Flow graph construction from unsupervised cooking recipes data

Oleg Grinchuk (grinchuk@mit.edu),
Aizhan Ibraimova (aijan@mit.edu),
Elena Shirokova (elenashi@mit.edu)

I. Abstract

Thousands of cooking recipes can be found on the web in a form of unstructured text. In this project we use this large unsupervised corpus of data to build a model that transforms recipe-for-human into a sequence of strict instructions (states), which can be executed by a robot. That defines what actions should be performed in what order, on which objects and with the help of which subject, taking into account the quantity of used ingredients. We use semantic role labeling for initial approximation to the states and then improve the baseline by using knowledge from dictionaries and by cleaning the predictions in a smart way. We also train a probabilistic model to build a statistical prediction and compare it with the SRL approach. Code is located at [8].

II. Introduction

There are lots of cooking recipes available online. They are written by humans for humans, and here comes a drawback - they cannot be recognized by automatic systems. . Hence, converting unstructured recipe text to a sequence of instructions can be useful in many applications. However, there is relatively little effort to design algorithms that can transform text to instructions. One of the approaches proposed by [1] is to create a special instructional language which is easy to interpret and where each sentence is splitted to the set of commands and then the multilabel classification applied. Another technique in [3] is based on dependency tree parser which converts recipe to tree structure. The main method in [2] is based on Hidden Markov Models. In addition, there are some machine learning approaches like Named Entity Recognition, mentioned in [4],[5]. In [6] the problem is solved by namely predicting the ordering of events based only on the identity of the words comprising their predicates and arguments.

In this paper we propose our approach to the problem of flow graph construction for cooking recipes. We use new language for robots in a form of sequence of states and we build those states by applying natural language processing techniques to the ininial raw recipe data. We introduce evaluation metric and show that our approach increases the performance compared to the baseline.

III. Model

We state the main goal ss to build a model, which transforms a human-written recipe into a sequence of clearly defined steps. As an input, system takes the list of directions

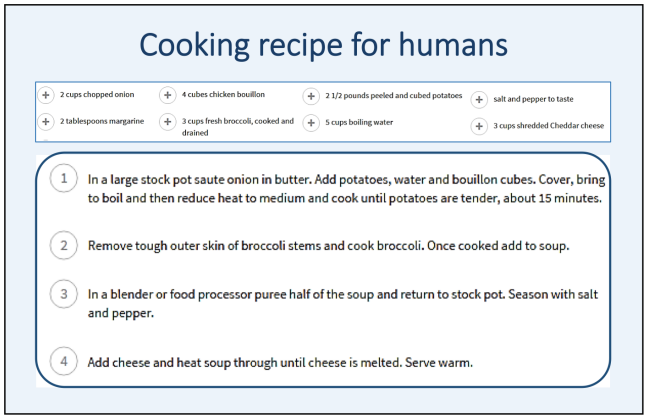

Fig. 1. The example of the raw recipe, taken from allrecipes.com

and list of ingredients in form of just strings. The output is presented as a sequence of states. Define *recipe state* S as following:
State=(ACTION,#A, OBJECT, #O, TARGET, #T),
where

- ACTION - a command from predefined set of commands, which robot can execute. If ACTION requires timing, #A displays the time in minutes, otherwise # A = 0 (undefined).
- OBJECT is some ingredient or utensil, which can be potentially used by ACTION.
- TARGET also comes from ingredients/utensils, but normally it represents the target of the ACTION.

I.e ACTION is applied to #T units of TARGET using #O units of OBJECT during #A mins. Intuitively, question Where is connected to TARGET and question What - to OBJECT. State parts are taken from dictionaries: OBJECT/TARGET from {ingredients}, {utensils}; ACTION from {all-possible-actions}. So, the total number of states is large, but finite.

Let's consider some examples:

Example 1:

Input:

Rub olive oil onto the outside of each sweet potato and sprinkle sea salt over each.

Output:

1. RUB, olive oil, sweet potato(4) ⇒
2. SPRINKLE, sea salt(1 teaspoon), sweet potato(4)

P.S.Here we get number 4 from ingredients list

Example 2:

Input:

Preheat oven to 400 degrees F

Output:

PREHEAT, oven(400 F), ∅

A. Challenges

This task poses unique challenges for semantic analysis. Despite the fact that input texts are noisy, words can be written with synonyms, slang, etc., there is another huge problem - co-reference resolution. This means that object or target is often omitted (e.g. "Bake for 50 minutes") and we need to reconstruct the context from previous sentences. Missed object is referred to some state before, but it is non-trivial task to recognize the exact state.

Next, as the recipe flow processes, one can use hypernyms to describe some combined ingredient (*'Put lemon mixture on top of prepared bird'*). We need to recognize such cases too.

The last, but not least - recipes almost always are written in imperative form. However, all pretrained NLP models use declarative sentences. This means that such models will score not really well on recipe data, so we need to either retrain the models or to adapt them by applying some extra algorithms. Since we have no labeled data to train our own model which requires supervision, we'll focus on the second approach.

B. Data

Our dataset consists of 70000 raw English recipes, which were downloaded from the website http://allrecipes.com.

A recipe consists of the list of foods used as ingredients and text describing the step-by-step instructions on how the dish can be cooked. In this project, we focus on the text part (let's call it a body of the recipe) from which the work flow should be constructed. In spite of this, we still use ingredients part for getting the quantity of each object or target and for improving the baseline algorithm.

A single recipe in the dataset is a file in json format with following fields:

- "recipe title"
- "recipe id"
- "review count"
- "ingr"
- "ingr id"
- "cook time"
- "directions".

"recipe title" and "recipe id" are unique for a single recipe, "review count" shows the number of comments written by another users (it can be one of the features of the popularity of a recipe). "ingr" is the list of ingredients used in the cooking process for this recipe and "ingr id" are unigue identification numbers corresponding for each of the ingredients in "ingr" (i.e. it is the list of the same size as "ingr"). "cook time" is the list of 3 elements corresponding to the preparation time, the baking time and the whole time of the cooking the recipe. "directions" are the list of elements, step-by-step instructions from the text part of the recipe.

IV. Proposed approach

A. Dictionaries

Since we can not always identify an action or an object by a single word, we've built four different dictionaries of words and word sequences which correspond to the actions, ingredients, utensils and units of measurement in a cooking world. All dictionaries' elements are chosen based on the ingredients and text parts of all recipes from the dataset. The figure 2 shows the random recipe which was labelled according to these dictionaries.

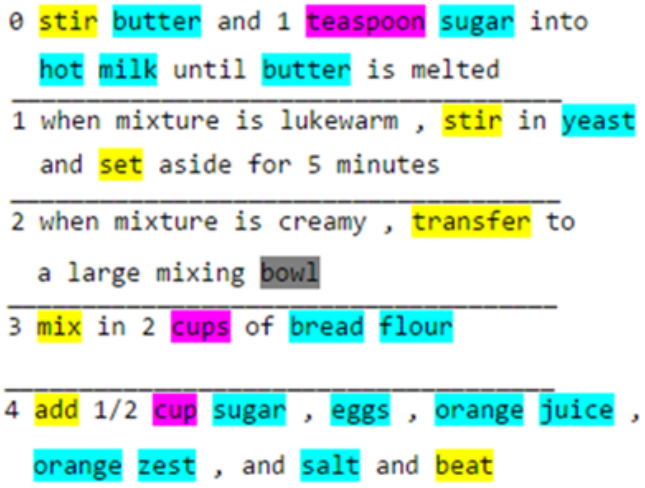

Fig. 2. The highlighted (labeled) recipe

B. Baseline

As a baseline, we applied a state-of-the art SRL system (Das et al., 2014) to the corpus. Semantic role labeling tells us the semantic structure of the sentence. If it was predicted correctly, then we can map those structure to our state. For instanse, 'location' role most likely corresponds to the target, verb in an infinitive form ("add", "stir", etc.) corresponds to the action.

C. Advanced method

The general text used to train the NLP modules has different structure from the text of a recipe. Therefore, it is important to perform the adaptation of usual NLP module. While using the usual SRL method, we don't take into account that we work

with recipes that usually have some structure. For instance, it's unlikely that a verb in the past tense appears in a body of a recipe, there is also no modality problems. Recipes are usually straightforward. Here we represent all events by the so-called "states" (action + object + target + the number of objects + the number of targets). But now we know that objects and targets can be either the elements of the earlier built dictionary "utensils" or the name of the ingredient which can be parsed from the ingredients part of the recipe. One state has no more than one object and one target, so we also split compound ones and created more states with the same action. For instance, the sentence "Add flour and salt" should be related to the states:

- add, flour, body
- add, salt, body

Furthermore, to create correct states, we deal with alternatives (example: "Add 1 lemon or lime"). Here we can choose one of the possible variants:

- add, lemon, body, 1, 0
- add, lime, body, 1, 0,

where "1" corresponds to the number of objects and "0" corresponds to the number of targets (i.e. we mean by "0" that body is the subject taken from the previous step and we don't care how much should we take it - we need to take everything created one step before). Considering all the unique features that recipes have and modifying the SRL, we improved the results (see Evaluation part).

Figures 3 and 4 demonstrate the example of the algorithm. The first figure shows the hand-labeled random taken recipe and the second figure shows the output of the advanced algorithm for the same recipe.

line_id	action	object	target	no	nt
0	combine	cheese	body	1/2 cup	0
0	combine	pepper	body	3/4 teaspoon	0
0	combine	garlic powder	body	1/2 teaspoon	0
1	unfold	pastry sheets	cutting board	1 (17.5 ounce)	0
2	brush	pastry sheets	egg	1 (17.5 ounce)	0
3	sprinkle	body	sheet	0	0
3	turn over	body	0	0	0
4	press	body	pastry	0	0
4	turn over	body	0	0	0
5	repeat	body	0	0	0
6	cut	body	strips	0	0
7	twist	body	0	0	0
8	place	body	sheet	0	0
8	bake	body	oven	0	350 f

Fig. 3. The hand-labeled recipe

line_id	action	object	target	no	nt
0	combine	cheese	body	1/2 cup	0
0	combine	pepper	body	3/4 teaspoon	0
0	combine	garlic powder	body	1/2 teaspoon	0
1	unfold	pastry sheets	cutting board	0	0
1	cutting	board	body	0	0
2	brush	body	egg white	0	1
3	sprinkle	cheese	body	1/4	0
4	press	body	pastry	0	0
4	turn over	body	0	0	0
6	cut	sheet	body	0	0
8	place	body	sheet	0	350 175 degrees

Fig. 4. The robot-format output of the algorithm

D. Bayesian Inference

In addition to Semantic Role Labeing, we also implemented a probabilistic model, based on bayesian approach. For the sake of simplicity, we assume that both object and target are action-dependent. These connections are shown in Figure 5. Then the probabilistic model can be written as

$$P(sent) = P(action)P(object|action)P(verb|action)$$

Here we denote conditional probabilites from the bag-bigram models. Fractional counts are calculated as all occurencies of object/target and action in one recipe direction, so

$$P(object|action) = \frac{count(object, action)}{count(action)}$$

This can be written for target as well.

Therefore, the likelihood of the whole recipe:

$$P(recipe) = \prod_{i=1}^{n} P(sent_i)$$

For the whole sentence we maximize likelihood. This means that for the current sentence and for current action we choose object and target with the highest probability. In this way we can generate states for every direction from the recipe. The example is shown in Figure 6.

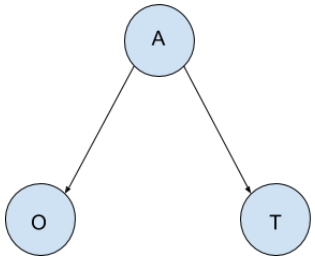

Fig. 5. Graphical model

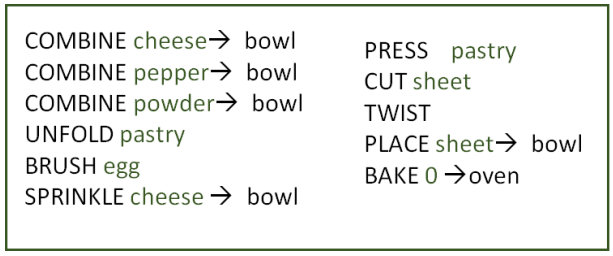

Fig. 6. The example of generated instruction using probabilistic model

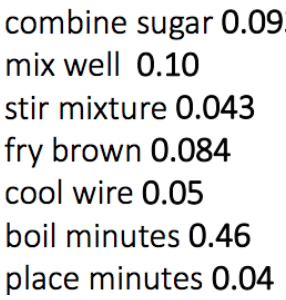

Fig. 7. Examples of some frequent patterns

This approach works pretty well for simple recipes, but for more complex ones it makes errors, especially in target field. However, we didn't apply any cleaning, dictionary learning or postprocessing to this model, so it can still be improved.

V. Evaluation

Since we deal with unsupervised data, evaluating a quality of some prediction is not a trivial task. We hand-labeled about 70 recipes, which allowed us to introduce a scoring function. Define Ea as fraction of correctly predicted Actions through recipe and E - fraction of correctly predicted States. Calculating these metrics for each labeled recipe and then taking mean, we obtained the following results:

Model	Ea	E
Baseline (SRL)	0.77	0.34
SRL+Dict	0.80	0.41
SRL+Dict+Cleaning	0.80	0.44

Actions were correctly recognized in majority of cases, the problem is more with detecting objects and targets. Straightforward Semantic Role Labeling gives only 0.34 accuracy in whole-state prediction. But after applying some common sence concepts, string cleaning and knowledge from dictionaries, we managed to get a 30 percentage relative increase in accuracy - up to 0.44.

VI. Conclusion

Automatic flow graph construction from unsupervised raw texts implies a lot of difficulties we need to handle with. However, we were able to obtain some pretty good results,evaluated on a small subset of labeled data. We used semantic role labeling to build a set of instructions from the sentence and then improved this baseline score by applying different post-processing functions. We also collected various dictionaries, which helped us to improve results a bit more. We tried to apply Bayesian inference model and it showed promising results but slightly worse than the main model. However, it has a huge possibility to be improved. Hence the task of building recipe flow contains a lot of difficulties, we showed that some of these problems can be solved separately.

References

[1] Dan Tasse and Noah A. Smith, *SOUR CREAM: Toward Semantic Processing of Recipes*, School of Computer Science Carnegie Mellon University, 2008.

[2] Jon Malmaud, Earl J. Wagner, Nancy Chang, Kevin Murphy, *Cooking with Semantics*.

[3] Jermsak Jermsurawong and Nizar Habash, *Predicting the Structure of Cooking Recipes*, New York University Abu Dhabi.

[4] Shinsuke Mori, Hirokuni Maeta, Yoko Yamakata, Tetsuro Sasada, *Flow Graph Corpus from Recipe Texts*, Kyoto University.

[5] Chloe Kiddon , Ganesa Thandavam Ponnuraj, Luke Zettlemoyer and Yejin Choia, *Mise en Place: Unsupervised Interpretation of Instructional Recipes*.

[6] Omri Abend, Shay B. Cohen and Mark Steedman, *Lexical Event Ordering with an Edge-Factored Model*.

[7] Luke S. Zettlemoyer and Michael Collins, *Learning to Map Sentences to Logical Form: Structured Classification with Probabilistic Categorial Grammars*, MIT CSAIL.

[8] https://github.com/oleggrinch/recipes