Segmenting low-level instructions into high-level instructions*

Amit Goyal and Jiarong Jiang and Hal Daumé III
Dept. of Computer Science, University of Maryland, College Park, MD 20742
{amit,jiarong,hal}@umiacs.umd.edu

Written instructions on the web has become a common way to teach people how to accomplish certain tasks. For example, like a cooking recipe, instructions are given in step-by-step manner to teach how to cook a dish. Now a days, many instructions are distributed in electronic form. For example, in a company, employees are routinely sent instructions via email on how to update their emergency contact information, to setup a direct deposit in a bank, and how to apply for reimbursements for business trips. On the web, people are sharing instructions via forums and blogs to help each other accomplish tasks in various applications. In general, written instructions can be crudely divided in two categories: one is high-level instructions which are meant for experts, and other is detailed low-level instructions which are for amateurs.

In this work, our goal is to segment written instructions into high-level instruction and each high-level instruction maps out to multiple low-level instructions. This work is important, since if I am an expert cook, I only need high-level instructions to accomplish a task. However, an amateur needs detailed low level instructions for it. This work has two parts to it. First, to segment written instructions into high-level instructions. Second, to identify the name of high-level instruction. For example, in cooking recipe, given instructions on how to cook rice, we want to map these low level instructions to high-level state that is cook rice. For this work, we just focus on segment written instructions into high-level instructions and not identifying state name i.e. cook rice in the last example.

The instruction data-set's temporal ordering is linear in nature. Hence, we make use of unsupervised Hidden Markov Model (HMM) to segment the data. In order to automatically handle the length of the state sequence problem, we use the infinite Hidden Markov Model with beam sampling [1] .

1 Models

To begin, we use verb to define a step in an instruction. Hence, given a recipe, we get a chain of verbs to represent a recipe. Our goal is to segment these verbs into states which represent high-level instructions.

Synthetic Data For this work, we will show our results only on synthetic data set. The actual number of high-level instructions (states) in the data set is 12. For each fully ordered chain, they are composed of 5 to 13 low-level instructions (verbs). For each document in the training data, it has different length of state sequences. In order to gradually increase the complexity of the data set, we run our experiments on the following three versions:

1. Different high-level instruction (state) sequences do not share any common verb and for each verb sequence, if it appears in the document, the whole verb sequence will be observed.
2. Different states do not share any common verb and for each verb sequence, it might occur in the form of subsequences. For example, in the recipe script, the full ordered sequence contains "heat-remove-stir-set-cool" and in some documents, only partial of this sequence could be observed, say "heat-remove".
3. Different state sequences may share some verbs and each sequence may occur in the form of subsequences.

Hidden Markov Model First of all, we can model the segmentation problem by the unsupervised Hidden Markov Model (HMM). Each state represents an high level instruction category and given the state, it emits an low level instruction (verb) according to the high level instruction sequence category. The transition probability for a state to itself should be higher than other states as events in a sequence should prefer to co-occur in the same event sequence category.

We convert HMM to the finite state transducer and use CARMEL[1] with EM optimization to train the transition and emission matrices on the second data set. When setting the number of states to be 12 (the truth) and initialize

*Topic: sequence modeling and Preference: oral
[1] http://www.isi.edu/licensed-sw/carmel/

emission matrix randomly and the transition matrix with 0.9 probability to stay in the same state and 0.1 to go to other states. For 100 files, it can get more than 80% accuracy.

However, when the number of states is set more than the truth, the model will try to uniformly distribute the length of sequences in one document. On the other hand, if the number of states is set fewer, the emission probability distribution will be close to uniform given any state which will put the whole document into one sequence.

Infinite Hidden Markov Model In order to automatically handle the number of event sequences problem, we use the infinite Hidden Markov Model with beam sampling [1] where the number of event sequences could be arbitrary. The plate diagram of iHMM is shown in Fig. 1.

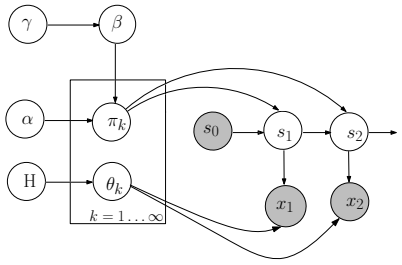

Figure 1: Plate diagram for iHMM model

The generative model is as follows:

1. for $k = 1, 2, \ldots, \infty$
   (a) Sample the stick length β with parameter γ.
   (b) Sample the DP $G_k \sim DP(\alpha, \beta)$ for transition probability from state S_k to other states
   (c) Draw the distribution over events $\phi_k \sim H$
2. for $t = 1, 2, \ldots, T$
   (a) Sample a state S_t based on the previous state $S_t | S_{t-1} \sim Multinomial(\pi_{S_{t-1}})$.
   (b) Draw an event $x_t | S_t \sim Multinomial(\phi_{S_t})$

Results using Precision, Recall, and F-measure on the three synthetic data sets are summarized below:

Synthetic Data Set	# of docs	Precision	Recall	F-measure
1	100	0.7404	0.8743	0.8018
2	1000	0.8052	0.4989	0.6161
3	1000	0.8580	0.5201	0.6476

On all three datasets, iHMM performs reasonable. We hope that these results on some real world data-sets like the recipe data-set.

Discussion In the above models, we have not taken the order of the verbs sequence into consideration. This means that it does not use the assumption that verbs in the sequences should have some specific ordering. For this ordering, we could add sequential dependency for the verbs in the iHMM. While sampling an event from a sequence, we need to sample it from the emission matrix at state S_t and an addition transition matrix from x_{t-1} to x_t given S_t. In addition, to capture verb ordering we can make use of informative prior that can encode dependence among events. For example, "heat" and "remove" should have more weight. We can use large corpus like Gigaword and use conditional probabilities among events as the initial prior.

References

[1] Jurgen Van Gael, Yunus Saatci, Yee W. Teh, and Zoubin Ghahramani. Beam sampling for the infinite hidden Markov model. In *ICML '08: Proceedings of the 25th international conference on Machine learning*, pages 1088–1095. ACM, 2008.