Unsupervised Interpretation of Instructional Food Recipes

Andy Aliko
Natural Language Processing 1
Masters of Artificial Intelligence
University of Amsterdam
1012 WX Amsterdam
aliko.andy@gmail.com

Baris Demirdelen
Natural Language Processing 1
Masters of Artificial Intelligence
University of Amsterdam
1012 WX Amsterdam
barisdemirdelen@gmail.com

Oswald Hotz de baar
Natural Language Processing 1
Masters of Artificial Intelligence
University of Amsterdam
1012 WX Amsterdam
oz_hotz@yahoo.co.uk

Helena Russello
Natural Language Processing 1
Masters of Artificial Intelligence
University of Amsterdam
1012 WX Amsterdam
helena.russello@gmail.com

Abstract

This report details the implementation project of an algorithm for unsupervised interpretation of instructional recipes based on a paper of the same title[1]. In this project, food recipes are used as an example of how an unsupervised learning algorithm -using Expectation Maximization- can be used to map a text containing instructional language into action graphs. Recipes contain many real language problems, implicit arguments are used and objects are combined and change their name through the cooking process. This report shows how syntactic, semantic and world knowledge can be combined with unsupervised learning to infer likely occurrences to improve an action graph from a sequential baseline and provide statistical information of recipe language.

1 Introduction

Despite instructional language being an extremely useful part of language there has been little research in this area. In this report an unsupervised learning approach will be used to interpret text based recipes into a list of cooking actions.

The recipes are segmented into a list of actions, defined by their verb and their arguments. Then a graph model is produced, with edges being connections between actions. The objects in each action should receive only the connections that represent their origin, from either raw ingredients, previous actions or locations, and it should receive all the connections it needs. A sequential baseline will be applied and then this will be improved upon using semantic knowledge to determine likelihoods and local search with an Expectation Maximization algorithm. In the end, the recipe will be a graph of actions with edges connecting the output of each action to where it is subsequently used.

Recipes, like all human language, pose many challenges. Firstly, many sentences contain implicit arguments, which must be inferred, for example, Press into the pan does not explicitly say what to press into the pan. Secondly, even if the sentence was Press the mixture into the pan, world

[1]Chloe Kiddon, Ganesa Thandavam Ponnuraj, Luke Zettlemoyer & Yejin Choi. (2015) *Mise en Place: Unsupervised Interpretation of Instructional Recipes*. In EMNLP, pages 982-992.

knowledge is required to know that the mixture was produced by the action mix the meat and herbs. Further, a range of verbs could be used to describe the action of mixing, for example blend or even add, and action steps can change the way an object is referred to in the text, so dough can be put into the oven, but then bread taken out. Consequently, semantic analysis of the text is needed before inference can be made, but inference between stages of the recipe is required to produce the correct semantic analysis. This is a chicken and egg problem, both tasks need to be performed together, and this can be resolved through use of the Expectation Maximization (EM) algorithm. Semantic knowledge will be found using probabilistic models, for example counting the percentage of times a word is used in a location or the number of times a food has a certain action performed on it.

2 Recipe preprocessing

Firstly, each recipe R is segmented into a list of actions $E_R = \{e_1 : (v_1, a_{1j}), ..., e_n : (v_n, a_{nj})\}$, where E_R is the action list for each recipe R, e_i is an action in the recipe R, v_i is the verb for this action and a_i are the arguments associated with this verb v_i. Each argument is defined as $a_{ij} = (t_{ij}^{syn}, t_{ij}^{sem}, S_{ij})$, containing three parts, the syntactic type (t_{ij}^{syn}), semantic type (t_{ij}^{sem}), and the string spans S_{ij}. The string spans are a list of words $S_{ij} = \{s_{ij}^1, ..., s_{ij}^k\}$, where s_{ij}^k is the kth span in the jth argument of verb v_i.

2.1 Syntactic type

We use a simplified system for the syntactic type, with t_{ij}^{syn} allowed to be either $DOBJ$ (direct object) or PP (prepositional phrase). This is determined using existing segmented recipes that pre-define all arguments as the correct syntactic type.

2.1.1 Semantic type

We will use a simple toy example of semantic knowledge, with t_{ij}^{sem} being either $\{food, location, duration, other\}$, which, despite being simple, includes a lot of the important semantic knowledge of recipes. The semantic type will be determined through the use of statistical models.

2.2 Implicit arguments

Implicit arguments were also added if an action did not contain any arguments. If an action does not contain any $DOBJ$ arguments then we will add a $DOBJ$ argument with empty string span S_{ij}. If the action does not contain any PP arguments then we will add a PP argument with empty string span S_{ij}. This assumes that every action in the kitchen takes place in a location and is done on a food, either a raw ingredient or a compound object.

3 Connections

The segmented recipe graph is then ready to add edges, which are connections between the actions. A connection is between o, the origin argument (the processed output of an action or raw ingredient), to a string span S_{ij} of a future action e_i . Each connection includes the syntactic (t^{syn}) and semantic type (t^{sem}), defined to be the same as the origin argument. Thus the connections C are completely defined by the $(o, i, j, k, t^{syn}, t^{sem})$. If there is a connection from a raw ingredient then it will be labeled with origin 0.

3.1 Sequential Baseline

The connections were initialized in a sequential baseline. A connection is made from each action e_i to the first available string span in the next action e_{i+1}. The connection $C = (i, i+1, j, k, t^{syn}, t^{sem})$ for each action in the recipe, where j and k will be 1 if there are no connections from the raw ingredients. It is usually the case that actions follow from one line to the next, making a sequential baseline a good approach, giving a good starting position.

In pre-processing, raw ingredients and locations have been found by a string comparison with the recipe ingredients and location list respectively. The t^{sem} for these connections will be automatically set to $\{food\}$ if they are a raw ingredient and $\{location\}$ if they are locations. The other connections will be learnt during the algorithm.

4 Statistical Models

To improve on the sequential baseline, a likelihood is needed to maximize the probability of a certain set of connections C over all recipes. This is achieved using a variety of different models that encode different aspects of world knowledge about recipes.

We maximize the joint probability over the connections and recipes $P(R, C)$. This can be decomposed as:

$$P(R, C) = P(C)P(R|C)$$

$P(C)$ will be found using the Connection Prior Model and $P(R|C)$ using the Recipe Model explained below.

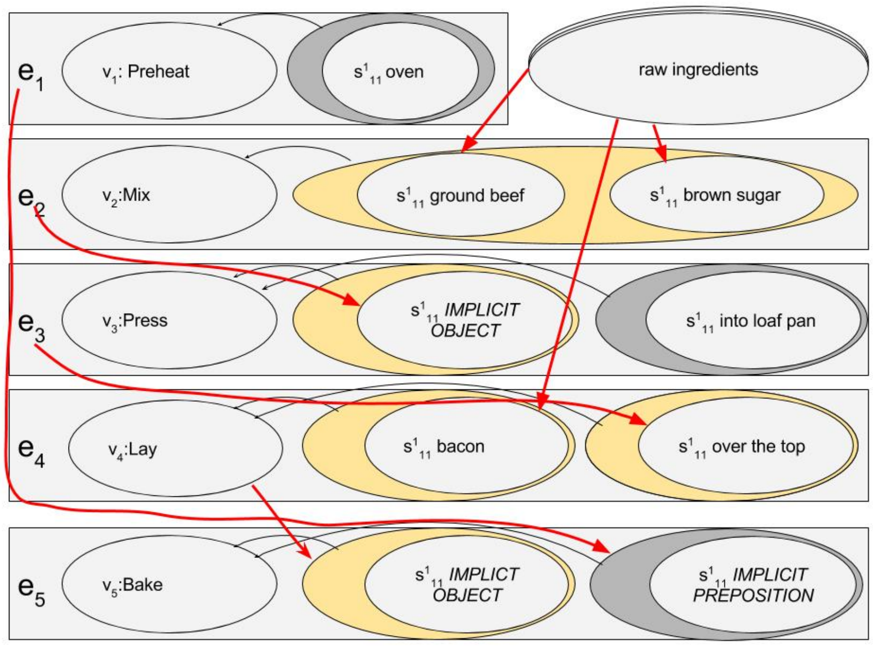


Figure 1: Partial Action graph corresponding to the Amish Meatloaf recipe (http://allrecipes.com/recipe/amish-meatloaf)

4.1 Connection Prior Model

The connection prior model is the main model that calculates $P(C)$. It is consists of the verb signature model and connection origin model.

4.2 Verb Signature Model

The verb signature model calculates the probability that an action contains food arguments of a certain syntactic type, and if it is a leaf or not. The syntactic type set could contain $DOBJ$, PP, Both or Neither, depending on the syntactic type of the food arguments in the action. If the action is the destination of a connection from a raw ingredient (it has an incoming connection with origin

0), the verb signature is considered to be a leaf, otherwise not. This leads to 8 options for the verb signature.

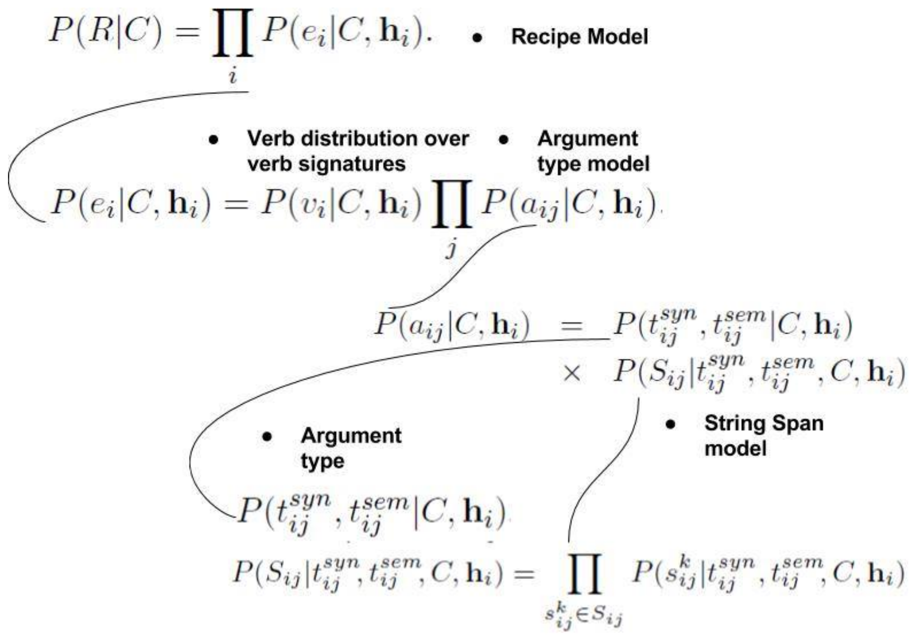
We count how many times each verb signature appears across all recipes and return the probability of a specific action having a specific verb signature.

4.3 Connection Origin Model

We calculate the probability of a connection given all previous actions and connections. We say that an action having a connection to a sequential action is much more probable than connecting to a later action. Also, if an origin has been used in a previous connection, it is much less likely to be used again.

4.4 Recipe Model

Recipe model is the main model that calculates the $P(R|C)$. It consists of argument types model and string span models.



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

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

$$P(a_{ij}|C, \mathbf{h}_i) = P(t_{ij}^{syn}, t_{ij}^{sem}|C, \mathbf{h}_i) \times P(S_{ij}|t_{ij}^{syn}, t_{ij}^{sem}, C, \mathbf{h}_i)$$

$$P(t_{ij}^{syn}, t_{ij}^{sem}|C, \mathbf{h}_i)$$

$$P(S_{ij}|t_{ij}^{syn}, t_{ij}^{sem}, C, \mathbf{h}_i) = \prod_{s_{ij}^k \in S_{ij}} P(s_{ij}^k|t_{ij}^{syn}, t_{ij}^{sem}, C, \mathbf{h}_i)$$

Figure 2: Recipe model is the main model that calculates the $P(R|C)$. It consists of argument types model and string span models.

4.5 Argument Types Model

The argument types model $P(t_{ij}^{syn}, t_{ij}^{sem}|C, h_i)$ returns zero if either the semantic or syntactic types of the connections do not match their destination argument, otherwise it returns 1. This ensures that there is no mismatch between the connections and their destination arguments.

4.6 String Span Model

The string span model returns the probability of a specific string-span given the encapsulating argument type they belong to, the connections that reach them and the connection history (h_i). One of the following three models below is chosen, based on the semantic type, t_{ij}^{sem}, of the incoming connection. The joint probability if there are multiple string spans in an argument are assumed to be independent:

$$P(S_{ij}|t_{ij}^{syn}, t_{ij}^{sem}, C, h_i) = \prod_{k_{ij} \in Sij} P(s_{ij}^k|t_{ij}^{syn}, t_{ij}^{sem}, C, h_i)$$

4.7 Raw Food Model

Raw ingredients are automatically identified as $food$ if they are in the ingredient list for that recipe. We did this by a string matching in ingredient list at the preprocessing part and took no action to recalculate a probability on the fly. The incoming connection has origin 0.

4.8 Part Composite Model

When the incoming connection is $food$, but it is not a raw ingredient (origin $\neq$ 0), the part composite model is used. For this, the number of times each food string is the destination of a connection from another non-origin food string is counted. For example, pesto will more commonly be the destination food string for oil and basil than sugar and strawberry. The part composite model then returns the probability of each $food$ string span $P(s_{ij}^k|food(s_{ij}^k, C))$ given its represented ingredient and the connections.

4.9 Location Model

The locations model $P(S_ij|t_i^{syn}j, t_i^{sem}j, C, h)$ returns the probability of a location being propagated to a specific destination string(connections propagate an argument from the origin verb to a consequent string-span). If the string-span is not implicit (non-empty) then the model uses a string-matching heuristic to determine whether the two locations are the same. It returns 1 if they are and 0 otherwise. Finally, in the case that the destination location string-span is implicit, then we retrieve the probability from a multinomial distribution that indicates how likely the location string span that is propagated to it is, given the verb. $P(verb|location)$. The model uses a counting function to return how many times a verb occurs with said location over all times the verb occurs generally.

5 Local Search

To put all the models together and infer the best set of connections, we use a local search method. Firstly we initialize every connection in the sequential baseline. We use a swap operator that tries random swaps between two connections. If this swap improves the probabilities from our models, $P(R, C)$ we will keep the swapped connections. We then repeat this until convergence. If the probabilities are exactly equal we still keep the swapped connections to increase the number of possibilities. If the probability $P(R, C)$ is worse after the swap, we discard the swapped connections.

5.1 The Zero Heuristic

This was a small addition that we made to the original system. We noticed that, initially many models were returning 0 probability, so $P(R|C)$ remained 0 for many search iterations and the algorithm repeatedly swapped between 0 probability connections. To overcome this issue we added a zero heuristic.

6 Expectation Maximization

We use EM to learn the probabilistic models. After initialization, we use local search to find the current best connections (the E-step), and then recalculate the distributions from the models (the M-step). We loop this until convergence.

7 Results

We ran the Expectaion Maximization algorithm for 30 iterations. The results obtained were similar to the results found in the paper of reference[2].

Table 1 shows the most likely locations that verbs used in. It seems reasonable, many verbs like preheat commonly occur in the obvious location oven whereas others are used in many locations.

Table 2 shows the verb signatures after EM. Some results here seem strange, but others are reasonable. It should be unlikely that the verb signature for cook is a leaf.

The final action graphs (see fig.1 for an example) were usually reasonable, leading us to conclude that the system worked overall.

Verb	**Top location tokens**	
Preheat	oven - 39%	pan - 15%
Cook	pot - 17%	skillet - 13%
Stir	oven - 15%	dish - 15%
Place	oven - 26%	bowl - 13%
Drain	bowl - 19%	pot - 18%
Bring	pot - 18%	bowl - 15%
Mix	oven - 22%	pan - 19%
Pour	oven - 53%	pan - 9%
Remove	bowl - 20%	oven - 12%
Cover	oven - 24%	pot - 17%

Table 1: The top location tokens for verbs. The percentage represents how often the verb has a location token

Verb	**Top verb signature**	**(%)**
Stir	{DOBJ, PP} - 36%	{PP}:leaf - 13%
Cook	{}:leaf - 29%	{DOBJ}:leaf - 25%
Add	{DOBJ}:leaf - 33%	{DOBJ, PP} - 30%
Place	{DOBJ}:leaf - 38%	{}:leaf - 21%
Mix	{DOBJ}:leaf - 37%	{DOBJ, PP} - 25%
Drain	{}:leaf - 33%	{DOBJ, PP} - 22%
Bake	{DOBJ} - 63%	{}:leaf - 11%
Bring	{}:leaf - 44%	{DOBJ, PP} - 25%
Pour	{DOBJ, PP} - 37%	{DOBJ} - 21%
Preheat	{}:leaf - 88%	{PP} - 6%

Table 2: The top verb signature for verbs. The percentage represents how often the verb has a particular verb signature

8 Conclusion

The model we implemented provided better results than the baseline. We made a number of simplification adjustments such as the raw food argument string matching.

An EM approach for mapping instructional language into action graphs could be extended into other domains, such as construction, farming and logistics. It could provide an new way to provide an interface between humans and computers. Segmented annotated text is needed to teach the system for each area of language, but the same algorithm could be used.

[2]Chloe Kiddon, Ganesa Thandavam Ponnuraj, Luke Zettlemoyer & Yejin Choi. (2015) *Mise en Place: Unsupervised Interpretation of Instructional Recipes.* In EMNLP, pages 982-992.

One of the challenges we faced was debugging the software for this model. Given the statistical nature of the problem, it is not always easy to tell when a returned probability is the result of a logical or otherwise error in the code.

Finally the data gathered from the system could be used to perform different everyday tasks, like finding a recipe with tomatoes, or a stew without tomatoes. It could also be used to give suggestions of meals based on set ingredients.

References

[1] Chloe Kiddon, Ganesa Thandavam Ponnuraj, Luke Zettlemoyer & Yejin Choi. (2015) *Mise en Place: Unsupervised Interpretation of Instructional Recipes*. In EMNLP, pages 982-992.