of the 29th Annual Meeting on Association for Computational Linguistics*, pages 17–24.

Yuan Zhang, Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2014. Greed is good if randomized: New inference for dependency parsing. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1013–1024.

Arnold M. Zwicky. 1988. On the subject of bare imperatives in english. In C. Duncan-Rose and T. Vennemann, editors, *On Language: Rhetorica, Phonologica, Syntactica - A Festschrift for Robert P. Stockwell from His Friends and Colleagues*, pages 437–450. Routledge, London.