Unsupervised Semantic Parsing of Video Collections Supplementary Materials

Ozan Sener[1,2] Amir Zamir[1] Silvio Savarese[1] Ashutosh Saxena[1,2,3]

[1] Department of Computer Science, Stanford University
[2] Department of Computer Science, Cornell University
[3] Brain of Things Inc.

{osener,zamir,ssilvio}@stanford.edu, asaxena@cs.stanford.edu

1. Introduction

In this supplementary material, we explain the Markov Chain Monte Carlo (MCMC) sampling algorithm we use in order to learn our non-parameteric model, and we also explain the details of the natural language description generation algorithm. Moreover, we include an additional video of discovered activities and the parsing results in addition to this manuscript.

2. Learning via MCMC

In this section, we explain how we learn the non-parametric model we defined in Section 5.1 of the main paper. We follow the exact sampler proposed by Fox et al.[**?**]. It marginalize over activity likelihoods w and activity assignments $\mathbf{z}$ and samples the rest. MCMC procedure iteratively samples the conditional likelihood of activity matrix $\mathbf{F}$, activity parameters θ and transition weights η. We divide the explanation of this samplers into two sections, sampling the activities through activity matrix ($\mathbf{F}$) and activity parameters (θ), and sampling the HMM parameters η. Marginalization over activity assignments follows the efficient dynamic programming approach.

2.1. Sampling the Activities

Consider the binary activity inclusion matrix $\mathbf{F}$ such that $F_{i,j}$ is 1 if the i^{th} video has the j^{th} activity. Following the sampler of Fox et al.[**?**], we divide the sampling $\mathbf{F}$ into two parts, namely, sampling the shared activities and sampling the novel activities. Sampling shared activities correspond the re-sampling of existing entries of $\mathbf{F}$. We simply iterate over each entry and propose a flip (*i.e.* if the i^{th} video has the j^{th} activity, we propose to flip it and not to include j^{th} activity in the i^{th} video). We accept or reject this proposals following the Metropolis-Hasting rule.

In order to sample the novel activities, we follow the data-driven sampler [**?**]. Consider the case in which we want to propose a novel activity by setting the $F_{i,j+1}$ to 0. In other words, we introduce a new activity ($j+1^{th}$ activity) such that i^{th} video includes it. In order to sample the parameters θ_{j+1} of it, we first sample a temporal window W over the i^{th} video. This window is sampled by sampling the starting frame and the length of the window from a uniform distribution. Then, we sample the novel activity from *Beta distribution* as;

$$\theta_{k,n}|W \sim \text{Beta}(\alpha_n, \beta_n) \tag{1}$$

where $\theta_{k,n}$ is the n^{th} entry of θ_k, α_n is the number of frames in the window W which have the atom n, and β_n is the number of frames which do not have the atom n. We use *Beta distribution* because it is the conjugate prior of the *Bernoulli distribution* that we use to model activities.

2.2. Sampling the HMM Parameters

When the activities are defined via Θ and each video selects a subset of them via ($\mathbf{F}$), we can compute the likelihood of each state assignment by using the dynamic programming given the transition probabilities η. By using the likelihoods, we sample the state assignments $\mathbf{z}$.

When the states are sampled, we can use the closed-form sampler derived in [**?**]. Fox et al.[**?**] shows that the transition probabilities can be sampled through a *Dirichlet* random variable and scaling it with a *Gamma* random variable as;

$$\pi^{(\mathbf{i})} \sim Dir(\ldots, N_{j,k}^{(i)} + \alpha + \delta_{j,k}\kappa, \ldots) \tag{2}$$

followed by $\eta^{(\mathbf{i})} = \pi^{(\mathbf{i})} \times C^{(i)}$ such that $C^{(i)} \sim Gamma(K_+^{(i)}\lambda + \kappa, 1)$. Here, $N_{j,k}^{(i)}$ represents the number of transitions between state j and state k in the video i, α, λ and κ are hyperparameters which we learn with cross-validation, $\delta_{j,k}$ is 1 if $j=k$ and 0 o.w., and $K_+^{(i)}$ is the number of activities the i^{th} video has chosen.

3. Generation of Language Description

In this section, we explain how we generated the text description for the activity steps we discovered. We included these descriptions in Figure 8 of the main paper as well as in the supplementary videos.

In order to generate the descriptions, we simply used a Markov text generator. We collected all subtitles of all videos we included in our dataset. After combining them, we trained a 3^{rd} order Markov model by using the subtitles we downloaded. Main purpose of this training is learning the context dependent language model. Although this step can be accomplished by various of methods in the NLP literature, we choose Markov language model because of its simplicity. Indeed, this model is learned purely for visualization purposes and neither the activity step discovery nor the parsing algorithm uses this model.

After the model is learned, we need to generate a text description for each discovered activity. Since each discovered activity is represented as Bernoulli random variable, we have likelihood for each language atom. Our description generation strategy is sampling a large collection of descriptions and ranking them for their closeness to the discovered activities. We compute this closeness with the parameters of the Bernoulli random variable. Formally, given large-set of sampled descriptions $\{S_i\}_{i\in[1,K]}$, we rank them using the weights of the Bernoulli random variable as;

$$r_i = \frac{\sum_j \left[S_i^j = w^j\right] \theta^j}{\sum_j \left[S_i^j = w^j\right] 1}$$

Here, $[\cdot]$ is an indicator function, S_i^j is the j^{th} word of i^{th} description, w^j is the j^{th} word and θ^j is the j^{th} entry of the activity description. We simply choose the description having largest rank.