Grounded Language Learning from Video Described with Sentences

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

We present a method that learns representations for word meanings from short video clips paired with sentences. Unlike prior work on learning language from symbolic input, our input consists of video of people interacting with multiple complex objects in outdoor environments. Unlike prior computer-vision approaches that learn from videos with verb labels or images with noun labels, our labels are sentences containing nouns, verbs, prepositions, adjectives, and adverbs. The correspondence between words and concepts in the video is learned in an unsupervised fashion, even when the video depicts simultaneous events described by multiple sentences or when different aspects of a single event are described with multiple sentences. The learned word meanings can be subsequently used to automatically generate description of new video.

1 Introduction

People learn language through exposure to a rich perceptual context. Language is grounded by mapping words, phrases, and sentences to meaning representations referring to the world. Siskind (1996) has shown that even with referential uncertainty and noise, a system based on *cross-situational* learning can robustly acquire a lexicon, mapping words to word-level meanings from sentences paired with sentence-level meanings. However, it did so only for symbolic representations of word- and sentence-level meanings that were not perceptually grounded. An ideal system would not require detailed word-level labelings to acquire word meanings from video but rather could learn language in a largely unsupervised fashion, just as a child does, from video paired with sentences.

There has been recent research on grounded language learning. Roy (2002) pairs training sentences with vectors of real-valued features extracted from synthesized images which depict 2D blocks-world scenes, to learn a specific set of features for adjectives, nouns, and adjuncts. Yu and Ballard (2004) paired training images containing multiple objects with spoken name candidates for the objects to find the correspondence between lexical items and visual features. Dominey and Boucher (2005) paired narrated sentences with symbolic representations of their meanings, automatically extracted from video, to learn object names, spatial-relation terms, and event names as a mapping from the grammatical structure of a sentence to the semantic structure of the associated meaning representation. Chen and Mooney (2008) learned the language of sportscasting by determining the mapping between game commentaries and the meaning representations output by a rule-based simulation of the game. Kwiatkowski et al. (2012) present an approach that learns Montague-grammar representations of word meanings together with a combinatory categorial grammar (CCG) from child-directed sentences paired with first-order formulas that represent their meaning.

Although most of these methods succeed in learning word meanings from sentential descriptions they do so only for symbolic or simple visual input (often synthesized); they fail to bridge the gap between language and computer vision, *i.e.*, they do not attempt to extract meaning representations from complex visual scenes. On the other hand, there has been research on training object and event models from large corpora of complex images and video in the computer-vision community (Kuznetsova et al., 2012; Sadanand and Corso, 2012; Kulkarni et al., 2011; Ordonez et al., 2011; Yao et al., 2010). However, most such work requires training data that labels individual concepts with individual words (*i.e.*, ob-

Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics, pages 53–63,
Sofia, Bulgaria, August 4-9 2013.

jects delineated via bounding boxes in images as nouns and events that occur in short video clips as verbs). There is no attempt to model phrasal or sentential meaning, let alone acquire the object or event models from training data labeled with phrasal or sentential annotations. Moreover, such work uses distinct representations for different parts of speech; *i.e.*, object and event recognizers use different representations.

In this paper, we present a method that learns representations for word meanings from short video clips paired with sentences. Our work differs from prior work in three ways. First, our input consists of realistic video filmed in an outdoor environment. Second, we learn the entire lexicon, including nouns, verbs, prepositions, adjectives, and adverbs, simultaneously from video described with whole sentences. Third we adopt a uniform representation for the meanings of words in all parts of speech, namely Hidden Markov Models (HMMs) whose states and distributions allow for multiple possible interpretations of a word or a sentence in an ambiguous perceptual context.

We employ the following representation to ground the meanings of words, phrases, and sentences in video clips. We first run an object detector on each video frame to yield a set of *detections*, each a subregion of the frame. In principle, the object detector need just detect the objects rather than classify them. In practice, we employ a collection of class-, shape-, pose-, and viewpoint-specific detectors and pool the detections to account for objects whose shape, pose, and viewpoint may vary over time. Our methods can learn to associate a single noun with detections produced by multiple detectors. We then string together detections from individual frames to yield *tracks* for objects that temporally span the video clip. We associate a feature vector with each frame (detection) of each such track. This feature vector can encode image features (including the identity of the particular detector that produced that detection) that correlate with object class; region color, shape, and size features that correlate with object properties; and motion features, such as linear and angular object position, velocity, and acceleration, that correlate with event properties. We also compute features between pairs of tracks to encode the relative position and motion of the pairs of objects that participate in events that involve two participants. In principle, we can also compute features between tuples of any number of tracks.

Following Yamoto et al. (1992), Siskind and Morris (1996), and Starner et al. (1998), we represent the meaning of an intransitive verb, like *jump*, as a two-state HMM over the velocity-direction feature, modeling the requirement that the participant move upward then downward. We represent the meaning of a transitive verb, like *pick up*, as a two-state HMM over both single-object and object-pair features: the agent moving toward the patient while the patient is as rest, followed by the agent moving together with the patient. We extend this general approach to other parts of speech. Nouns, like *person*, can be represented as one-state HMMs over image features that correlate with the object classes denoted by those nouns. Adjectives, like *red*, *round*, and *big*, can be represented as one-state HMMs over region color, shape, and size features that correlate with object properties denoted by such adjectives. Adverbs, like *quickly*, can be represented as one-state HMMs over object-velocity features. Intransitive prepositions, like *leftward*, can be represented as one-state HMMs over velocity-direction features. Static transitive prepositions, like *to the left of*, can be represented as one-state HMMs over the relative position of a pair of objects. Dynamic transitive prepositions, like *towards*, can be represented as HMMs over the changing distance between a pair of objects. Note that with this formulation, the representation of a verb, like *approach*, might be the same as a dynamic transitive preposition, like *towards*. While it might seem like overkill to represent the meanings of words as one-state-HMMs, in practice, we often instead encode such concepts with multiple states to allow for temporal variation in the associated features due to changing pose and viewpoint as well as deal with noise and occlusion. Moreover, the general framework of modeling word meanings as temporally variant time series via multi-state HMMs allows one to model denominalized verbs, *i.e.*, nouns that denote events, as in *The jump was fast*.

Our HMMs are parameterized with varying arity. Some, like *jump*(α), *person*(α), *red*(α), *round*(α), *big*(α), *quickly*(α), and *leftward*(α) have one argument, while others, like *pick-up*(α, β), *to-the-left-of*(α, β), and *towards*(α, β), have two arguments (In principle, any arity can be supported.). HMMs are instantiated by mapping their arguments to tracks. This

involves computing the associated feature vector for that HMM over the detections in the tracks chosen to fill its arguments. This is done with a two-step process to support compositional semantics. The meaning of a multi-word phrase or sentence is represented as a joint likelihood of the HMMs for the words in that phrase or sentence. Compositionality is handled by *linking* or *coindexing* the arguments of the conjoined HMMs. Thus a sentence like *The person to the left of the backpack approached the trash-can* would be represented as a conjunction of *person*(p_0), *to-the-left-of*(p_0, p_1), *backback*(p_1), *approached*(p_0, p_2), and *trash-can*(p_2) over the three participants p_0, p_1, and p_2. This whole sentence is then grounded in a particular video by mapping these participants to particular tracks and instantiating the associated HMMs over those tracks, by computing the feature vectors for each HMM from the tracks chosen to fill its arguments.

Our algorithm makes six assumptions. First, we assume that we know the part of speech C_m associated with each lexical entry m, along with the part-of-speech dependent number of states I_c in the HMMs used to represent word meanings in that part of speech, the part-of-speech dependent number of features N_c in the feature vectors used by HMMs to represent word meanings in that part of speech, and the part-of-speech dependent feature-vector computation Φ_c used to compute the features used by HMMs to represent word meanings in that part of speech. Second, we pair individual sentences each with a short video clip that depicts that sentence. The algorithm is not able to determine the alignment between multiple sentences and longer video segments. Note that there is no requirement that the video depict *only* that sentence. Other objects may be present and other events may occur. In fact, nothing precludes a training corpus with multiple copies of the same video, each paired with a different sentence describing a different aspect of that video. Moreover, our algorithm potentially can handle a small amount of noise, where a video clip is paired with an incorrect sentence that the video does not depict. Third, we assume that we already have (pre-trained) low-level object detectors capable of detecting instances of our target event participants in individual frames of the video. We allow such detections to be unreliable; our method can handle a moderate amount of false positives and false negatives. We do not need to know the mapping from these object-detection classes to words; our algorithm determines that. Fourth, we assume that we know the arity of each word in the corpus, *i.e.*, the number of arguments that that word takes. For example, we assume that we know that the word *person*(α) takes one argument and the word *approached*(α, β) takes two arguments. Fifth, we assume that we know the total number of distinct participants that collectively fill all of the arguments for all of the words in each training sentence. For example, for the sentence *The person to the left of the backpack approached the trash-can*, we assume that we know that there are three distinct objects that participate in the event denoted. Sixth, we assume that we know the *argument-to-participant mapping* for each training sentence. Thus, for example, for the above sentence we would know *person*(p_0), *to-the-left-of*(p_0, p_1), *backback*(p_1), *approached*(p_0, p_2), and *trash-can*(p_2). The latter two items can be determined by parsing the sentence, which is what we do. One can imagine learning the ability to automatically perform the latter two items, and even the fourth item above, by learning the grammar and the part of speech of each word, such as done by Kwiatkowski et al. (2012). We leave such for future work.

Figure 1 illustrates a single frame from a potential training sample provided as input to our learner. It consists of a video clip paired with a sentence, where the arguments of the words in the sentence are mapped to participants. From a sequence of such training samples, our learner determines the objects tracks and the mapping from participants to those tracks, together with the meanings of the words.

The remainder of the paper is organized as follows. Section 2 generally describes our problem of lexical acquisition from video. Section 3 introduces our work on the sentence tracker, a method for jointly tracking the motion of multiple objects in a video that participate in a sententially-specified event. Section 4 elaborates on the details of our problem formulation in the context of this sentence tracker. Section 5 describes how to generalize and extend the sentence tracker so that it can be used to support lexical acquisition. We demonstrate this lexical acquisition algorithm on a small example in Section 6. Finally, we conclude with a discussion in Section 7.

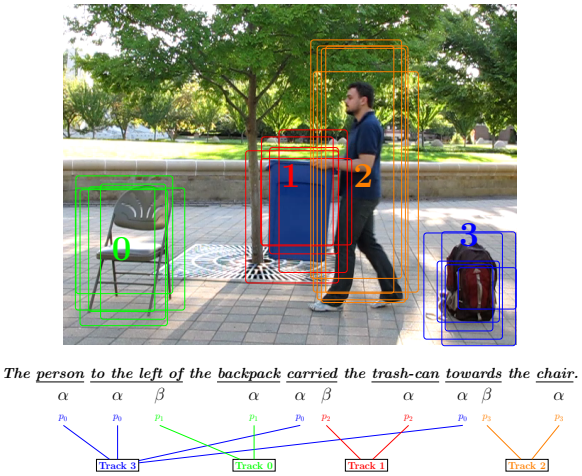

Figure 1: An illustration of our problem. Each word in the sentence has one or more arguments (α and possibly β), each argument of each word is assigned to a participant ($p_0, \ldots, p_3$) in the event described by the sentence, and each participant can be assigned to any object track in the video. This figure shows a possible (but erroneous) interpretation of the sentence where the mapping is: $p_0 \mapsto$ **Track 3**, $p_1 \mapsto$ **Track 0**, $p_2 \mapsto$ **Track 1**, and $p_3 \mapsto$ **Track 2**, which might (incorrectly) lead the learner to conclude that the word *person* maps to the backpack, the word *backpack* maps to the chair, the word *trash-can* maps to the trash-can, and the word *chair* maps to the person.

2 General Problem Formulation

Throughout this paper, lowercase letters are used for variables or hidden quantities while uppercase ones are used for constants or observed quantities.

We are given a lexicon $\{1, \ldots, M\}$, letting m denote a lexical entry. We are given a sequence $D = (D_1, \ldots, D_R)$ of video clips D_r, each paired with a sentence S_r from a sequence $S = (S_1, \ldots, S_R)$ of sentences. We refer to D_r paired with S_r as a *training sample*. Each sentence S_r is a sequence $(S_{r,1}, \ldots, S_{r,L_r})$ of words $S_{r,l}$, each an entry from the lexicon. A given entry may potentially appear in multiple sentences and even multiple times in a given sentence. For example, the third word in the first sentence might be the same entry as the second word in the fourth sentence, in which case $S_{1,3} = S_{4,2}$. This is what allows cross-situational learning in our algorithm.

Let us assume, for a moment, that we can process each video clip D_r to yield a sequence $(\tau_{r,1}, \ldots, \tau_{r,U_r})$ of object tracks $\tau_{r,u}$. Let us also assume that D_r is paired with a sentence $S_r =$ *The person approached the chair*, specified to have two participants, $p_{r,0}$ and $p_{r,1}$, with the mapping *person*$(p_{r,0})$, *chair*$(p_{r,1})$, and *approached*$(p_{r,0}, p_{r,1})$. Let us further assume, for a moment, that we are given a mapping from participants to object tracks, say $p_{r,0} \mapsto \tau_{r,39}$ and $p_{r,1} \mapsto \tau_{r,51}$. This would allow us to instantiate the HMMs with object tracks for a given video clip: *person*$(\tau_{r,39})$, *chair*$(\tau_{r,51})$, and *approached*$(\tau_{r,39}, \tau_{r,51})$. Let us further assume that we can score each such instantiated HMM and aggregate the scores for all of the words in a sentence to yield a sentence score and further aggregate the scores for all of the sentences in the corpus to yield a corpus score. However, we don't know the parameters of the HMMs. These constitute the unknown meanings of the words in our corpus which we wish to learn. The problem is to simultaneously determine (a) those parameters along with (b) the object tracks and (c) the mapping from participants to object tracks. We do this by finding (a)–(c) that maximizes the corpus score.

3 The Sentence Tracker

Barbu et al. (2012a) presented a method that first determines object tracks from a single video clip and then uses these fixed tracks with HMMs to recognize actions corresponding to verbs and construct sentential descriptions with templates. Later Barbu et al. (2012b) addressed the problem of solving (b) and (c), for a single object track constrained by a single intransitive verb, without solving (a), in the context of a single video clip. Our group has generalized this work to yield an algorithm called the *sentence tracker* which operates by way of a factorial HMM framework. We introduce that here as the foundation of our extension.

Each video clip D_r contains T_r frames. We run an object detector on each frame to yield a set D_r^t of detections. Since our object detector is unreliable, we bias it to have high recall but low precision, yielding multiple detections in each frame. We form an object track by selecting a single detection for that track for each frame. For a moment, let us consider a single video clip with length T, with detections D^t in frame t. Further, let us assume that we seek a single object track in that video clip. Let j^t denote the index of the detection from D^t in frame t that is selected to form the track. The object detector scores each detection. Let $F(D^t, j^t)$ denote that score. More-

over, we wish the track to be temporally coherent; we want the objects in a track to move smoothly over time and not jump around the field of view. Let $G(D^{t-1}, j^{t-1}, D^t, j^t)$ denote some measure of coherence between two detections in adjacent frames. (One possible such measure is consistency of the displacement of D^t relative to D^{t-1} with the velocity of D^{t-1} computed from the image by optical flow.) One can select the detections to yield a track that maximizes both the aggregate detection score and the aggregate temporal coherence score.

$$\max_{j^1,\ldots,j^T} \left(\begin{array}{l} \sum_{t=1}^{T} F(D^t, j^t) \\ + \sum_{t=2}^{T} G(D^{t-1}, j^{t-1}, D^t, j^t) \end{array} \right) \quad (1)$$

This can be determined with the Viterbi (1967) algorithm and is known as *detection-based tracking* (Viterbi, 1971).

Recall that we model the meaning of an intransitive verb as an HMM over a time series of features extracted for its participant in each frame. Let λ denote the parameters of this HMM, $(q^1, \ldots, q^T)$ denote the sequence of states q^t that leads to an observed track, $B(D^t, j^t, q^t, \lambda)$ denote the conditional log probability of observing the feature vector associated with the detection selected by j^t among the detections D^t in frame t, given that the HMM is in state q^t, and $A(q^{t-1}, q^t, \lambda)$ denote the log transition probability of the HMM. For a given track $(j^1, \ldots, j^T)$, the state sequence that yields the maximal likelihood is given by:

$$\max_{q^1,\ldots,q^T} \left(\begin{array}{l} \sum_{t=1}^{T} B(D^t, j^t, q^t, \lambda) \\ + \sum_{t=2}^{T} A(q^{t-1}, q^t, \lambda) \end{array} \right) \quad (2)$$

which can also be found by the Viterbi algorithm.

A given video clip may depict multiple objects, each moving along its own trajectory. There may be both a person jumping and a ball rolling. How are we to select one track over the other? The key insight of the sentence tracker is to bias the selection of a track so that it matches an HMM. This is done by combining the cost function of Eq. 1 with the cost function of Eq. 2 to yield Eq. 3, which can also be determined using the Viterbi algorithm. This is done by forming the cross product of the two lattices. This jointly selects the optimal detections to form the track, together with the optimal state sequence, and scores that combination.

$$\max_{\substack{j^1,\ldots,j^T \\ q^1,\ldots,q^T}} \left(\begin{array}{l} \sum_{t=1}^{T} F(D^t, j^t) \\ \quad + B(D^t, j^t, q^t, \lambda) \\ + \sum_{t=2}^{T} G(D^{t-1}, j^{t-1}, D^t, j^t) \\ \quad + A(q^{t-1}, q^t, \lambda) \end{array} \right) \quad (3)$$

While we formulated the above around a single track and a word that contains a single participant, it is straightforward to extend this so that it supports multiple tracks and words of higher arity by forming a larger cross product. When doing so, we generalize j^t to denote a sequence of detections from D^t, one for each of the tracks. We further need to generalize F so that it computes the joint score of a sequence of detections, one for each track, G so that it computes the joint measure of coherence between a sequence of pairs of detections in two adjacent frames, and B so that it computes the joint conditional log probability of observing the feature vectors associated with the sequence of detections selected by j^t. When doing this, note that Eqs. 1 and 3 maximize over $j^1, \ldots, j^T$ which denotes T sequences of detection indices, rather than T individual indices.

It is further straightforward to extend the above to support a sequence $(S_1, \ldots, S_L)$ of words S_l denoting a sentence, each of which applies to different subsets of the multiple tracks, again by forming a larger cross product. When doing so, we generalize q^t to denote a sequence $(q_1^t, \ldots, q_L^t)$ of states q_l^t, one for each word l in the sentence, and use q_l to denote the sequence $(q_l^1, \ldots, q_l^T)$ and q to denote the sequence $(q_1, \ldots, q_L)$. We further need to generalize B so that it computes the joint conditional log probability of observing the feature vectors for the detections in the tracks that are assigned to the arguments of the HMM for each word in the sentence and A so that it computes the joint log transition probability for the HMMs for all words in the sentence. This allows selection of an optimal sequence of tracks that yields the highest score for the sentential meaning of a sequence of words. Modeling the meaning of a sentence through a sequence of words whose meanings are modeled by HMMs, defines a *factorial* HMM for that sentence, since the overall Markov process for that sentence can be factored into inde-

pendent component processes (Brand et al., 1997; Zhong and Ghosh, 2001) for the individual words. In this view, q denotes the state sequence for the combined factorial HMM and q_l denotes the factor of that state sequence for word l. The remainder of this paper wraps this sentence tracker in Baum Welch (Baum et al., 1970; Baum, 1972).

4 Detailed Problem Formulation

We adapt the sentence tracker to training a corpus of R video clips, each paired with a sentence. Thus we augment our notation, generalizing j^t to j_r^t and q_l^t to $q_{r,l}^t$. Below, we use j_r to denote $(j_r^1, \ldots, j_r^{T_r})$, j to denote $(j_1, \ldots, j_R)$, $q_{r,l}$ to denote $(q_{r,l}^1, \ldots, q_{r,l}^{T_r})$, q_r to denote $(q_{r,1}, \ldots, q_{r,L_r})$, and q to denote $(q_1, \ldots, q_R)$.

We use discrete features, namely natural numbers, in our feature vectors, quantized by a binning process. We assume the part of speech of entry m is known as C_m. The length of the feature vector may vary across parts of speech. Let N_c denote the length of the feature vector for part of speech c, $x_{r,l}$ denote the time-series $(x_{r,l}^1, \ldots, x_{r,l}^{T_r})$ of feature vectors $x_{r,l}^t$, associated with $S_{r,l}$ (which recall is some entry m), and x_r denote the sequence $(x_{r,1}, \ldots, x_{r,L_r})$. We assume that we are given a function $\Phi_c(D_r^t, j_r^t)$ that computes the feature vector $x_{r,l}^t$ for the word $S_{r,l}$ whose part of speech is $C_{S_{r,l}} = c$. Note that we allow Φ to be dependent on c allowing different features to be computed for different parts of speech, since we can determine m and thus C_m from $S_{r,l}$. We choose to have N_c and Φ_c depend on the part of speech c and not on the entry m since doing so would be tantamount to encoding the to-be-learned word meaning in the provided feature-vector computation.

The goal of training is to find a sequence $\lambda = (\lambda_1, \ldots, \lambda_M)$ of parameters λ_m that best explains the R training samples. The parameters λ_m constitute the meaning of the entry m in the lexicon. Collectively, these are the initial state probabilities $a_{0,k}^m$, for $1 \le k \le I_{C_m}$, the transition probabilities $a_{i,k}^m$, for $1 \le i, k \le I_{C_m}$, and the output probabilities $b_{i,n}^m(x)$, for $1 \le i \le I_{C_m}$ and $1 \le n \le N_{C_m}$, where I_{C_m} denotes the number of states in the HMM for entry m. Like before, we could have a distinct I_m for each entry m but instead have I_{C_m} depend only on the part of speech of entry m, and assume that we know the fixed I for each part of speech. In our case, $b_{i,n}^m$ is a discrete distribution because the features are binned.

5 The Learning Algorithm

Instantiating the above approach requires a definition for what it means to *best explain the R training samples*. Towards this end, we define the score of a video clip D_r paired with sentence S_r given the parameter set λ to characterize how well this training sample is explained. While the cost function in Eq. 3 may qualify as a score, it is easier to fit a likelihood calculation into the Baum-Welch framework than a MAP estimate. Thus we replace the $\max$ in Eq. 3 with a $\sum$ and redefine our scoring function as follows:

$$L(D_r; S_r, \lambda) = \sum_{j_r} P(j_r|D_r) P(x_r|S_r, \lambda) \quad (4)$$

The score in Eq. 4 can be interpreted as an expectation of the HMM likelihood over all possible mappings from participants to all possible tracks. By definition, $P(j_r|D_r) = \frac{V(D_r, j_r)}{\sum_{j_r'} V(D_r, j_r')}$, where the numerator is the score of a particular track sequence j_r while the denominator sums the scores over all possible track sequences. The $\log$ of the numerator $V(D_r, j_r)$ is simply Eq. 1 without the $\max$. The $\log$ of the denominator can be computed efficiently by the forward algorithm (Baum and Petrie, 1966). The likelihood for a factorial HMM can be computed as:

$$P(x_r|S_r, \lambda) = \sum_{q_r} \prod_{l} P(x_{r,l}, q_{r,l}|S_{r,l}, \lambda) \quad (5)$$

i.e., summing the likelihoods for all possible state sequences. Each summand is simply the joint likelihood for all the words in the sentence conditioned on a state sequence q_r. For HMMs we have

$$P(x_{r,l}, q_{r,l}|S_{r,l}, \lambda) = \prod_t a_{q_{r,l}^{t-1}, q_{r,l}^t}^{S_{r,l}} \prod_n b_{q_{r,l}^t, n}^{S_{r,l}}(x_{r,l,n}^t) \quad (6)$$

Finally, for a training corpus of R samples, we seek to maximize the joint score:

$$L(D; S, \lambda) = \prod_r L(D_r; S_r, \lambda) \quad (7)$$

A local maximum can be found by employing the Baum-Welch algorithm (Baum et al., 1970; Baum, 1972). By constructing an auxiliary function (Bilmes, 1997), one can derive the reestimation formulas in Eq. 8, where $x_{r,l,n}^t = h$ denotes the selection of all possible j_r^t such that the nth

$$a^m_{i,k} = \theta^m_i \sum_{\substack{r=1 \\ s.t. S_{r,l}=m}}^{R} \sum_{l=1}^{L_r} \sum_{t=1}^{T_r} \underbrace{\frac{L(q^{t-1}_{r,l} = i, q^t_{r,l} = k, D_r; S_r, \lambda')}{L(D_r; S_r, \lambda')}}_{\xi(r,l,i,k,t)}$$

$$b^m_{i,n}(h) = \psi^m_{i,n} \sum_{\substack{r=1 \\ s.t. S_{r,l}=m}}^{R} \sum_{l=1}^{L_r} \sum_{t=1}^{T_r} \underbrace{\frac{L(q^t_{r,l} = i, x^t_{r,l,n} = h, D_r; S_r, \lambda')}{L(D_r; S_r, \lambda')}}_{\gamma(r,l,n,i,h,t)} \quad (8)$$

feature computed by $\Phi_{C_m}(D^t_r, j^t_r)$ is h. The coefficients θ^m_i and $\psi^m_{i,n}$ are for normalization.

The reestimation formulas involve *occurrence counting*. However, since we use a factorial HMM that involves a cross-product lattice and use a scoring function derived from Eq. 3 that incorporates both tracking (Eq. 1) and word models (Eq. 2), we need to count the frequency of transitions in the whole cross-product lattice. As an example of such cross-product occurrence counting, when counting the transitions from state i to k for the lth word from frame $t-1$ to t, *i.e.*, $\xi(r,l,i,k,t)$, we need to count all the possible paths through the adjacent factorial states $(j^{t-1}_r, q^{t-1}_{r,1}, \ldots, q^{t-1}_{r,L_r})$ and $(j^t_r, q^t_{r,1}, \ldots, q^t_{r,L_r})$ such that $q^{t-1}_{r,l} = i$ and $q^t_{r,l} = k$. Similarly, when counting the frequency of being at state i while observing h as the nth feature in frame t for the lth word of entry m, *i.e.*, $\gamma(r,l,n,i,h,t)$, we need to count all the possible paths through the factorial state $(j^t_r, q^t_{r,1}, \ldots, q^t_{r,L_r})$ such that $q^t_{r,l} = i$ and the nth feature computed by $\Phi_{C_m}(D^t_r, j^t_r)$ is h.

The reestimation of a single component HMM can depend on the previous estimate for other component HMMs. This dependence happens because of the argument-to-participant mapping which coindexes arguments of different component HMMs to the same track. **It is precisely this dependence that leads to cross-situational learning of two kinds: both inter-sentential and intra-sentential.** Acquisition of a word meaning is driven across sentences by entries that appear in more than one training sample and within sentences by the requirement that the meanings of all of the individual words in a sentence be consistent with the collective sentential meaning.

6 Experiment

We filmed 61 video clips (each 3–5 seconds at 640×480 resolution and 40 fps) that depict a variety of different compound events. Each clip depicts multiple simultaneous events between some subset of four objects: a person, a backpack, a chair, and a trash-can. These clips were filmed in three different outdoor environments which we use for cross validation. We manually annotated each video with several sentences that describe what occurs in that video. The sentences were constrained to conform to the grammar in Table 1. Our corpus of 159 training samples pairs some videos with more than one sentence and some sentences with more than one video, with an average of 2.6 sentences per video [1].

S	→	NP VP
NP	→	D N [PP]
D	→	*the*
N	→	*person* | *backpack* | *trash-can* | *chair*
PP	→	P NP
P	→	*to the left of* | *to the right of*
VP	→	V NP [ADV] [PPM]
V	→	*picked up* | *put down* | *carried* | *approached*
ADV	→	*quickly* | *slowly*
PPM	→	PM NP
PM	→	*towards* | *away from*

Table 1: The grammar used for our annotation and generation. Our lexicon contains 1 determiner, 4 nouns, 2 spatial relation prepositions, 4 verbs, 2 adverbs, and 2 motion prepositions for a total of 15 lexical entries over 6 parts of speech.

We model and learn the semantics of all words except determiners. Table 2 specifies the arity, the state number I_c, and the features computed by Φ_c for the semantic models for words of each part of speech c. While we specify a different subset of features for each part of speech, we presume that, in principle, with enough training data, we could include all features in all parts of speech and automatically learn which ones are noninformative and lead to uniform distributions.

We use an off-the-shelf object detector (Felzenszwalb et al., 2010a; Felzenszwalb et al., 2010b) which outputs detections in the form of scored axis-aligned rectangles. We trained four object detectors, one for each of the four object classes in

[1] Our code, videos, and sentential annotations are available at `http://haonanyu.com/research/acl2013/`.

c	arity	I_c	Φ_c
N	1	1	α detector index
V	2	3	α VEL MAG α VEL ORIENT β VEL MAG β VEL ORIENT α-β DIST α-β size ratio
P	2	1	α-β x-position
ADV	1	3	α VEL MAG
PM	2	3	α VEL MAG α-β DIST

Table 2: Arguments and model configurations for different parts of speech c. VEL stands for velocity, MAG for magnitude, ORIENT for orientation, and DIST for distance.

our corpus: person, backpack, chair, and trashcan. For each frame, we pick the two highest-scoring detections produced by each object detector and pool the results yielding eight detections per frame. Having a larger pool of detections per frame can better compensate for false negatives in the object detection and potentially yield smoother tracks but it increases the size of the lattice and the concomitant running time and does not lead to appreciably better performance on our corpus.

We compute continuous features, such as velocity, distance, size ratio, and x-position solely from the detection rectangles and quantize the features into bins as follows:

velocity To reduce noise, we compute the velocity of a participant by averaging the optical flow in the detection rectangle. The velocity magnitude is quantized into 5 levels: *absolutely stationary*, *stationary*, *moving*, *fast moving*, and *quickly*. The velocity orientation is quantized into 4 directions: *left*, *up*, *right*, and *down*.

distance We compute the Euclidean distance between the detection centers of two participants, which is quantized into 3 levels: *near*, *normal*, and *far away*.

size ratio We compute the ratio of detection area of the first participant to the detection area of the second participant, quantized into 2 possibilities: *larger/smaller than*.

x-position We compute the difference between the x-coordinates of the participants, quantized into 2 possibilities: *to the left/right of*.

The binning process was determined by a preprocessing step that clustered a subset of the training data. We also incorporate the index of the detector that produced the detection as a feature. The particular features computed for each part of speech are given in Table 2.

Note that while we use English phrases, like *to the left of*, to refer to particular bins of particular features, and we have object detectors which we train on samples of a particular object class such as *backpack*, such phrases are only mnemonic of the clustering and object-detector training process. We do not have a fixed correspondence between the lexical entries and any particular feature value. Moreover, that correspondence need not be one-to-one: a given lexical entry may correspond to a (time variant) constellation of feature values and any given feature value may participate in the meaning of multiple lexical entries.

We perform a three-fold cross validation, taking the test data for each fold to be the videos filmed in a given outdoor environment and the training data for that fold to be all training samples that contain other videos. For testing, we hand selected 24 sentences generated by the grammar in Table 1, where each sentence is true for at least one test video. Half of these sentences (designated `NV`) contain only nouns and verbs while the other half (designated `ALL`) contain other parts of speech. The latter are longer and more complicated than the former. We score each testing video paired with every sentence in both `NV` and `ALL`. To evaluate our results, we manually annotated the correctness of each such pair.

Video-sentence pairs could be scored with Eq. 4. However, the score depends on the sentence length, the collective numbers of states and features in the HMMs for words in that sentence, and the length of the video clip. To render the scores comparable across such variation we incorporate a sentence prior to the per-frame score:

$$\hat{L}(D_r, S_r; \lambda) = [L(D_r; S_r, \lambda)]^{\frac{1}{T_r}} \pi(S_r) \quad (9)$$

where

$$\pi(S_r) = \exp \sum_{l=1}^{L_r} \left(\begin{array}{c} E(I_{C_{S_{r,l}}}) \\ + \sum_{n=1}^{N_{C_{S_{r,l}}}} E(Z_{C_{S_{r,l}},n}) \end{array} \right) \quad (10)$$

In the above, $Z_{C_{S_{r,l}},n}$ is the number of bins for the nth feature of $S_{r,l}$ of part of speech $C_{S_{r,l}}$ and $E(Y) = -\sum_{y=1}^{Y} \frac{1}{Y} \log \frac{1}{Y} = \log Y$ is the entropy of a uniform distribution over Y bins. This prior prefers longer sentences which describe more information in the video.

	CHANCE	BLIND	OUR	HAND
NV	0.155	0.265	0.545	0.748
ALL	0.099	0.198	0.639	0.786

Table 3: F1 scores of different methods.

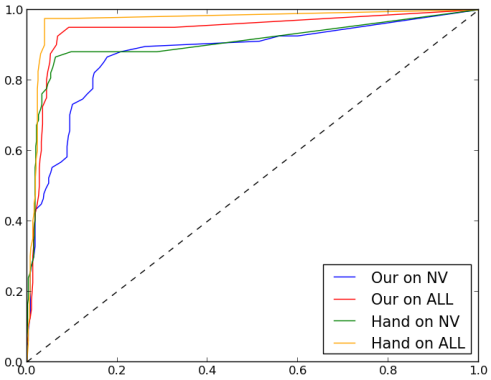

Figure 2: ROC curves of trained models and hand-written models.

The scores are thresholded to decide hits, which together with the manual annotations, can generate TP, TN, FP, and FN counts. We select the threshold that leads to the maximal F1 score on the training set, use this threshold to compute F1 scores on the test set in each fold, and average F1 scores across the folds.

The F1 scores are listed in the column labeled **Our** in Table 3. For comparison, we also report F1 scores for three baselines: **Chance**, **Blind**, and **Hand**. The **Chance** baseline randomly classifies a video-sentence pair as a hit with probability 0.5. The **Blind** baseline determines hits by potentially looking at the sentence but never looking at the video. We can find an upper bound on the F1 score that any blind method could have on each of our test sets by solving a 0-1 fractional programming problem (Dinkelbach, 1967) (see Appendix A for details). The **Hand** baseline determines hits with hand-coded HMMs, carefully designed to yield what we believe is near-optimal performance. As can be seen from Table 3, our trained models perform substantially better than the **Chance** and **Blind** baselines and approach the performance of the ideal **Hand** baseline. One can further see from the ROC curves in Figure 2, comparing the trained and hand-written models on both NV and ALL, that the trained models are close to optimal. Note that performance on ALL exceeds that on NV with the trained models. This is because longer sentences with varied parts of speech incorporate more information into the scoring process.

7 Conclusion

We presented a method that learns word meanings from video paired with sentences. Unlike prior work, our method deals with realistic video scenes labeled with whole sentences, not individual words labeling hand delineated objects or events. The experiment shows that it can correctly learn the meaning representations in terms of HMM parameters for our lexical entries, from highly ambiguous training data. Our maximum-likelihood method makes use of only positive sentential labels. As such, it might require more training data for convergence than a method that also makes use of negative training sentences that are not true of a given video. Such can be handled with discriminative training, a topic we plan to address in the future. We believe that this will allow learning larger lexicons from more complex video without excessive amounts of training data.

Acknowledgments

This research was sponsored 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.

A An Upper Bound on the F1 Score of any Blind Method

A **Blind** algorithm makes identical decisions on the same sentence paired with different video clips. An optimal algorithm will try to find a decision s_i for each test sentence i that maximizes the F1 score. Suppose, the ground-truth yields FP_i false positives and TP_i true positives on the test set when $s_i = 1$. Also suppose that setting $s_i = 0$ yields FN_i false negatives. Then the F1 score is

$$F1 = \frac{1}{1 + \underbrace{\frac{\sum_i s_i \text{FP}_i + (1 - s_i)\text{FN}_i}{\sum_i 2 s_i \text{TP}_i}}_{\Delta}}$$

Thus we want to minimize the term Δ. This is an instance of a 0-1 fractional programming problem which can be solved by binary search or Dinkelbach's algorithm (Dinkelbach, 1967).

References

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

A. Barbu, N. Siddharth, A. Michaux, and J. M. Siskind. 2012b. Simultaneous object detection, tracking, and event recognition. *Advances in Cognitive Systems*, 2:203–220, December.

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

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

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

J. Bilmes. 1997. A gentle tutorial of the EM algorithm and its application to parameter estimation for Gaussian mixture and hidden Markov models. Technical Report TR-97-021, ICSI.

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

D. L. Chen and R. J. Mooney. 2008. Learning to sportscast: A test of grounded language acquisition. In *Proceedings of the 25th International Conference on Machine Learning*.

W. Dinkelbach. 1967. On nonlinear fractional programming. *Management Science*, 13(7):492–498.

P. F. Dominey and J.-D. Boucher. 2005. Learning to talk about events from narrated video in a construction grammar framework. *Artificial Intelligence*, 167(12):31–61.

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

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

G. Kulkarni, V. Premraj, S. Dhar, S. Li, Y. Choi, A. C. Berg, and T. L. Berg. 2011. Baby talk: Understanding and generating simple image descriptions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1601–1608.

P. Kuznetsova, V. Ordonez, A. C. Berg, T. L. Berg, and Y. Choi. 2012. Collective generation of natural image descriptions. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers - Volume 1*, pages 359–368.

T. Kwiatkowski, S. Goldwater, L. Zettlemoyer, and M. Steedman. 2012. A probabilistic model of syntactic and semantic acquisition from child-directed utterances and their meanings. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 234–244.

V. Ordonez, G. Kulkarni, and T. L. Berg. 2011. Im2text: Describing images using 1 million captioned photographs. In *Proceedings of Neural Information Processing Systems*.

D. Roy. 2002. Learning visually-grounded words and syntax for a scene description task. *Computer Speech and Language*, 16:353–385.

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

J. M. Siskind and Q. Morris. 1996. A maximum-likelihood approach to visual event classification. In *Proceedings of the Fourth European Conference on Computer Vision*, pages 347–360.

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

T. Starner, J. Weaver, and A. Pentland. 1998. Real-time American Sign Language recognition using desk and wearable computer based video. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(12):1371–1375.

A. J. Viterbi. 1967. Error bounds for convolutional codes and an asymtotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, 13:260–267.

A. Viterbi. 1971. Convolutional codes and their performance in communication systems. *IEEE Transactions on Communication Technology*, 19(5):751–772.

J. Yamoto, J. Ohya, and K. Ishii. 1992. Recognizing human action in time-sequential images using hidden Markov model. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 379–385.

B. Z. Yao, X. Yang, L. Lin, M. W. Lee, and S.-C. Zhu. 2010. I2T: Image parsing to text description. *Proceedings of the IEEE*, 98(8):1485–1508, August.

C. Yu and D. H. Ballard. 2004. On the integration of grounding language and learning objects. In *Proceedings of the 19th National Conference on Artifical intelligence*, pages 488–493.

S. Zhong and J. Ghosh. 2001. A new formulation of coupled hidden Markov models. Technical report, Department of Electrical and Computer Engineering, The University of Texas at Austin.