Learning to Describe Video with Weak Supervision by Exploiting Negative Sentential Information

Haonan Yu and **Jeffrey Mark Siskind**
Purdue University
School of Electrical and Computer Engineering
465 Northwestern Ave.
West Lafayette, IN 47907-2035 USA
haonan@haonanyu.com, qobi@purdue.edu

Abstract

Most previous work on video description trains individual parts of speech independently. It is more appealing from a linguistic point of view, for word models for all parts of speech to be learned simultaneously from whole sentences, a hypothesis suggested by some linguists for child language acquisition. In this paper, we learn to describe video by discriminatively training positive sentential labels against negative ones in a weakly supervised fashion: the meaning representations (*i.e.*, HMMs) of individual words in these labels are learned from whole sentences without any correspondence annotation of what those words denote in the video. Textual descriptions are then generated for new video using trained word models.

Introduction

Generating textual description of video is attracting increasing attention both in the natural-language-processing and computer-vision communities. Prior work mainly falls into three categories. First, some methods craft rule-based systems in which language generation is treated as an engineering task (Kojima, Tamura, and Fukunaga 2002; Lee et al. 2008; Khan, Zhang, and Gotoh 2011a; 2011b; Hanckmann, Schutte, and Burghouts 2012; Siddharth, Barbu, and Siskind 2014). With *ad hoc* rules, they manually establish the correspondence between linguistic terms and visual elements, and then analyze the events or relations among the visual elements to generate sentences. Second, other work trains statistical models for lexical entries, where the trained word models mainly serve to eliminate the tedious effort of rule design when the problem size becomes large (Das et al. 2013; Guadarrama et al. 2013; Krishnamoorthy et al. 2013; Sun and Nevatia 2014). In these approaches, the word models are pretrained individually and separately; models of different parts of speech may have different mathematical representations or training strategies. Nouns, verbs, and prepositions are then mosaiced together to yield sentences. Hybrid approaches, such as Barbu et al. (2012), learn some of the words while engineering *ad hoc* rules for the others. Third, another line of work does not train word models explicitly (Rohrbach et al. 2013). Instead, they construct a structured model (*e.g.*, Conditional Random Field, CRF) which formulates the interaction among the words in a sentence by treating words as latent labels. Sentence generation is done by inferring the latent labels given the observed variables, *i.e.*, visual features. It is unclear, however, when a sentence is generated, whether the words capture the semantic meanings of the visual concepts or are produced simply due to the high correlations encoded in the structured model.

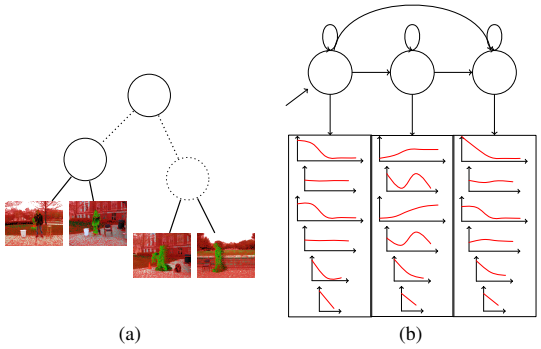

Figure 1: (a) Word classification example from Guadarrama et al. (2013): verbs are modeled by hierarchies with each leaf represented by Dense Trajectories. (b) Word grounding example: verbs are modeled by HMMs. Each state outputs distributions that have semantic meaning, such as: `move(agent,horizontally)`, `move(patient,horizontally)`, `close(agent,patient)`, and `bigger(agent,patient)`.

All the above methods that learn to describe video treat words merely as categories for the purpose of visual-feature classification. In essence, they learn to match visual features extracted from test video against those extracted from training video, to get the most likely key words from the sentences paired with the best matched training video. Then a language model is employed to generate sentences from those key words. None of them learn the actual language semantics. To truly understand video content and describe that content with natural language, a computer-vision system should aim at learning to ground words in video concepts, *i.e.*, map words to meanings. The distinction between using words for visual-feature classification and grounding

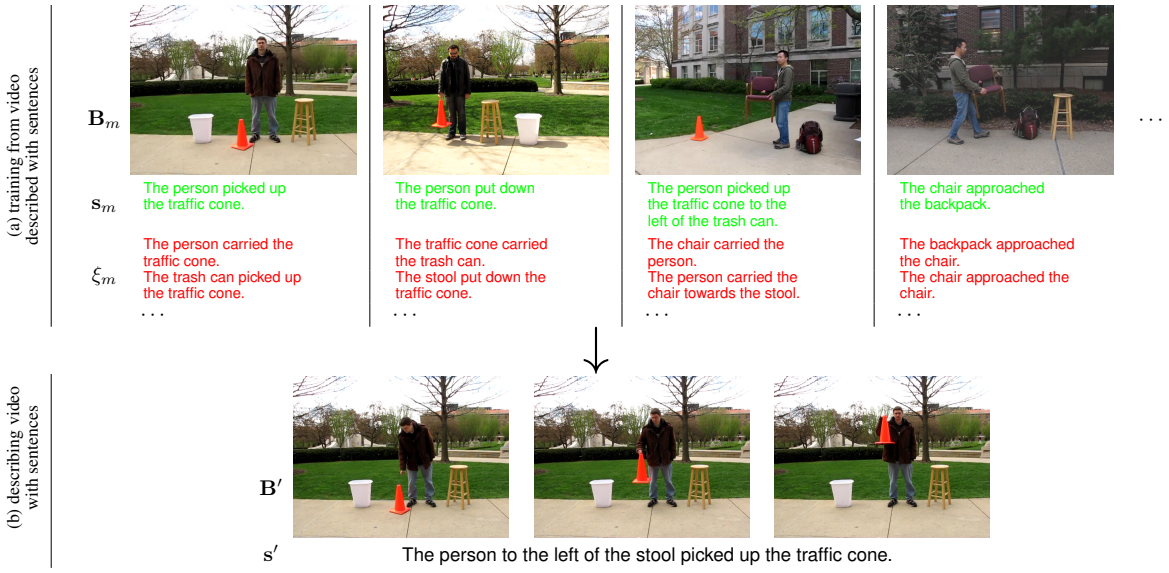

Figure 2: An overview of our problem. Sentences in green (red) are positive (negative) training labels. Our method takes $\{(\mathbf{B}_m, \mathbf{s}_m, \xi_m)\}$ as the only input, learns the meanings of the words that occur in $\mathbf{s}_m$ and ξ_m, and generates sentences $\mathbf{s}'$ for a new video clip $\mathbf{B}'$. Note that during this whole process there is no human intervention to map words to video regions/features.

words in visual concepts is illustrated in Fig. 1.

Beyond that, it is more appealing, from a linguistic point of view, for this grounding process to happen simultaneously for all parts of speech; words should be learned from scratch *simultaneously* from whole sentences. A related hypothesis is suggested by some linguists for child language acquisition (Siskind 1996): children are likely to learn word meanings under weak supervision, from what they see and hear, without conceptual annotation (*i.e.*, word-to-concept mappings). Our recent work (Yu and Siskind 2013) also provided experiment results in support of this. In this paper, we further explore this language-grounding problem by asking the question: does reasoning about semantic information that is *absent* from the video and the presented sentences help the learning of word meanings? We show that the answer is yes by demonstrating a significant improvement to sentence-generation accuracy over this prior work.

We attempt to accomplish a novel video-description task in which word models are learned solely from whole sentences that describe events that are both present and absent in the video. We train with positive and negative sentential labels (PLs and NLs) and generate textual description of new video using trained models. We show that "negative" information helps yield better trained word models. Our problem is illustrated in Fig. 2 and a roadmap of our approach is in Fig. 3. Learning in such a weakly supervised fashion requires three things. First, our method automatically learns the correspondence between words and video regions, and trains each word model with its corresponding region(s). Towards this end, the proposed method must posses the ability to implicitly annotate video data by itself for training different types of words. In our approach, this is achieved by exploiting language semantics. Second, the method extracts useful training information not only from PLs, but also from NLs. Although much prior work uses discriminative word labels for learning say, object or event models (Felzenszwalb, Girshick, and McAllester 2010; Sadanand and Corso 2012), to our knowledge, this is the first use of *discriminative sentential labels* to learn to describe video. Doing so requires major substantive mathematical machinery: we use the Growth Transformation (Gopalakrishnan et al. 1991) optimized with gradient ascent where the gradient is calculated by an adjoint graph derived similar to Back Propagation (Rumelhart, Hinton, and Williams 1986). Finally, all words from six different parts of speech that can occur in such sentential labels are represented in a unified form, and more importantly, are grounded simultaneously. This is done in a factorial Hidden Markov Model (HMM) framework (Brand, Oliver, and Pentland 1997) where each word is represented as a component HMM. In contrast, most prior work learns just object models, learns just event models, learns object and event models with differing representations, or learns such separately. Given that our task is difficult, our problem domain is currently restrictive; we focus on a small but representative lexicon instead of large quantities of video and text. Our lexicon contains 17 words from 6 parts of speech (nouns, verbs, adverbs, motion prepositions, spatial-relation prepositions, and determiners).

Here, we first give an example of our training samples. Suppose we have a training video clip (the first one shown in Fig. 2). For this particular video, the following sentences can be PLs:

1a The person picked up the traffic cone.
1b The person picked up the traffic cone to the left of the stool.

Moreover, any sentence containing at least one word that is untrue about the video is an NL, such as any of the following:[1]

[1]Note that while we use negative sentences for training pur-

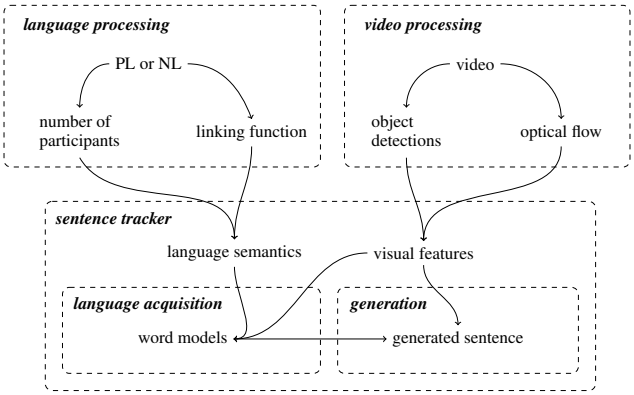

Figure 3: A roadmap of our approach. The dashed boxes represent the procedure components and the arrows represent flow of information between those components.

2a The person carried the traffic cone.
2b The person picked up the traffic cone to the right of the stool.
2c The traffic cone picked up the person.
2d The backpack approached the trash can.

Note that an NL can share most of the words with a PL; they might differ by just a single word (1a/2a and 1b/2b). An NL might even share all the words with a PL but differ in word order (1a/2c), because sentential semantics depends on that order: person should be the agent and traffic cone should be the patient. When a PL and an NL share common components, discriminative training will drive the remaining components to be different from each other. While PLs are manually annotated for a video, NLs are automatically generated by taking the complement of the PLs.

Problem

Our training samples are triples: $(\mathbf{B}, \mathbf{s}, \xi)$, where $\mathbf{B}$ represents the information extracted from a video clip, $\mathbf{s}$ represents the PL, and ξ represents a set of NLs. Note that while there is only one $\mathbf{s}$ in a training sample, there may be another $\mathbf{s}'$ in another training sample $(\mathbf{B}, \mathbf{s}', \xi')$ with the same video clip, *i.e.*, a video clip might be used multiple times. This is because one video may depict different events, each described by different sentences. Moreover, even a single event might be described by variant sentences that highlight different aspects of that event. A sequence of such triples $\{(\mathbf{B}_m, \mathbf{s}_m, \xi_m)\}_{m=1}^{M}$ comprise our training set (Fig. 2).

Language learning with only video-PL pairs $\{(\mathbf{B}_m, \mathbf{s}_m)\}_{m=1}^{M}$ was studied in Yu and Siskind (2013). To do so, they first model each word as an HMM where each state represents a certain phase of the event described by that word. For example, the meaning of a verb, such as pick up, could be represented as a two-state HMM, where the first state describes the motion of the hand of the agent towards the patient while it is at rest and the second state describes the joint motion of the hand of the agent together with the patient in the reverse direction. Similarly, the meaning of an adverb, such as quickly, could be represented as a single-state HMM describing the high velocity of an object. Likewise, meaning of a motion preposition, such as towards, could be represented as a single-state HMM describing the decreasing distance between two objects. Moreover, the meaning of a spatial-relation preposition, such as above, could be represented as a single-state HMM describing the relative position of two objects. Dynamic concepts, such as verbs, typically will be modeled as HMMs with multiple states that describe the different phases of the events described by such words. Stationary concepts, such as objects and spatial relations, can be viewed as events with only one phase. All such HMMs operate over features extracted from the video: the positions, velocities, and accelerations of individual objects, features of individual objects such as their color, shape, and size, and the (changing) relative spatial relations between pairs of objects, *i.e.*, the distance between two objects and the orientation of a vector from one object to the other.

They then define a sample score $\mathbf{L}(\mathbf{B}; \mathbf{s}, \Lambda)$, where Λ represents the word meanings to be learned, and maximize the training-set score:

$$\Lambda^* = \arg \max_{\Lambda} \prod_{m=1}^{M} \mathbf{L}(\mathbf{B}_m; \mathbf{s}_m, \Lambda) \quad (1)$$

Since sentences $\mathbf{s}_m$ are modeled as factorial HMMs, the solution can be found by maximum-likelihood estimation over M factorial HMMs, using Baum Welch (Baum et al. 1970; Baum 1972).

We build upon the sample score and define the *discrimination score* as the ratio of the sample score of a PL to the sum of the sample scores of the corresponding NLs, with each score multiplied by a sentence prior $\pi(\mathbf{s})$:

$$\mathbf{D}(\mathbf{B}; \mathbf{s}, \xi, \Lambda) = \frac{[\mathbf{L}(\mathbf{B}; \mathbf{s}, \Lambda)\pi(\mathbf{s})]^{\epsilon}}{\sum_{\mathbf{x} \in \xi} [\mathbf{L}(\mathbf{B}; \mathbf{x}, \Lambda)\pi(\mathbf{x})]^{\epsilon}} \quad (2)$$

where $0 < \epsilon \ll 1$ is a constant smoothing factor (*e.g.*, 0.01). The sentence prior $\pi(\mathbf{s})$ renders the scores comparable across variation in sentence length, as longer sentences tend to have lower score. It does so by assigning a higher prior to longer sentences. The discrimination score can also be seen as an approximation to the conditional probability of a given positive sentence: instead of summing all possible sentences in the sentence space, the partition function here is replaced by the sum of a sampled subset, which makes the problem tractable. Then, we try to maximize the total discrimination score for all of the training samples:

$$\Lambda^* = \arg \max_{\Lambda} \prod_{m=1}^{M} \mathbf{D}(\mathbf{B}_m; \mathbf{s}_m, \xi_m, \Lambda) \quad (3)$$

poses, we do not claim, from the perspective of the language-acquisition process, that children must hear hundreds of negative sentences in each scenario in order for them to learn the correct word meanings. In our method, sentences are only used to specify an internal logical representation that is used to generate the factorial HMM. We can directly generate such from negative logical representations without sentences. Such negative logical representations can be produced inferentially from the video input without any actual negative sentences. Thus we don't claim that children actually hear sentences that describe things that don't occur; all we presume is that they can infer what hasn't occurred in the video in some internal logical representation.

Again, each sentential label is modeled as a factorial HMM. The above objective function directly models the discrimination between PLs and NLs and thus produces better word models that, in turn, yield better video description than the maximum-likelihood framework (Eq. 1), as sentence generation can be viewed as a process that differentiates correct sentences from incorrect ones.

The Sentence Tracker

As the discrimination score builds on the sample score used in Yu and Siskind (2013), together with methods for computing that sample score, known as the *sentence tracker* (Siddharth, Barbu, and Siskind 2014), we first review that earlier work. A sample score $\mathbf{L}(\mathbf{B}; \mathbf{s}, \Lambda)$ scores a sentential label $\mathbf{s}$ for a video clip $\mathbf{B}$. Higher scores indicate that $\mathbf{s}$ is more likely to "correctly" depict $\mathbf{B}$, in terms of the word meanings Λ. Because each word in $\mathbf{s}$ is an HMM, and the sentence can be treated as a factorial HMM composed from the word HMMs, the score then depends on how well the visual features extracted from $\mathbf{B}$ satisfy the factorial HMM implied by $\mathbf{s}$. However, the word HMMs in $\mathbf{s}$ cannot be directly applied to visual features in $\mathbf{B}$, since we do not know (yet) which word HMMs should be applied to which subsets of visual features.

This can be mediated by a middle-layer concept, the *participants* that relate $\mathbf{s}$ with $\mathbf{B}$, in the following way. By parsing $\mathbf{s}$, the number L of participants can be determined, along with a *linking function* θ_w^i that specifies which participant fills argument i of word w, for each argument of each word in that sentence. For example, a sentence like

The person carried the chair towards the backpack. (4)

has 3 participants, X, Y, and Z, and linking functions

$$\theta^1_{\mathsf{person}} = X, \quad \theta^1_{\mathsf{carried}} = X, \quad \theta^2_{\mathsf{carried}} = Y, \quad \theta^1_{\mathsf{chair}} = Y,$$
$$\theta^1_{\mathsf{towards}} = X, \quad \theta^2_{\mathsf{towards}} = Z, \quad \theta^1_{\mathsf{backpack}} = Z$$

to construct the factorial HMM

$$\textsc{Person}(X) \times \textsc{Carried}(X, Y) \times \textsc{Chair}(Y) \times \textsc{Towards}(Y, Z) \times \textsc{Backpack}(Z)$$

The linking functions assign the argument of person and the first argument of both carried and towards to X, the argument of chair and the second argument of carried to Y, and the argument of backpack and the second argument of towards to Z. As a result, our sentence is not just a bag of words: the order of words does affect the semantics, since The backpack carried the chair is quite different from The chair carried the backpack. The linking functions are the mapping from the sentence layer to the participant layer. On the other hand, we associate each participant l with an object track $\mathbf{j}_l$ in $\mathbf{B}$. However, unlike the linking functions which are the result of a purely linguistic process applied to the sentence, this mapping from participants to tracks is determined automatically during the scoring process (Eq. 7) applied to both the sentence and the video. More specifically, we preprocess the video to yield a (noisy) collection of object detections in each frame, taking $\mathbf{B}$ to be T frames with J^t detections $\{b_j^t\}_{j=1}^{J^t}$ in frame t. From this we form track collections $\mathbf{J} = \{\mathbf{j}_l\}_{l=1}^{L}$ by selecting specific detections from an intentionally overgenerated set of detections.

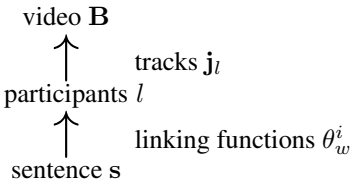

Figure 4: The three layers in the sample score.

Such track collections are not produced independently prior to learning but rather produced automatically and implicitly as part of the learning process. Thus a track $\mathbf{j}_l$, indexing a sequence of detections $\mathbf{B}_{\mathbf{j}_l} = \{b^t_{j^t_l}\}_{t=1}^{T}$ for participant l, serves as the mapping from the participant layer to the video layer (Fig. 4). The track $\mathbf{j}_l$ and the linking functions θ_w^i together "annotate" video region(s) for each word w in the sentence $\mathbf{s}$.

Now, the factorial HMM $\mathbf{s}$ can be applied to the visual features $\mathbf{B}$ through the participants l. A sentence $\mathbf{s}$ is represented by a sequence of words $\{s_w\}_{w=1}^{W}$, with each s_w referring to some entry e in the lexicon (*e.g.*, in Eq. 4, $W = 5$, $s_1 = e_{\mathsf{person}}, \cdots, s_5 = e_{\mathsf{backpack}}$). A lexical entry e is modeled as an HMM. To decide the output of a word HMM s_w, we first use the linking functions to obtain the participants θ_w^i that fill the arguments i of the word, and use the tracks $\mathbf{j}_{\theta_w^i}$ to select the detections $\mathbf{B}_{\mathbf{j}_{\theta_w^i}}$ from $\mathbf{B}$. Then the output of the HMM at frame t is computed as a feature vector $\Phi_{s_w}(\{b^t_{j^t_{\theta_w^i}}\}_i)$ of length N_{s_w}, using detections $\mathbf{B}_{\mathbf{j}_{\theta_w^i}}$ of all the arguments i of the word. Each feature $\Phi^n_{s_w}(\{b^t_{j^t_{\theta_w^i}}\}_i)$ $(1 \le n \le N_{s_w})$ is then quantized into bins. The computation function Φ_e of the feature vector depends on the lexical entry e. For example, HMMs for nouns that serve as object classifiers can output detector indices (see the Experiments section) from a single participant track while HMMs for verbs that serve as event recognizers can output distances between detections from two participant tracks.

Now, the only issue that remains is how hidden state sequences $\mathbf{k}_w = \{k_w^t\}_{t=1}^{T}$ of word HMMs s_w come into play. The sentence-tracker framework (Siddharth, Barbu, and Siskind 2014) exploits an analogy between a track $\mathbf{j}$ and a state sequence $\mathbf{k}$: detections of a track correspond to HMM states of a state sequence, the detection score corresponds to the HMM output probability, the temporal-coherence score corresponds to the HMM state-transition probability, and finding a track corresponds to estimating the state sequence. Relying on this analogy, the sample score $\mathbf{L}(\mathbf{B}; \mathbf{s}, \Lambda)$ is

$$Q(\mathbf{J}) \sum_{\mathbf{J},\mathbf{K}} \left[\prod_{l=1}^{L}\left(\prod_{t=1}^{T} f(b^t_{j^t_l})\right)\left(\prod_{t=2}^{T} g(b^{t-1}_{j^{t-1}_l}, b^t_{j^t_l})\right)\right] \left[\prod_{w=1}^{W}\left(\prod_{t=1}^{T} h_{s_w}\left(k_w^t, \Phi_{s_w}(\{b^t_{j^t_{\theta_w^i}}\}_i)\right)\right)\left(\prod_{t=2}^{T} a_{s_w}(k_w^{t-1}, k_w^t)\right)\right] \tag{7}$$

which produces a huge cross-product lattice among L participants and W word HMMs. In the above, f is the detection score and g is the temporal-coherence score. Both scores are within $[0, 1]$ and are adopted from Siddharth, Barbu, and Siskind (2014). The quantity h_{s_w} is the output distribution

$$\overline{a^e_{v,u}} = \sum_{t=1}^{T} \sum_{\substack{w=1 \\ s_w=e}}^{W} \sum_{\substack{\mathbf{j}^t,\mathbf{k}^t \\ k^t_w=u}} \left[\overline{\alpha^t_{\mathbf{j}^t,\mathbf{k}^t}} \cdot \delta^t_{\mathbf{j}^t,\mathbf{k}^t} \cdot \sum_{\substack{\mathbf{j}^{t-1},\mathbf{k}^{t-1} \\ k^{t-1}_w=v}} \alpha^{t-1}_{\mathbf{j}^{t-1},\mathbf{k}^{t-1}} \left(\prod_{\substack{w'=1 \\ w'\neq w}}^{W} a_{s'_w}(k^{t-1}_{w'}, k^t_{w'}) \right) \left(\prod_{l=1}^{L} g(b^{t-1}_{j^{t-1}_l}, b^t_{j^t_l}) \right) \right] \tag{5}$$

$$\overline{h^n_e(u,z)} = \sum_{t=1}^{T} \sum_{\substack{w=1 \\ s_w=e}}^{W} \sum_{\substack{\mathbf{j}^t,\mathbf{k}^t \\ k^t_w=u \\ \Phi^n_e(\{b^t_{j^t_{\theta^i_w}}\}_i)=z}} \left[\overline{\alpha^t_{\mathbf{j}^t,\mathbf{k}^t}} \cdot \delta^t_{\mathbf{j}^t,\mathbf{k}^t} \cdot \left(\prod_{\substack{w'=1 \\ w'\neq w \vee n'\neq n}}^{W} \prod_{n'=1}^{N_{s_{w'}}} h^{n'}_{s_{w'}} \left(k^t_{w'}, \Phi^{n'}_{s_{w'}}(\{b^t_{j^t_{\theta^i_{w'}}}\}_i) \right) \right) \left(\prod_{l=1}^{L} f(b^t_{j^t_l}) \right) \right] \tag{6}$$

of lexical entry s_w, a_{s_w} is the state-transition probability, and $Q(\mathbf{J})$ is a positive quantity independent of HMMs for normalizing track scores. (The initial probability of state k for HMM e is defined as $a_e(0,k)$.) The output distribution h and state-transition probability a together constitute the word meanings Λ. Thus a sentence $\mathbf{s}$ is essentially a large factorial HMM with each component s_w as a word HMM. The forward algorithm (Baum and Petrie 1966) computes the above score, including $Q(\mathbf{J})$, efficiently in polynomial time. Readers are referred to (Siddharth, Barbu, and Siskind 2014) for a detailed analysis of the time complexity of the sentence tracker.

Training Discriminative Sentential Labels

We find a local maximum to Eq. 3 using the Growth Transformation (GT). First, we rewrite the objective function in its logarithm form:

$$O(\Lambda) = \sum_{m=1}^{M} \left[\begin{array}{l} \epsilon \cdot \log \mathbf{L}(\mathbf{B}_m; \mathbf{s}_m, \Lambda) + \epsilon \cdot \log \pi(\mathbf{s}_m) \\ -\log \sum_{\mathbf{x}\in\xi_m} \left(\mathbf{L}(\mathbf{B}_m; \mathbf{x}, \Lambda)\pi(\mathbf{x})\right)^{\epsilon} \end{array} \right]$$

Note that $\epsilon \cdot \log \pi(\mathbf{s}_m)$ can be ignored since it's independent of Λ. The fact that the numerator and denominator in the discrimination score $\mathbf{D}(\mathbf{B}; \mathbf{s}, \xi, \Lambda)$ in Eq. 2 are both polynomials makes the following GT reestimation formula applicable to $O(\Lambda)$ as shown by Gopalakrishnan et al. (1991):

$$\lambda^{(n)}_{i,j} = \lambda^{(n-1)}_{i,j} \left(\left. \frac{\partial O(\Lambda)}{\partial \lambda_{i,j}} \right|_{\lambda_{i,j}=\lambda^{(n-1)}_{i,j}} + C^{(n)}_i \right) D^{(n)}_i \tag{8}$$

where $\lambda_{i,j}$ is one of the parameters (*i.e.*, either an output probability h or a state-transition probability a in Λ) that satisfies the sum-to-one constraint $\sum_j \lambda_{i,j} = 1$ of some distribution $\{\lambda_{i,j}\}_j$, and $D^{(n)}_i$ is for renormalization. The above update formula guarantees the growth $O(\Lambda^{(n)}) \geq O(\Lambda^{(n-1)})$ given that a sufficiently large damping factor $C^{(n)}_i$ is provided for each $\lambda^{(n)}_{i,j}$ in the distribution $\{\lambda^{(n)}_{i,j}\}_j$. Generally, the formula first takes the derivative of $O(\Lambda)$ *w.r.t.* each parameter, then adds a large positive value to the result, multiplying the sum by the old parameter, and finally renormalizes the parameters in each distribution.

While it is easy to apply GT to our objective function and get the general update formula, the main issue resides in how to compute the derivative $\frac{\partial O(\Lambda)}{\partial \lambda_{i,j}}$ for every parameter in Eq. 8. To start with, notice that only $\mathbf{L}$ in $O(\Lambda)$ is a function of Λ. Thus the problem boils down to how to compute the derivative of $\mathbf{L}$ *w.r.t.* each $\lambda_{i,j}$. For compact representation, we denote $\overline{\lambda_{i,j}} = \frac{\partial \mathbf{L}(\mathbf{B};\mathbf{s},\Lambda)}{\partial \lambda_{i,j}}$. To compute this, we need to rewrite Eq. 7 using the forward algorithm

$$\mathbf{L}(\mathbf{B}; \mathbf{s}, \Lambda) = Q(\mathbf{J}) \sum_{\mathbf{j}^T,\mathbf{k}^T} \alpha^T_{\mathbf{j}^T,\mathbf{k}^T}$$

where $\mathbf{j}^t$ is the sequence $\{j^t_l\}^L_{l=1}$ of detection indices for the L participants at frame t and $\mathbf{k}^t$ is the sequence $\{k^t_w\}^W_{w=1}$ of HMM states for the W words at frame t, respectively. Here $\alpha^t_{\mathbf{j}^t,\mathbf{k}^t}$ represents the partial score of observing detections $\{b^1_j\}^{J^1}_{j=1}, \cdots, \{b^t_j\}^{J^t}_{j=1}$ up to frame t and having detection indices $\mathbf{j}^t$ and HMM states $\mathbf{k}^t$ at frame t. This quantity is analogous to α in the general forward algorithm (Levinson, Rabiner, and Sondhi 1983) and can also be recursively computed

$$\alpha^t_{\mathbf{j}^t,\mathbf{k}^t} = \underbrace{\left[\sum_{\mathbf{j}^{t-1},\mathbf{k}^{t-1}} \alpha^{t-1}_{\mathbf{j}^{t-1},\mathbf{k}^{t-1}} \left(\prod_{w=1}^{W} a_{s_w}(k^{t-1}_w, k^t_w) \right) \left(\prod_{l=1}^{L} g(b^{t-1}_{j^{t-1}_l}, b^t_{j^t_l}) \right) \right]}_{\gamma^t_{\mathbf{j}^t,\mathbf{k}^t}} \underbrace{\left[\left(\prod_{w=1}^{W} \prod_{n=1}^{N_{s_w}} h^n_{s_w} \left(k^t_w, \Phi^n_{s_w}(\{b^t_{j^t_{\theta^i_w}}\}_i) \right) \right) \left(\prod_{l=1}^{L} f(b^t_{j^t_l}) \right) \right]}_{\delta^t_{\mathbf{j}^t,\mathbf{k}^t}} \tag{9}$$

on the huge cross-product lattice over L participants and W word HMMs. Then, before getting to $\overline{\lambda_{i,j}}$, we need to compute the derivative $\overline{\alpha^t_{\mathbf{j}^t,\mathbf{k}^t}}$ of $\mathbf{L}(\mathbf{B}; \mathbf{s}, \Lambda)$ *w.r.t.* each $\alpha^t_{\mathbf{j}^t,\mathbf{k}^t}$. When $t = T$, it is trivial to see that $\overline{\alpha^T_{\mathbf{j}^T,\mathbf{k}^T}} = Q(\mathbf{J})$. For $t < T$, using the chain rule and Eq. 9, we get

$$\overline{\alpha^t_{\mathbf{j}^t,\mathbf{k}^t}} = \sum_{\mathbf{j}^{t+1},\mathbf{k}^{t+1}} \left(\prod_{w=1}^{W} a_{s_w}(k^t_w, k^{t+1}_w) \right) \left(\prod_{l=1}^{L} g(b^t_{j^t_l}, b^{t+1}_{j^{t+1}_l}) \right) \overline{\alpha^{t+1}_{\mathbf{j}^{t+1},\mathbf{k}^{t+1}}} \delta^{t+1}_{\mathbf{j}^{t+1},\mathbf{k}^{t+1}}$$

where $\delta^t_{\mathbf{j}^t,\mathbf{k}^t}$ is the second term in Eq. 9.

Now, we are able to derive $\overline{\lambda_{i,j}}$: the partial derivatives *w.r.t.* the transition probability $a^e_{v,u}$ from state v to state u of the HMM e, and the probability $h^n_e(u,z)$ of outputting the value z as the nth feature at state u of the HMM e, are given in Eqs. 5 and 6 respectively, where $\gamma^t_{\mathbf{j}^t,\mathbf{k}^t}$ is the first term in Eq. 9. After this, the derivatives are used to obtain

Figure 5: Examples of our generated sentences.

$$
\begin{aligned}
\text{S} &\rightarrow \text{NP VP} \\
\text{NP} &\rightarrow \text{D N [PP]} \\
\text{D} &\rightarrow \text{the} \\
\text{N} &\rightarrow \text{person} \mid \text{backpack} \mid \text{chair} \mid \text{traffic cone} \mid \\
&\quad\ \text{trash can} \mid \text{stool} \\
\text{PP} &\rightarrow \text{P NP} \\
\text{P} &\rightarrow \text{to the left of} \mid \text{to the right of} \\
\text{VP} &\rightarrow \text{V NP [Adv] [PP}_\text{M}\text{]} \\
\text{V} &\rightarrow \text{approached} \mid \text{carried} \mid \text{picked up} \mid \text{put down} \\
\text{Adv} &\rightarrow \text{quickly} \mid \text{slowly} \\
\text{PP}_\text{M} &\rightarrow \text{P}_\text{M}\ \text{NP} \\
\text{P}_\text{M} &\rightarrow \text{towards} \mid \text{away from}
\end{aligned}
$$

Figure 6: The grammar used for the experiment. Our lexicon contains 17 lexical entries over 6 parts of speech (6 nouns, 4 verbs, 2 adverbs, 2 motion prepositions, 2 spatial-relation prepositions, and 1 determiner). Note that the grammar allows for infinite recursion in the noun phrase.

$\frac{\partial O(\Lambda)}{\partial \lambda_{i,j}}$ which are in turn put back into Eq. 8 for parameter reestimation. Computing the derivatives (*i.e.*, Eqs. 5 and 6) turns out to have the same time complexity with the sentence tracker reviewed in the Sentence Tracker section, *i.e.*, $O\left(T(J^L K^W)^2\right)$, where T is the video length, J is the maximal number of detections per frame, L is the number of participants, K is the maximal number of states in the HMMs, and W is the number of words in the sentence. Usually, L, W are quite small (*e.g.*, $L \leq 3$ and $W \leq 5$) and thus the algorithm runs approximately in polynomial time. In practice, pruning performed on the sparse cross-product state space can make the running time even shorter.

The damping factor C_i affects both the performance and the efficiency of the training procedure. Generally C_i should always guarantee the nonnegativity of $\lambda_{i,j}$, otherwise the objective might decrease. However, when $C_i \rightarrow \infty$, the update step size is so small that the parameters remain almost unchanged and it takes a long time to converge. Thus, we employ an adaptive method. The idea is that if the current iteration increases the objective, we always pick C_i as small as possible for the next iteration. If the objective decreases, we discard the new parameters, keep increasing C_i, and reupdate the parameters until the objective increases or C_i reaches the upper limit. More specifically, let n' be the last successfully updated iteration. The value of $C_i^{(n)}$ is determined by comparing iteration $n-1$ with iteration n':

$$
C_i^{(n)} = \begin{cases} \max\left[0, -\min\limits_j \dfrac{\partial O}{\partial \lambda_{i,j}}\Big|_{\lambda_{i,j}=\lambda_{i,j}^{(n-1)}} + \varepsilon\right] & n' = n-1 \\ \chi \cdot \max(\varepsilon, C_i^{(n-1)}) & n' < n-1 \end{cases}
$$

where $\chi > 1$ is a fixed punishment factor and $0 < \varepsilon \ll 1$. With this strategy, our method usually converges within a few dozen iterations ($20 \sim 30$).

Our iterative estimation procedure with GT is a local optimization method. The objective function $O(\Lambda)$ in Eq. 2 is nonconvex and highly complex since it is a rational function of factorial HMMs. As a result, the function surface might be jagged and a local-search procedure like ours may be easily trapped into one of the many local maxima, the majority of which are far from optimal (Jiang 2010). Thus, the smoothing factor ϵ in Eq. 2 is crucial for a good solution; if it is sufficiently small ($0 < \epsilon \ll 1$), it removes most shallow maxima areas and flattens the function surface. In the experiment, we set ϵ to be the reciprocal of the video length, *i.e.*, $1/T$. Conceptually, this renders Eq. 2 as a discrimination between scores on a per-frame basis, enabling the comparison between two videos that have different lengths.

Experiment

To evaluate our approach, we filmed a corpus of 94 video clips. The corpus was filmed at 640×480 resolution at 30 fps. Each clip varies in length between 2 and 5 seconds. The clips were filmed in four outdoor environments. Each clip contains a single person from a collection of four actors, as well as either two or three objects out of a collection of five objects: a backpack, a chair, a traffic cone, a trash can, and a stool. Every video clip depicts multiple simultaneous events and even a single event can be described with sentences that refer to different aspects of that event (Fig. 2).

To obtain detections $\mathbf{B}$ (the Sentence Tracker section), an off-the-shelf object detector (Felzenszwalb, Girshick, and McAllester 2010; Felzenszwalb et al. 2010) is run on each frame to obtain rectangular detections around objects. Detections from each object class have a unique *detector index* (*e.g.*, number or symbol) that differentiates them from detections from other classes. We trained six object detectors, one for each of the six object classes in our corpus: person, backpack, chair, traffic cone, trash can, and stool. Note

	person	backpack	chair	traffic cone	trash can	stool
person	93	0	3	1	0	3
backpack	0	91	1	1	7	0
chair	0	12	81	5	0	2
traffic cone	0	11	5	82	0	2
trash can	0	20	5	3	71	1
stool	0	0	1	8	0	91

Figure 7: Examples of the learned output probabilities (in percentages) for one-state noun HMMs. Each row represents a lexical entry. Each column represents a detector index.

that detector index of an object class has no specified correspondence with any noun in the lexicon; instead, the correspondence is learned. We pick the 2 highest-scoring detections of each object class and pool them to get $6 \times 2 = 12$ detections per frame. Then features in Yu and Siskind (2013) are computed for the detections: *detector index* and *velocity magnitude/orientation* of one participant, and *distance*, *size ratio*, and *x-position* between two participants.

We annotate and obtain a total of 276 PLs for all the video clips, with 2.94 PLs per clip on average. Then N candidate sentences are generated from the grammar (Fig. 6) for each video-PL pair. To best exploit the power of the discriminative-training framework, we define priority on the generated candidate sentences. We wish to select NLs that are maximally effective; NLs that differ greatly from a given PL are the least *confusable* with that PL and offer the least discriminative power. On the other hand, those that are highly similar to the PL are the most confusable and are the most effective for learning. Thus we first consider M *near misses*. A near miss is generated by replacing a single word in the PL (*e.g.*, 2a *w.r.t.* 1a and 2b *w.r.t.* 1b in the Introduction). Near misses are effective because they tell the training procedure which single word of the PL is to be trained against that of the corresponding NL.[2] They drive the training procedure to differentiate pairs of words of the same part of speech. After this, we consider $N - M$ random samples in the sentence space determined by our grammar. Finally, the candidates that are true of the video are filtered out and the remaining ones are kept as NLs, where the filtering process operates by deciding whether a given candidate is in the true sentence set annotated by the human. In the experiment, we empirically set $N = 120$, resulting in 33,396 candidate sentential labels in total.

In the test, given a video clip $\mathbf{B}$, the sentence generator outputs the sentence $\mathbf{s}^*$, and its correctness is then decided. The sentence generator is adopted from Siddharth, Barbu, and Siskind (2014), in which beam search is used to approximate the search for the highest-scoring sentence:

$$\mathbf{s}^* = \arg \max_{\mathbf{s}} \mathbf{L}(\mathbf{B}, \mathbf{s}, \Lambda)$$

[2]In this case, we are actually training words against words, like most prior discriminative-training methods. However, unlike these prior methods, our framework does not need the annotated video regions for the individual words that are trained, as this is determined implicitly.

Rand	**Blind**	**ML**	**Our**	**Hand** (ground truth)
0.00	0.06	0.26	0.45	0.63

Table 1: Accuracies of different methods.

Some examples of generated sentences are given in Fig. 5.

To evaluate our method quantitatively, we perform three-fold cross validation. For each fold, about 20 video clips are selected for testing and the remaining ones are used for training. We compute the generation accuracy, *i.e.*, the number of hits divided by the number of total test cases. The final accuracy is computed as the average of the accuracies over three folds. For comparison, we report accuracies of four other baselines: **Rand**, **Blind**, **ML**, and **Hand**.[3] The **Rand** baseline uses randomized HMMs for generation. The **Blind** baseline outputs the same sentence for all test video clips. If this baseline works well, it means that our corpus is highly biased and poorly designed, in the sense that most clips depict the same sentence. We find the upper bound for the accuracy of this baseline by trying all AVP (agent-verb-patient) triplets in the grammar, and take the one that correctly describes the most clips as the blind decision. Any other sentence from the grammar cannot possibly perform better than this one because adding words is only likely to change a correct sentence to an incorrect one, but not vice versa. The **ML** baseline (Yu and Siskind 2013) trains the word models using only the 276 video-PL pairs. Finally, the **Hand** baseline generates sentences using hand-crafted HMMs, similar to the prior work (Siddharth, Barbu, and Siskind 2014) in which hand-crafted FSMs are used for sentence generation. We developed the **Hand** models manually with validation on the test set,[4] as an estimate of the upper bound on our learning performance. The accuracies of different methods are shown in Table 1. As can be seen from the table, our corpus is not biased (**Rand** and **Blind**). Randomly guessing a sentence for each test video clip and blindly choosing a same sentence for all test video clips both result in nearly zero accuracy. This excludes the possibility of an algorithm achieving high accuracy by relying on the biased distribution of the sentences depicted by the test video. Our approach outperforms **ML** significantly (73.1% improvement) and is much closer to **Hand**.

To analyze the failure modes of our method, we define and compute the *minimal Hamming distance* of a generated sentence. Namely, for each generated sentence, we find an equal-length sentence from the grammar that correctly describes the paired video while yielding the minimal Hamming distance, *i.e.*, the minimal number of words that must be replaced in the generated sentence to render it a correct description of the video. Then, we obtain an averaged min-

[3]Although it is desirable that our approach also be compared with the latest work on video description reviewed in the Introduction, to the best of our knowledge none of those methods have end-to-end, open-access code or software.

[4]From a machine-learning perspective, this would constitute "training on the test set". A machine-learning method like ours that does not have access to the test set will necessarily perform worse.

	noun	verb	preposition	adverb	sentence
ML	1.25	0.58	0.20	0.00	2.03
Our	0.69	0.37	0.00	0.00	1.06

Table 2: Average minimal Hamming distances. We list the average number of words that need to be replaced for each part of speech and add them up for the whole sentence.

imal Hamming distance over all the test samples. Instead of a binary judgment made in the above, this will give a more detailed and quantitative evaluation of how far away a generated sentence is from a correct one. We compute this metric for both our method and the **ML** baseline, shown in Table 2. One can see that, on average, one needs to replace roughly one word to make the sentences generated by our method correct, while it takes two for the **ML** baseline.

Finally, to take a closer look at the underlying representations learned by our method, we show the confusion matrix of the learned output probabilities for one-state noun HMMs in Fig. 7: all noun models successfully converge to the correct modes starting from randomly initialized distributions, without any initially specified noun-to-detector correspondence. Once this mapping is correctly learned, it is clear that other parts of speech will also be properly trained, as in supervised training with manual annotation.

Conclusion

We provide a framework for learning to describe video in which discriminative training is used for learning word meanings from video paired with positive and negative sentential labels. The training procedure requires only weak supervision. Because the proposed approach utilizes negative sentential labels, better performance is obtained on the video-description task, compared to training with only positive sentential labels by maximum likelihood. Promising results have been shown on sentence generation for new video with trained word models.

Acknowledgments

This research was sponsored, in part, by the Army Research Laboratory and was accomplished under Cooperative Agreement Number W911NF-10-2-0060. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either express or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes, notwithstanding any copyright notation herein.

References

Barbu, A.; Bridge, A.; Burchill, Z.; Coroian, D.; Dickinson, S.; Fidler, S.; Michaux, A.; Mussman, S.; Siddharth, N.; Salvi, D.; Schmidt, L.; Shangguan, J.; Siskind, J. M.; Waggoner, J.; Wang, S.; Wei, J.; Yin, Y.; and Zhang, Z. 2012. Video in sentences out. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, 102–112.

Baum, L. E., and Petrie, T. 1966. Statistical inference for probabilistic functions of finite state Markov chains. *The Annals of Mathematical Statistics* 37(6):1554–1563.

Baum, L. E.; Petrie, T.; Soules, G.; and Weiss, N. 1970. A maximization technique occuring in the statistical analysis of probabilistic functions of Markov chains. *The Annals of Mathematical Statistics* 41(1):164–171.

Baum, L. E. 1972. An inequality and associated maximization technique in statistical estimation of probabilistic functions of a Markov process. *Inequalities* 3:1–8.

Brand, M.; Oliver, N.; and Pentland, A. 1997. Coupled hidden Markov models for complex action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 994–999.

Das, P.; Xu, C.; Doell, R. F.; and Corso, J. J. 2013. A thousand frames in just a few words: Lingual description of videos through latent topics and sparse object stitching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2634–2641.

Felzenszwalb, P. F.; Girshick, R. B.; McAllester, D.; and Ramanan, D. 2010. Object detection with discriminatively trained part-based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 32(9):1627–1645.

Felzenszwalb, P. F.; Girshick, R. B.; and McAllester, D. A. 2010. Cascade object detection with deformable part models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2241–2248.

Gopalakrishnan, P. S.; Kanevsky, D.; Nadas, A.; and Nahamoo, D. 1991. An inequality for rational functions with applications to some statistical estimation problems. *IEEE Transactions on Information Theory* 37(1):107–113.

Guadarrama, S.; Krishnamoorthy, N.; Malkarnenkar, G.; Mooney, R.; Darrell, T.; and Saenko, K. 2013. Youtube2text: Recognizing and describing arbitrary activities using semantic hierarchies and zero-shot recognition. In *Proceedings of the IEEE International Conference on Computer Vision*, 2712–2719.

Hanckmann, P.; Schutte, K.; and Burghouts, G. J. 2012. Automated textual descriptions for a wide range of video events with 48 human actions. In *Proceedings of the European Conference on Computer Vision Workshops and Demonstrations*, 372–380.

Jiang, H. 2010. Discriminative training of HMMs for automatic speech recognition: A survey. *Computer Speech and Language* 24(4):589–608.

Khan, M. U. G.; Zhang, L.; and Gotoh, Y. 2011a. Human focused video description. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 1480–1487.

Khan, M. U. G.; Zhang, L.; and Gotoh, Y. 2011b. Towards coherent natural language description of video streams. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 664–671.

Kojima, A.; Tamura, T.; and Fukunaga, K. 2002. Natural language description of human activities from video images

based on concept hierarchy of actions. *International Journal of Computer Vision* 50(2):171–184.

Krishnamoorthy, N.; Malkarnenkar, G.; Mooney, R. J.; Saenko, K.; and Guadarrama, S. 2013. Generating natural-language video descriptions using text-mined knowledge. In *Proceedings of the Conference on Artificial Intelligence*, 541–547.

Lee, M. W.; Hakeem, A.; Haering, N.; and Zhu, S.-C. 2008. SAVE: A framework for semantic annotation of visual events. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 1–8.

Levinson, S. E.; Rabiner, L. R.; and Sondhi, M. M. 1983. An introduction to the application of the theory of probabilistic functions of a Markov process to automatic speech recognition. *Bell System Technical Journal* 62(4):1035–1074.

Rohrbach, M.; Qiu, W.; Titov, I.; Thater, S.; Pinkal, M.; and Schiele, B. 2013. Translating video content to natural language descriptions. In *Proceedings of the IEEE International Conference on Computer Vision*, 433–440.

Rumelhart, D. E.; Hinton, G. E.; and Williams, R. J. 1986. Learning representations by back-propagating errors. *Nature* 323:533–536.

Sadanand, S., and Corso, J. J. 2012. Action bank: A high-level representation of activity in video. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1234–1241.

Siddharth, N.; Barbu, A.; and Siskind, J. M. 2014. Seeing what you're told: Sentence-guided activity recognition in video. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Siskind, J. M. 1996. A computational study of cross-situational techniques for learning word-to-meaning mappings. *Cognition* 61(1-2):39–91.

Sun, C., and Nevatia, R. 2014. Semantic aware video transcription using random forest classifiers. In *Proceedings of the European Conference on Computer Vision*, 772–786.

Yu, H., and Siskind, J. M. 2013. Grounded language learning from video described with sentences. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, 53–63.