Constructing Action Graphs for Material Synthesis Procedures

Joseph Kim, Dongyoung Kim

6.864 Final Report

December 14th, 2015

Abstract

There exists a significant bottleneck in deploying task plans directly from natural language instructions. Often additional effort is required to create formatted plan representations. There is a growing interest in reducing the translation burden by automatically generating action graphs, which summarize the text into a sequence of ordered actions. In this project, we apply several supervised methods to construct action graphs for the domain of materials synthesis. We investigate the subproblems of identifying key actions, extracting arguments (i.e., inputs and outputs of actions), and refining the global order.

I. Introduction

In material science, a researcher often has to search through numerous scholarly articles to discover a synthesis procedure for a certain material compound. It would be useful to have a database where the compound can be queried and its synthesis procedure can be visualized. One intuitive way to visualize procedures is through action graphs, which parse text descriptions into sequences of actions. Such a task falls under the growing research area of "instruction interpretation" where various learning models have been proposed to interpret cooking recipes (Mori et al., 2012), (Kiddon et al., 2015), (Maeta et al, 2015), game tutorials (Branavan et al., 2009), how-to instructions (Lau et al., 2009), and navigation instructions (Chen and Mooney, 2011). Unlike the aforementioned examples, synthesis procedures in material science literature are significantly more verbose, complex, and contain increased noise in the raw text. In this project, we explore how the instruction interpretation task generalizes to the material synthesis domain.

II. Related Work

Related work for mapping instructions to an actionable form include supervised methods where labeled examples are provided, or a semantic analyzer to test the understanding of the instructions is available (Maeta et al, 2015), (Bollini et al., 2013). Unsupervised methods attempt to uncover actions from a large amount of data, but they are limited to solving a specific set of subproblems (i.e., segmentation and coreference) (Kiddon et al., 2015). Unsupervised methods have only been applied to cooking recipes where a large database is easily accessible (Kiddon et al., 2015). Meanwhile, reinforcement learning assumes an access to a "reward-giver" that defines the quality of sequences of actions. Such a framework has been applied in video games where rewards can be directly observed from the environment (Branavan et al., 2009). Learning from demonstration have been used to interpret navigation instructions after analyzing observations of humans following such instructions (Chen and Mooney, 2011).

Material synthesis procedures does not fit well in either reinforcement learning or learning from demonstration framework because having an access to reward-givers or external observations would be impractical. Learning would need to occur using text alone. An unsupervised approach would be challenging due to the sparsity of the papers in the field. In other words, individual articles generally describe a synthesis procedure for a unique compound. The field is notably different than that of cooking recipes, where multiple variations of identical title compounds are available (For example, there are over 400 published instructions on "macaroni and cheese" recipe alone on allrecipes.com). It may be difficult to extract high-level relationships when compounds themselves are all unique. For this project, we explored supervised methods for instruction interpretation. It was also appropriate given the small size of our dataset. We anticipate the feasibility of unsupervised approaches by leveraging numerous scholarly articles, but this was left as part of future work.

III. Problem Statement

The input to our problem is a block of text, *d*, taken directly from a "Synthesis" or "Experimental Procedure" subsection in the papers. An example block is given below:

(Zn(1-x),Mgx)2(Ti0.88,Sn0.12)O4 (x = 0.05, 0.10, 0.15 and 0.20) ceramics were prepared by solid state reaction method with ZnO (>99.7%), MgO (>99.0%), TiO2 (>99.9%) and SnO2 (>99.9%) as the starting materials. According to composition of (1-x) ZnO-xMgO-0.88TiO2-0.12SnO2 (x = 0.05, 0.10, 0.15 and 0.20), MgO, ZnO and SnO2 powder was weighed and mixed using planetary milling with zirconia balls in ethanol for 12 h. The mixture was dried and calcined at 800 #C for 2 h. The calcined powder was milled for 4 h and dried. The resultant powder was granulated using a 5 wt% poly vinyl alcohol (PVA) solution and pelleted to 12 mm diameter and 1.2 mm thick disks. The ceramics were prepared by sintering at temperatures from 900 to 1200 #C for 4 h with a heating rate 10 #C/min and cooled to room temperature in the furnace.

Given *d*, the desired output is a sequence of actions, ***a*** = (*a_0, ... a_n-1*), where each action *a_i* = (*c,**I**,o*) is a tuple encompassing a command *c*, inputs ***I***, and output *o*. While the input contains a set, we assume that output is a single variable (this was a valid assumption for our dataset). An example sentence, *"the mixture was dried and calcined at 800 #C for 2h"* would correspond to a sequence of *a_1* = ('dry' , mixture **,** [undefined])*, a_2* = ('calcine', [undefined], [undefined]). The [undefined] tags represent implicit arguments. They generally refer to the solution as a result of the previous action. Additionally, to account for extraneous verb phrases that do not describe relevant actions, *c* can be null. Null commands occur frequently in synthesis procedures, where they provide additional descriptors and references.

IV. Dataset

The dataset is a collection of 23 synthesis paragraphs taken from the material science literature. The title compounds are all unique and differ in subfields from metallics, nanocrystals, magnetic materials, semiconductors, and more. Along with the text, fully annotated graphs were provided. Each procedure consisted of approximately 8 full length sentences with average size of 25 verbs. Furthermore, there were 11 noun nodes on average per procedure, in which approximately 2 of them were implicit noun nodes. Meanwhile the annotated graphs on average contained 6 action nodes. This highlights a challenge where there are many irrelevant verb phrases in the raw paragraph.

V. Approach

We dissociated our problem into a cascade of subproblems. We first detected the action units from the raw paragraph. Given the detected action units, we identified its inputs and output (thereby composing a full action tuple). Finally, with the list of action tuples, we applied a refinement model to determine the global ordering. Such a dissociation may lose useful joint relationship across the variables. We leave the exploration of full joint model for future work.

VI: Preprocessing

The raw paragraphs contained lots of noise. Employing off-the-shelf tokenizers and parsers did not perform well over the non-alphanumeric characters and the complex vocabulary such as chemical formula and units. We preprocessed the raw text in order to remove the incoming noise as much as possible. We suspected that this would increase the efficacy of the subsequent parsers. The list below highlights our preprocessing steps.

1. Removal of descriptions in closed parentheses and in closed brackets
2. Removal of units such as '°C', 'wt.%', '#C/min'
3. Replacement of numeric ratios and ranges (e.g., 0.1/0.2, 5.0~5.7) to a token of NUMRATIO
4. Segmentation of paragraph into a list of sentences
5. Named entity recognition (NER) to substitute chemical formula to a token of COMPOUND

For step 4, we used the Stanford named entity recognizer (NER) to replace the chemical formulae into temporary placeholders labeled 'COMPOUND.' In our analysis, the base NER was only able to detect 62% of the true chemical formulae. False negative examples included formulae which contained non-alphanumeric characters: 'Ba(NO3)2', 'Ce(NO3)2*6H20', and complex titles such as 'Dy3+-dopedCaMoO4.' In order to improve the formulae detection, we created a list of all elements from the periodic table and employed a rule that checked if a word was composed of at least 40% by elements. We also checked if non-alphanumeric characters were present. The result increased recall to 0.94. Table 1 shows the result of the preprocessor.

Table 1. Result of preprocessing

Raw Sentence	Preprocessed Sentence
Cube-shaped crystals of Yb2CuGe6 formed from the reaction with small amount of copper (2 mmol) and mixture of Yb2CuGe6 and rod-shaped Yb3Cu4Ge4 crystals were obtained from the reaction with high amount of copper (3 mmol).	Cube-shaped crystals of COMPOUND formed from the reaction with small amount of copper and mixture of COMPOUND and rod-shaped COMPOUND crystals were obtained from the reaction with high amount of copper .
A suspension of 400 mg VO2(A) powders in 50 ml 0.05 M Ba(NO3)2 solution was sealed in a quartz ampule followed by hydrothermal treatment in an autoclave at 350 #C for 40 h.	A suspension of 400 COMPOUND powders in 50 0.05 M COMPOUND solution was sealed in a quartz ampule followed by hydrothermal treatment in an autoclave at 350 for 40 .

VII: Main Action (Command) Detection

The ground truth (gold) graphs only contained actions that characterize the synthesis process. All other descriptors were pruned, which highlighted the summarization purpose. From the dataset, we noticed that actions were almost always verbs, and were often written in the past participle form (i.e., 'was *mixed'*, 'was *dried'*). Our first hypothesis was that these action verbs can be readily extracted from the standard constituency-based parse trees. Thus, the first baseline model employed the following heuristics:

- Instantiate a constituency-based parse tree and extract the highest past participle verb (VBN)
- If multiple VBNs exists on the same height (typically connected by a conjunction), extract those as well.

Meanwhile, the second baseline model employed a dependency parser and extracted the ROOT words as actions. The third baseline model, which was the most naive approach, extracted all verbs in the given sentence and considered all of them as relevant actions.

In the supervised learning model, we used a MaxEnt classifier with the following feature sets:

1. Verb identifiers (one-hot vector with a size of $|V|$). The vocabulary of verbs were collected during each training folds.
2. Parts-of-speech (POS) tags. The average tag set size during each training fold was 10 and typically consisted of { JJ, NN, NNS, RP, VB, VBD, VBG, VBN, VBP, VBZ}.
3. Universal dependency labels. We extracted the labels of verbs as child words. When given sample phrase like *'..was processed by heating,'* dependency labels characterize the relationship from 'heating' to 'processed.' The size of the label set was about 10 and consisted of {advcl, advmod, ccomp, conj, csbujpass, dep, nmod, nsubj, nsubjpass, xcomp}.

Since gold verbs provided in present tense, extracted verbs were all processed through a NTLK Lemmatizer to reduce them into present tense form. Due to small size of our dataset, we performed 10-fold cross validation in order to increase the number of training data per fold.

VII-A: Model Performance and Discussion

Table 2. Action detection performance.

	Precision	Recall	F1 Score
Baseline #1: Constituency-based parse: heuristics	0.378	0.649	0.478
Baseline #2: Dependency parse: root word	0.371	0.287	0.324

Baseline #3: All verbs	0.201	0.874	0.327
MaxEnt Model	**0.793**	**0.874**	**0.831**

Looking at Baseline #3 in Table 2, extracting all verbs maximized recall at 0.874, but suffered from low precision of 0.201 (i.e., too many false positive verbs). Extracting root words from dependency parses (Baseline #2) performed badly as well. Baseline #1 with heuristic rules performed slightly better than the other two baselines, but still demonstrated low performance with F1 score of 0.478. The supervised MaxEnt model essentially employed feature sets motivated from all baselines and performed reasonably well with F1 score of 0.831. Recall was comparable to that of extracting all verbs.

In order to further explore the information contained within features, we evaluated models trained separately by individual feature sets. The cross-results are shown in Table 3 where the first three rows show performance with individual feature sets and last three show the effect of removing individual ones from the full model. The result demonstrated that verb indicators carry the most informative signal. There seems to be a standard collection of verbs used to describe a synthesis procedure. (i.e, positive examples = {mix, heat, cool, flame-seal, dissolve, etc..} negative examples = {process, collect, keep, prepare, etc..} The negative examples tend to be more 'general'). There was a significant performance reduction when the verb indicators were removed from the model. Model with only POS tags were not able to detect any of the gold verbs. Dependency labels by themselves only appeared to provide a small amount of predictive signal.

Table 3. Action detection performance by feature sets.

	Precision	Recall	F1 Score
MaxEnt: 1) Verb indicators only	0.793	0.860	0.825
MaxEnt: 2) POS tags only	undefined	0.00	undefined
MaxEnt: 3) Dependency labels only	0.357	0.556	0.435
MaxEnt: 1) Verb indices removed	0.314	0.571	0.406
MaxEnt: 2) POS tags removed	0.800	0.868	0.833
MaxEnt: 3) Dependency labels removed	0.793	0.867	0.828

VII-B: Challenges

One of the difficulties in action detection resulted from annotators inherent labeling process. In the dataset, several gold verbs did not appear in the raw text. These cases occurred where annotators had substituted certain phrases into verbs with similar meaning. For example, when given the following sentence: *"To obtain LiFePO4/C composites, the precursor powders were well mixed with glucose and treated at 975 K and 1025 K at Ar-4%H2 atmosphere,"* our detector was able to extract the actions 'mix' and 'treat.' However, the annotator had labeled 'treat' as 'heat' even when the word 'heat' did not appear in the original paragraph. What happened was that the phrase, 'treated at 975 K and 1025 K,' was substituted as 'heat'. Such substitution examples occurred frequently, including cases like 'prepared from reaction mixtures' → 'mix', 'ball mix' → 'mill', and 'ignite' → 'heat.' This highlighted the difficulty replicating the summarization behavior of domain experts . An intelligent learner would need to be able to extract the knowledge that 'treat at [temperature]' maps to an action called 'heat.' Possible ways to do this could involve using word similarity measures to replace non-standard verbs to one closely resembling the vocabulary of the training set. Word2vec could be a resource in performing such translation. Other learning methods on clustering phrases and semantic role labeling could be leveraged.

VIII: Input and Output Identification

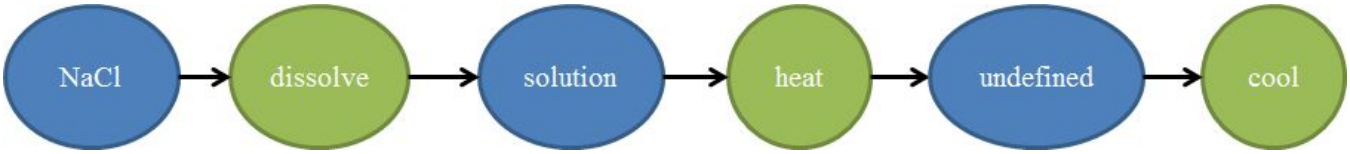

Figure 1. Sample action graph.

The main goal of Input and Output Identification is to extract the edges of the action graph. The edges represent verb-noun relationships. Consider the following procedure: '*Dissolve the NaCl into a solution. Heat it for 30 minutes and then let it cool.'* Representation for the sentence is depicted as an action graph in Figure 1. There are two types of edges: the Argument-Verb edge and Verb-Product edge. In the example above, (NaCl, solution) represents the Argument-Verb edge and (dissolve, solution) represents the Verb-Product edge. To extract the edges, we must infer whether either type of relationship exists given any pair of word within a sentence.

Furthermore, one must consider the presence of implicit nouns in the text. Continuing with the same example as above, note that the argument solution is heated and subsequently cooled. Since 'heat' and 'cool' are sequential actions, there must exist an intermediary substance even though it is not explicitly mentioned in the text. Apart from extracting edges, it is also important to identify these implicit nodes as well.

VIII-A: Model

The goal of Input Output Identification is to infer whether there exists an edge relationship between any given two words within a sentence. The model setup we employed was to approach this as a binary classification problem. The inputs to our model were the entire procedure text and a pair of words within the text. Given the considered words and its context, we generated features based on the dependency parse tree structures, and then used the MaxEnt model for the classification algorithm. Since we have two states of interest, which are Argument-Verb and Verb-Product, we will use a separate MaxEnt classifier for each.

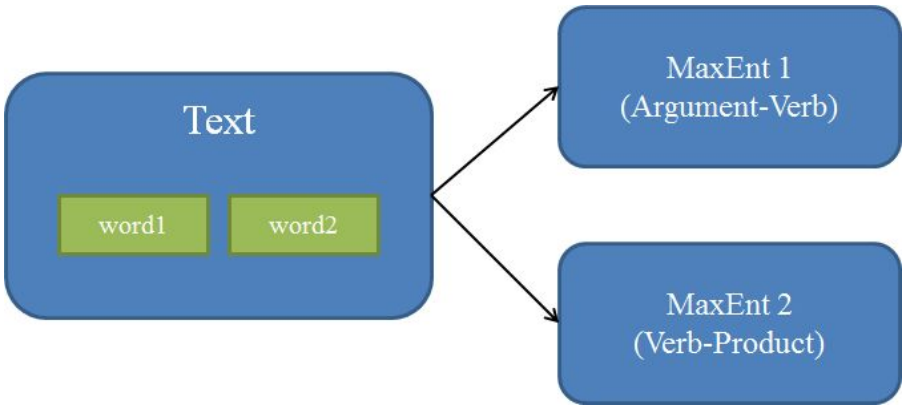

Figure 2. We utilized two separate MaxEnt models for the classification task. The inputs to each model consists of samples of word pairs, and its corresponding context.

The input features for MaxEnt consisted of true/false binary categorical data. The feature matrix included distinct columns for each parts-of-speech tags, as well as information about dependency parse tree structure. Example features are shown in the table below. Furthermore, in order to detect implicit nouns, we trained and tested for word pair samples where only one word was an existing word in the text and the other was tagged as 'undefined'. For such words, all boolean columns were labeled as false except for a single column which labeled if the word was assumed to be implicit. For features that represent cardinal values (e.g. the offset of two sentences that each contains word1 and word2), we bucketed each cardinal quantities into distinct boolean columns.This was due to the fact that implicit nouns will have undefined values for cardinal features,

and hence we should have categorical columns to represent data samples that are undefined in the cardinal feature space. For example, instead of having a single integer column that represents the sentence offset, we have multiple columns for each value of sentence offsets.

Table 4. Examples of features used in the MaxEnt models.

Edge type	**Features**
Argument-Verb	Parts-of-speech tag Offset of container sentence indices Is word1 parent of word2 within the parse tree? (vice versa) Is word a named entity? Word's position in sentence. Is word implicit?
Verb-Product	Parts-of-speech tag Offset of container sentences indices Is word the root of the tree? Is word second level of the tree? Does the word have a direct child node that is a verb? Does the word have a subtree? Is word implicit?

To prepare the data for training and testing, we sampled pairs of words from the procedural text. First, we included all word pairs that represented true edges. In our dataset, there were 173 Verb-Product edges and 235 Argument-Verb edges. Then, we randomly sampled word pairs from the corpus to represent the negative samples that do not represent either type of edges. Negative samples with a random word in the corpus paired with an implicit noun were included as well. The total number of such negative samples were 10,000 for each MaxEnt models and the respective feature space. To balance the number of positive and negative samples, we duplicated the positive samples so that the ratio of positive to negative samples became 1 to 1. The training set and test set was again selected randomly maintaining the same ratio.

VIII-B: Results

For Input and Output Identification, we used a simple baseline heuristic to benchmark our results. For a given word pair, we assumed that the Argument-Action relationship holds for any verb-noun.

To evaluate our MaxEnt model, we first ran tenfold cross-validation over the dataset where ratio of positive and negative samples were set to be equal. The results of the MaxEnt against the baseline is found below.

Table 5. F1 score results on dataset that have been equally balanced in number of positive and negative samples.

Argument-Action	Precision	Recall	F1 Score
Baseline heuristic	0.014	0.7395	0.028
MaxEnt	0.89	0.92	0.90
Action-Product	Precision	Recall	F1 Score
Baseline heuristic	0.008	0.28	0.016
MaxEnt	0.91	0.82	0.87

While the MaxEnt model worked very well on the equally balanced dataset, one must consider that in a practical setting, there are much fewer true edges than the number of dummy pair of words in a typical chemical recipe text. In fact, the ratio of true edges within the raw word pair dataset was 0.06% and 0.12% for Argument-Action and Action-Product, respectively. Since our model is trained on a equally balanced dataset and therefore overweights the probability of a positive sample, precision will suffer in practical datasets where positive samples are sparse. However, the majority of the false-positives that are given by the model in such settings is the case where the two words actually do represent a linguistic verb-argument relationship but nonetheless labeled as negative because the corresponding words were not the key action verb of interest. If we assume that the action verbs are perfectly pre-detected, which is a standalone problem we try to solve in the previous section, we can remove the majority of the false positives and therefore enhance the precision even in settings where number of positive samples are extremely sparse. Below are the results for datasets that represent biased ratio of positive to negative samples, and with the assumption that the key action verbs were perfectly pre-detected.

Table 6. Precision results on datasets that have biased ratio of positive and negative samples

Argument-Action		**Action-Product**	
Baseline	0.062	Baseline	0.033
MaxEnt	0.33	MaxEnt	0.26

After filtering out non key action verbs, the precision of the predicted positive samples became 33.43% and 26.09% for each respective MaxEnt models. While the MaxEnt model still overestimates the probability of a positive sample, the enhancement in precision is notable especially given the extremely biased ratio of the dataset.

VIII-C: Combined results

We investigated the performance of overall action graph after combining results from Action Detection and Input Output Identification. The Action Detection would give the verb nodes in the graph, whereas the Input Output Identification will give the noun nodes and the edges. The training and testing process was implemented such that the model would be tested on a single document and trained on rest of the documents (leave-one-out CV).

Since we discovered that the MaxEnt model overweights positive predictions, we ranked the positive samples when there were too many arguments or products for a single verb. When there were two or more competing Verb-Product positive predicted samples with the same verb, we took the edge with the higher log-probability. Product from the previous verb was automatically assumed to be inputs to the following verb. Additional input arguments were given by the MaxEnt model. When there were too many inputs to a verb, we used top two arguments according to their log-probabilities (or more than two if the log-probabilities were exactly same).

To evaluate the similarity of two trees, we calculated the similarity of nodes and edges separately. The evaluation formulae was given by following:

$$\textit{Node similarity score} = \frac{2 \times \textit{Number of identical nodes}}{\textit{Total number of nodes in ground-truth graph} + \textit{Total number of nodes in predicted graph}}$$

$$\textit{Edge similarity score} = \frac{2 \times \textit{Number of identical edges}}{\textit{Total number of edges in ground truth} + \textit{Total number of edges in predicted graph}}$$

Each score has a scale from 0 to 1 where perfectly identical graphs would give 1. The average similarity scores (n = 23 graphs) for the predicted graphs is reported below:

$$Node\ similarity\ score\ = 0.62$$
$$Edge\ similarity\ score\ = 0.47$$

To visualize the graph generation in action, we produced an example action graph predicted by our model, and visually compared with the ground-truth action graph. An example document and the corresponding action graphs are given below:

The precursor powders of LiFePO4 were prepared by chemical co-precipitation method using stoichiometric LiH2PO4 (Alfa, 97%), FeSO47H2O(Alfa, 99%) and LiOHH2O (Alfa, 98%) as starting materials. The suspension was stirred at 0-4 #C for about 30 min and aged for about 2 h. The final precursor powder product was then filtered, washed with deionized water and ethanol, and dried at 100 #C in a vacuum oven for 2 h. To obtain LiFePO4/C composites, the precursor powders were well mixed with glucose and treated at 975 K and 1025 K at Ar-4%H2 atmosphere.

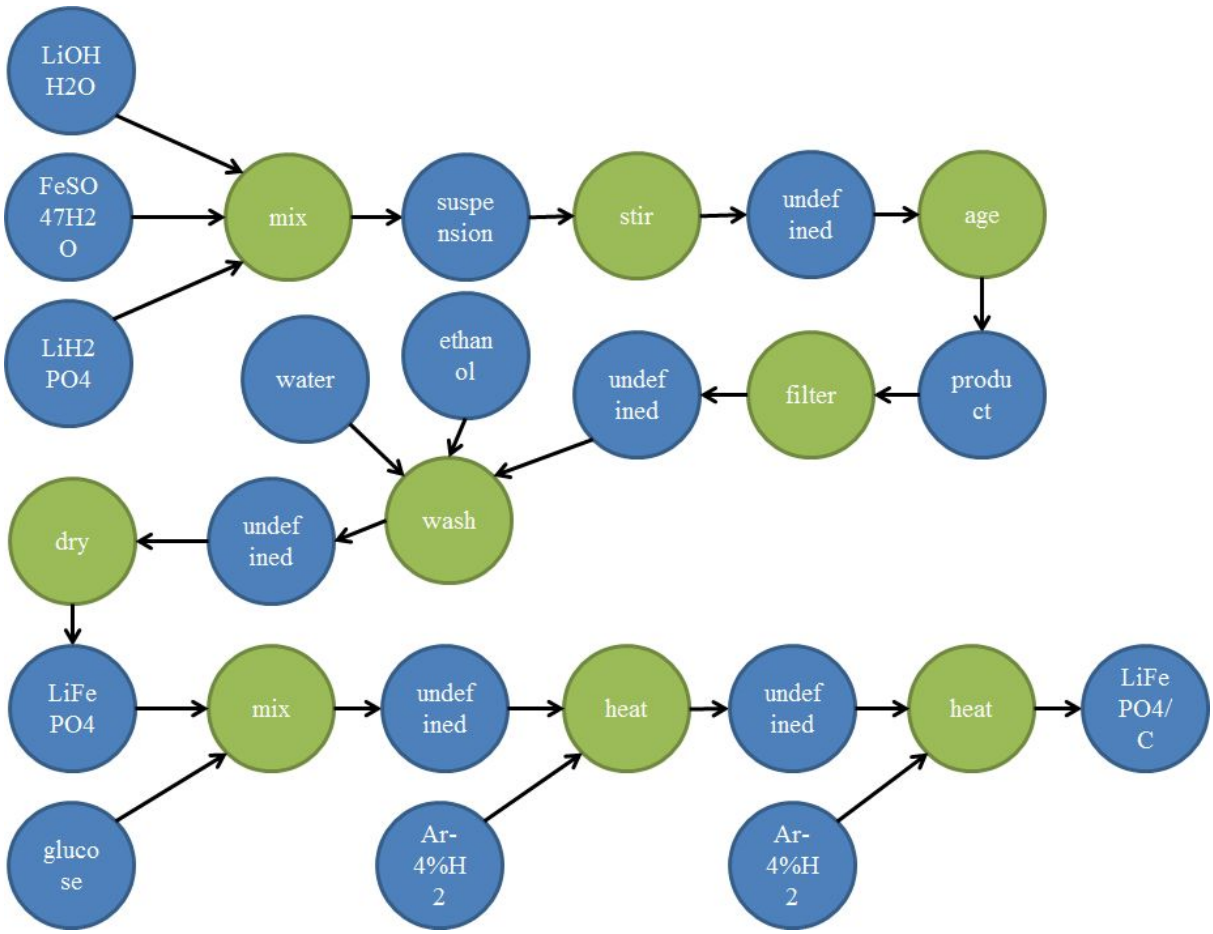

Figure 3. Ground-truth action graph.

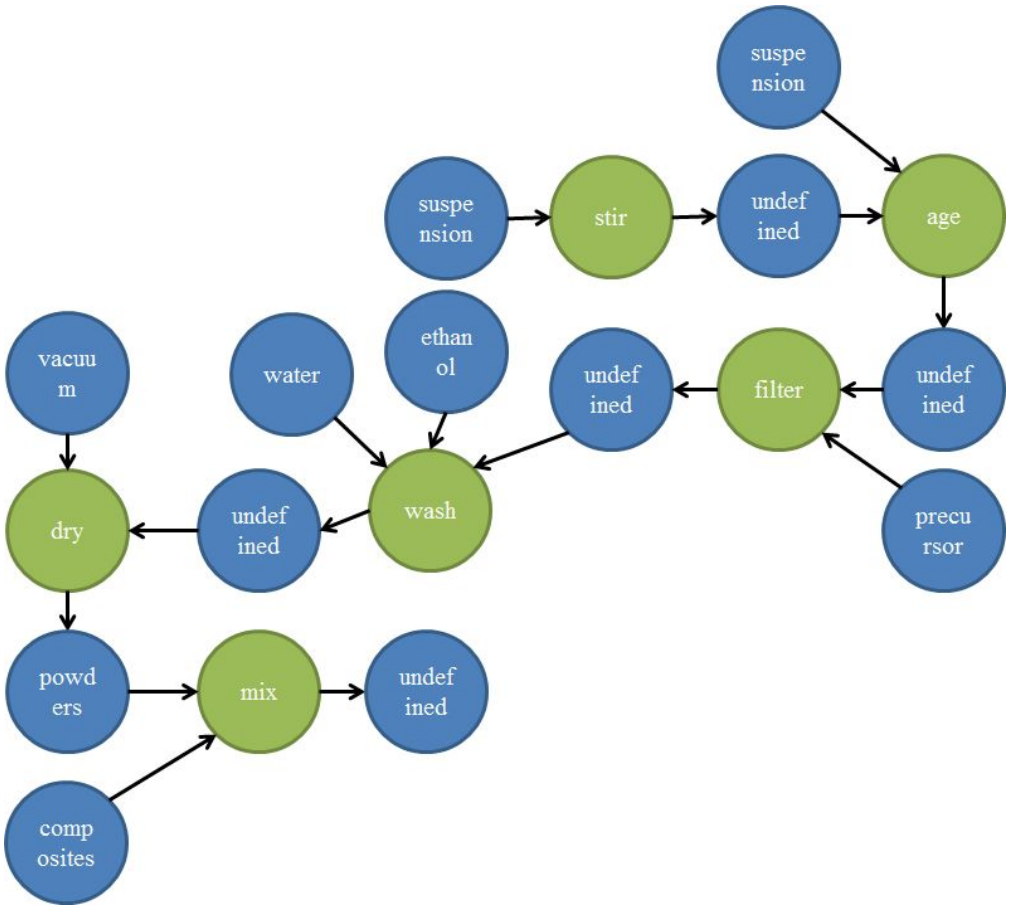

Figure 4. Our predicted action graph.

By inspecting the action graphs, one can notice that while minor differences exists, the resulting action graph still preserves the overall structure of the process. It is also able to distinguish the intermediary implicit noun nodes, labeled as 'undefined' in Figures 3 and 4. However, the part where the model suffered the most was when the verbs were implicit. In the example above, verbs such as '*heat*' and '*mix*' are not shown explicitly in the text. As a result, an entire sub-procedure is left out from the predicted action graph.

Another important improvement point is that the classifications for each word-pairs are independent. This leads to erroneous cases where the same noun arguments appear multiple times within the graph. In Figure 4, '*suspension'* appears as an input for both '*stir'* and '*age'*. We can further improve this model by introducing joint distributions over nodes such that nodes that have been referred in one context is unlikely to be referred in another.

IV: Order Refinement

Sometimes, plans are not specified in the order that they are written. Given *"solution was heated after the materials were milled and stirred,"* the order of extracted verbs: heat → mill → stir should be rearranged to mill → stir → heat due to the meaning of the preposition, 'after.' Because action graphs corresponds to a schedule, it needs to be properly ordered. In order to make such corrections, we applied an order refinement model. Our approach was to train a classifier to turn all temporal expressions to that of partial ordering moves. We extracted all forms of temporal prepositions, conjunctions, and adverbs and classified them to partial moves of **'Forward', 'Backward', 'Beginning', and 'End.'**

Table 7. A vocabulary of temporal expressions

then, starting, followed by, finally, subsequently, first, after, once after, meanwhile, subsequent, before, previously,..etc.

The 'Forward' move leaves action at its current place, signifying that current action follows after the previous action. The 'Backward' move puts action ahead of the previous-mentioned action. The 'Beginning' move relocates action to the beginning of the list. Such a case may be triggered by words like 'Starting' and 'first.' Finally, the 'End' move puts the action at the end of the list. Because temporal expressions are generally context-independent (i.e., each preposition has a defined temporal logic), we used only indicator functions and trained a decision tree with four-way classification (10-fold cross validation).

IV-A: Performance with refinement

Sequence scoring was performed by sliding windows of bigrams. More conservative measures of higher-order n-grams could have been used. There are a variety of sequence scoring methods like the Kendall rank correlation coefficient, but there assumes that gold and predicted sequences are equal in length. This was not the case for our prediction problem (i.e. the number of action nodes can differ). Table 8 highlights the performance of our refinement approach. Surprisingly, the refinement model degraded the performance of chronological baseline. Chronological baseline represents leaving the order of actions "as it is." There was a slight reduction of F1 score from 0.645 to 0.625 when using the result of our Maxent action detector. The result can be more clearly seen when we assume perfect action detection (i.e., working with gold verbs). Chronological baseline, essentially representing all 'Forward' moves performed remarkably well with F1 score of 0.951. The refinement model negatively perturbed the performance.

Table 8. Ordering performance with and without the refinement model.

With MaxEnt Action Detect			
	Precision	Recall	F1 Score
Chronological Baseline	0.607	0.688	0.645
Refinement Model	0.589	0.667	0.625
Assuming Perfect Action Detection			
	Precision	Recall	F1 Score
Chronological Baseline	0.951	0.951	0.951
Refinement Model	0.923	0.943	0.933

Table 9. Frequency of partial moves

Partial Moves	Frequency	Relative Frequency	Sample Prepositions
Forward	36	92%	then, followed by, first
Backward	2	5%	after, once after
Beginning	1	3%	meanwhile
End	0	0%	[empty]

In order to further explore the reason, we post-processed the result and analyzed relative frequency of partial moves. Table 9 confirmed how 'Backward', 'Beginning', and 'End' partial orders rarely occur. It was interesting to discover that ordering problem may not be a crucial learning problem even with more complex text like synthesis procedures. The behavior was similar to that of other instructions like cooking recipes and how-to instructions. A well-tuned decision tree may have prevented negative 'overfitting' effect of our approach.

We assumed that ordering was context-independent. Specifically assuming independence from input and output arguments. There was strong evidence favoring such an assumption due to the chronological baseline being a good heuristic. Perhaps, an ordering refinement subproblem is more interesting to human-spoken dialogue. We tend to write instructions in a structured, ordered format, but perturbations may be more frequent when we have to explain plans through speech.

V. Conclusion

We have attempted a supervised framework to construct action graphs from instructional text blocks. We explored the task on the domain of material synthesis. The MaxEnt model demonstrated improved performance compared to heuristic baselines in almost all of our tasks such as identifying the key nodes and extracting the edge relationships. However, since one has to manually annotate the action graphs, supervised method have a natural limitation for large-scale training. We have shown that supervised methods in this domain can have meaningful results, which gives the potential to explore into semi-supervised or unsupervised methods.

References

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.

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.

Chloé Kiddon, Ganesa T. Ponnuraj, Luke Zettlemoyer, and Yejin Choi.. 2015. "Mise en Place: Unsupervised Interpretation of Instructional Recipes." In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

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.

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.

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).*

Github link: https://github.mit.edu/jokim/actiongraph