**A Uniform Representation for Visual Concepts**

by Nicolas Rakover

B.S., C.S. M.I.T., 2015

Submitted to the
Department of Electrical Engineering and Computer Science
in Partial Fulfillment of the Requirements for the Degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

Massachusetts Institute of Technology

June 2016



Author: _____

Department of Electrical Engineering and Computer Science
April 4, 2016

Certified by: _____

Boris Katz
Principal Research Scientist, MIT CSAIL
Thesis Supervisor
April 4, 2016

Accepted by: _____

Dr. Christopher J. Terman, Chairman, Masters of Engineering Thesis Committee

# A Uniform Representation for Visual Concepts

by
Nicolas Rakover

Submitted to the Department of Electrical Engineering and Computer Science on
April 4, 2016 in Partial Fulfillment of the Requirements for the Degree of Master of Engineering
in Electrical Engineering and Computer Science

## ABSTRACT

We present a method for learning visually-grounded word meanings, given as input a set of videos paired with natural-language sentences describing them. Our method uses a uniform feature representation for all words and word types rather than relying on hand-crafted features specific to each word. We learn words in a weakly-supervised manner, with no need for annotated bounding boxes around objects of interest. We encode words as Hidden Markov models such that word models can be composed according to a sentence's semantic structure to efficiently recognize events in videos. We use a discriminative variant of Baum-Welch to learn the parameters for our word models, and demonstrate that our approach is able to learn words capturing appearance, spatial relations, and temporal dynamics.

Thesis Supervisor: Boris Katz
Title: Principal Research Scientist, MIT CSAIL

# Contents

# 1 Introduction

Some systems today solve complicated scene understanding tasks. Siddharth et al. (2014), for instance, use natural language to describe simple events and to guide disambiguation of scenes depicting multiple events. However, in order to learn to recognize words or events, existing approaches often rely on hand-crafted features or explicit annotation of training examples with bounding boxes. Hand-crafted features tend to work well for the specific kinds of objects, attributes, or events they were designed for, but their performance often fails to generalize. Moreover, annotating images with bounding boxes for learning object detectors is a tedious manual process that simply doesn't scale to the number of objects that humans can effortlessly learn.

We propose a system to learn the meanings of words given videos labeled only with natural-language sentence descriptions. The goal is to do this in a weakly-supervised manner, meaning the videos have no bounding boxes around objects or any additional annotations apart from the the labeling sentences. In order to make the system flexible to the kinds of words it can learn, we will have a uniform representation for the features used by all the words, where the relevance of each feature is learned on a per-word basis. Features can be any discernible cue in the video that could characterize a word -- verbs might be characterized by motion, adjectives by textures or shape, nouns by object class identity. With hand-crafted features one needs to know a priori which cues are relevant to each word, and scaling to new words might involve incorporating new cues. Using a common representation for all features, on the other hand, allows a system to be agnostic to word types, making it more easily extensible to new words and word types.

Our common feature representation will include the output from the last hidden layer of a pre-trained convolutional neural network (Simonyan & Zisserman, 2014). As in (Siddharth et al., 2014), words will be encoded by Hidden Markov models over image features, and these HMMs can be composed according to a sentence's semantics. Namely, the dependency parse for the sentence is used to determine the number of participants in the described event and how each word in the sentence relates to the participants. In addition to the word HMMs, there is a tracker HMM for each participant that localizes it in each frame of the video. The word HMMs observe the outputs of corresponding trackers, where the mapping from words to trackers is determined from the dependency parse. The word and tracker HMMs are composed into a *sentence tracker*. Given a video-sentence pair, the sentence tracker scores how closely the sentence matches the scene depicted by the video.

Whereas Siddharth et al. (2014) used hand-crafted features in the HMM emission models, we will use our uniform feature representation and model HMM emissions with linear classifiers. By encoding all words through the same representation we can jointly learn many word classes, including nouns, adjectives and verbs along with the corresponding object detectors, attribute detectors and action detectors.

Object detectors are typically trained using images labeled with the presence or absence of the object of interest along with a bounding box drawn around the relevant image region. This is not only time-consuming but also fails to provide an explanation of how children learn without explicit supervision and bounding boxes. By doing this task in a weakly-supervised framework, without explicit bounding boxes, and relying on the co-occurrence of words in different sentences and videos we aim to provide a more cognitively-plausible approach.

Weakly-supervised approaches to learning object detectors already exist (Nguyen et al., 2009; Crandall & Huttenlocher, 2006; Prest et al., 2012) but they do not incorporate any other

parts of speech such as verbs or adjectives. Typically, weakly-supervised refers to the fact that images are labeled only with the presence or absence of an object class, but we extend this by labeling entire videos with whole sentences. Learning from videos makes the problem harder in some sense, since there is more ambiguity about what each word could refer to. At the same time, this adds more constraints derived from laws that object follow when moving and interacting with other objects. We leverage these physical constraints and those imposed by sentence structure to reduce ambiguity and learn word meanings.

# 2 Related Work

## 2.1 Sentence-guided Activity Recognition

Many approaches for activity recognition take inspiration from techniques developed for object detection tasks and often extend these by incorporating motion or other temporal cues. Thurau and Hlaváč (2008) extend the use of histograms of oriented gradients (Dalal & Triggs, 2005) to recognition of human actions in videos and still images. Ali and Shah (2010) propose a set of features derived from optical flow to recognize human actions in videos.

Sentence-guided activity recognition is a way to leverage high-level knowledge about the world in the activity recognition task. The work done by Siddharth et al. (2014) exploits structural similarities between spatio-temporal visual events and the semantics expressed by sentences. They integrate the high-level information provided by the sentence with the low-level visual signals in the video to score a sentence-video pair.

Their system uses two kinds of HMMs -- one to track object detections and one to model words. The tracker HMMs select a detection from each frame such that the track of detections optimizes detection confidence and temporal coherence, meaning the motion of the object is smooth. Each word is modeled by an HMM over hand-crafted features extracted from the detections in a track. Features include the average optical flow vector, horizontal position, object class, and average color of the detection region. Each word in a sentence selects an object track that optimizes the state dynamics of its HMM, constrained by the word relations imposed by the sentence. By jointly optimizing the words and trackers, the appropriate tracks are identified and the sentence-video pair can be scored.

The main shortcoming of this work that we address is its reliance on hand-crafted features.

## 2.2 Weakly-supervised Classification and Localization

A related line of work is weakly-supervised classification and localization. In an early approach, Crandall and Huttenlocher (2006) learned part-based object models from weakly labeled images. They introduced a method to jointly learn the appearance of object parts and the spatial relations between them. Recently, Wang et al. (2014) used a latent category learning scheme, inspired by the topic modeling literature. They jointly modeled objects and backgrounds and learned to suppress the background to facilitate object classification. Bilen et al. (2014) proposed an approach that leverages features extracted by convolutional neural networks along with domain-specific knowledge. They show the importance of structuring the distributions of latent variables that guide localization in the weakly-supervised setting and demonstrate the benefits of using soft-max over the localization scores as opposed to a hard maximum to enable detection of multiple object instances.

Nguyen et al. (2009) employ an EM-like optimization scheme that jointly learns to classify objects and to localize them within the image. They generalize their method to classification of time series, rather than still images, and propose a representation for image sequences to enable detection of complex events. Prest et al. (2012) use videos known to contain objects of a target class to improve the performance of detectors trained on fully annotated still images.

In contrast to the approaches mentioned in this section, we learn exclusively from videos. Moreover, these videos are labeled with sentences, not the presence or absence of

specific object classes. We learn several parts of speech, and the words we learn can be composed to encode events that were never seen during training.

## 2.3 Neural Network-based Feature Extraction

One of the main goals for our work is to use a common representation from which to derive all the words. To do this we leveraged convolutional neural networks as feature extractors. Networks such as OverFeat (Sermanet et al., 2013) and VGG (Simonyan & Zisserman, 2014) have achieved state-of-the-art performance on object classification and localization benchmarks (Russakovsky et al., 2014; Everingham et al., 2015). Evidence from object detection shows that neural networks learn representations that are more invariant with respect to high-level features than the raw pixels, which enable learning with significantly fewer examples. We use such pre-trained networks to extract features from image regions, rather than let our word models observe the raw pixels directly.

## 2.4 Discriminant HMMs

As in (Siddharth et al., 2014), we are going to model the underlying dynamics of videos using Hidden Markov models (HMMs). However, discriminative models often outperform generative models particularly when the features being modeled are not interpretable and come from the output of a neural network. To this end we employed HMMs that use linear classifiers as output models.

Several approaches have demonstrated the power of these hybrid or discriminant HMMs. For example, Valstar and Pantic (2007) use a hybrid SVM-HMM to model dynamics of facial expressions. Their method proved to model the time dynamics of facial expressions as well as traditional HMMs, but offered much better discriminative performance than variants using Gaussian mixture emission models. Ganapathiraju et al. (2000) use a similar model for

speech recognition. Their system outperformed traditional HMMs while requiring only a fraction of the training data. Both papers address the interpretation of the SVM score as the posterior probability that the HMM expects by passing the score through a sigmoid with parameters learned by cross-validation. Altun et al. (2013) propose the HM-SVM, which encodes the dependency structure of an HMM within the constraints of an SVM. This model is capable of learning nonlinear discriminant functions over features that can cross time-steps, yet it can be efficiently trained and decoded due to its Markov structure.

A related topic is discriminative training of HMMs. Collins (2002) proposes a discriminative training method for HMMs that extends the perceptron algorithm (Rosenblatt, 1958) for learning linear classifiers. It involves comparing a known, desired state sequence with the most likely state sequence as determined by the current HMM parameters, and penalizing discrepancies between the two sequences. The algorithm is shown to converge on HMM parameters that score the correct sequences higher than the competing sequences. This approach requires known state sequences for training, but we avoid this constraint by wrapping the algorithm's update rule with the expectation-maximization algorithm.

## 2.5 Class-independent Object Proposals

In order to avoid relying on annotated bounding boxes for training, we use an off-the-shelf object proposal mechanism to identify image regions likely to contain objects of interest. Generic object detection, as the task is often called, attempts to localize all the objects in an image using cues, such as edges and saliency, that are class-independent (Chavali et al., 2015). Alexe et al. (2012) propose an objectness measure for image regions that incorporates, among other features, a metric for closed boundary. Manen et al. (2013) present an efficient algorithm for this task based on randomized Prim's algorithm (Prim, 1957). While the majority of the object

proposals for a particular video-sentence pair are irrelevant to our task, the proposal set does cover the regions of interest and is small enough for our inference to be tractable.

# 3 Methods

## 3.1 Overview

We built a system for learning visually-grounded word meanings, where all words are encoded with a uniform representation as Hidden Markov models. Our system is able to learn the meaning of diverse classes of words given as input a set of videos paired with sentences describing them. The words learned by the system can be easily composed to encode whole sentences, which can then be used to disambiguate a video and to track different objects of interest.

Previous approaches for learning visually-grounded word meanings either used hand-crafted features specific to each word or relied on more explicit annotation of videos and images, with bounding boxes placed around objects of interest. Both hand-crafted features and object-level annotation present a bottleneck for scaling up a system to learn large collections of words, since different words often depend on very different features and the annotation process is a tedious manual endeavor.

The core of our contribution is the use of a uniform representation for all words, where a word is encoded as an HMM that models word-independent image features extracted by a pre-trained neural network as well as optical flow and position features. This allows the system to learn a large and diverse set of words that encode any combination of appearance, spatial relation, motion, and temporal dynamics. Additionally, we avoid object-level annotation by relying on a pre-trained object proposal generator and leveraging the mutual constraints imposed by words in a sentence to select the correct proposals.

We will consider the system successful if it learns meanings for words such that we can use these words to disambiguate scenes. For example, in order to test whether the system correctly learned the meaning of the verbs ‘approach’ and ‘leave’, we would show the system a video of two people, where one person approaches a bin and the other person walks away from the bin. If we ask the system to show us the person approaching the bin, we expect it to identify the correct person, and if we ask for the person leaving the bin we would expect it to identify the other person. We want to demonstrate that the system is able to learn words that capture appearance, motion, and spatial relations.

The following sections describe our system in detail. Section 3.2 describes the input to the system and how we preprocess the videos. Section 3.3 presents how the system encodes words and sentences, and how it uses this encoding to score object tracks through a video. In section 3.4 we discuss our approach for filtering object proposals as a means of making the inference and learning algorithms tractable. Finally, section 3.5 discusses how we learn words.

## 3.2 Input

Our objective is to learn the meanings of words given only videos and sentences describing them. More formally, let our input be $V$ and $S$. $V$ is a set of videos, where the $i^{th}$ video $v_i \in V$ is a sequence of frames. Let $S_i^P \subset S$ be a set of sentences that are true about video $v_i$, whereas $S_i^N \subset S$ is a set of sentences that are false about video $v_i$. We refer to $S_i^P$ as the *positive* sentences for $v_i$, and $S_i^N$ as the *negative* sentences for $v_i$.

We first run the videos through a preprocessing phase that converts them into a format more readily used by the rest of our system. This consists primarily of generating object proposals, extracting image features, and computing optical flow.

First, we downsample each video frame to 256x192, which allows us to cover the objects of interest with a manageable number of object proposals. We use a frame rate of 3 fps; lower frame rates can result in inadequate optical flow, while a large number of frames makes for a longer preprocessing phase.

Next we generate object proposals for each video frame. Object proposals are axis-aligned rectangular regions of an image with a high probability of encompassing an object, and they are generated using signals that try to be agnostic to object class. We often refer to an object proposal as a *detection*. Using object proposals allows our system to consider a drastically smaller set of image regions than if we enumerated every possible box in a frame. We use a MATLAB toolkit developed by Chavali et al. (2015) to generate 250 class-agnostic object proposals for each video frame. The algorithm used, proposed by Manen et al. (2013), is based on Randomized Prim's algorithm (Prim, 1957). In addition to these object proposals, we use the MATLAB built-in Person Detector (Dalal & Triggs, 2005) to generate proposal detections for people in each frame. We set the classification threshold low to achieve near-perfect recall. We use this additional proposal mechanism because in our experience the class-agnostic object proposals, while generally suitable for simpler or smaller objects, sometimes failed to generate the desired bounding boxes around people.

Once we have object proposals, we want to compute a more readily useful representation of the corresponding image regions than the raw pixels. We use the 19-layer VGG network (Simonyan & Zisserman 2014) to extract features from each proposal region. It has been shown that such features more succinctly encode the information necessary for recognizing object classes, shapes, and textures than the raw pixels they are extracted from, so we hypothesize that using these features will enable our system to learn from fewer examples. We take the 4096-dimensional output of the second fully-connected layer of the network.

Lastly, we compute the optical flow (Horn & Schunck, 1981) for each video frame using an implementation of Liu's algorithm (Liu 2009). We convert the optical flow into integral optical flow so that finding the average flow in subregions of a frame becomes a constant-time operation. If the optical flow at pixel $i,j$ is $u_{ij}$, then the integral optical flow at pixel $i,j$ is equal to $u^I{}_{ij} = \sum_{i' \le i, j' \le j} u_{i'j'}$. Then finding the total flow in the region $[x_1, x_2] \times [y_1, y_2]$ reduces to the four arithmetic operations $u^I{}_{y_2 x_2} + u^I{}_{y_1 x_1} - u^I{}_{y_2 x_1} - u^I{}_{y_1 x_2}$.

## 3.3 Sentence Tracker

The sentence tracker is the workhorse of our system. It is based on (Siddharth et al., 2014) and allows us to leverage the semantic structure of a sentence in order to constrain the tracks of each word in the sentence, which is crucial for the inference our system performs.

### Relevant Terms

A track is a sequence of bounding boxes through the frames of a video. The simplest kind of track is an object track, which consists of a single bounding box per frame, *tracking* the object's position through the video. A word can also have a track, which consists of a track for each object referred to by the word, or each of the word's *arguments*. For example, an adjective's track would be the track for the object it modifies, while the track for a transitive verb would consist of the track for its subject and the track for its object. Similarly, a sentence has a track consisting of one track for each object participating in the sentence.

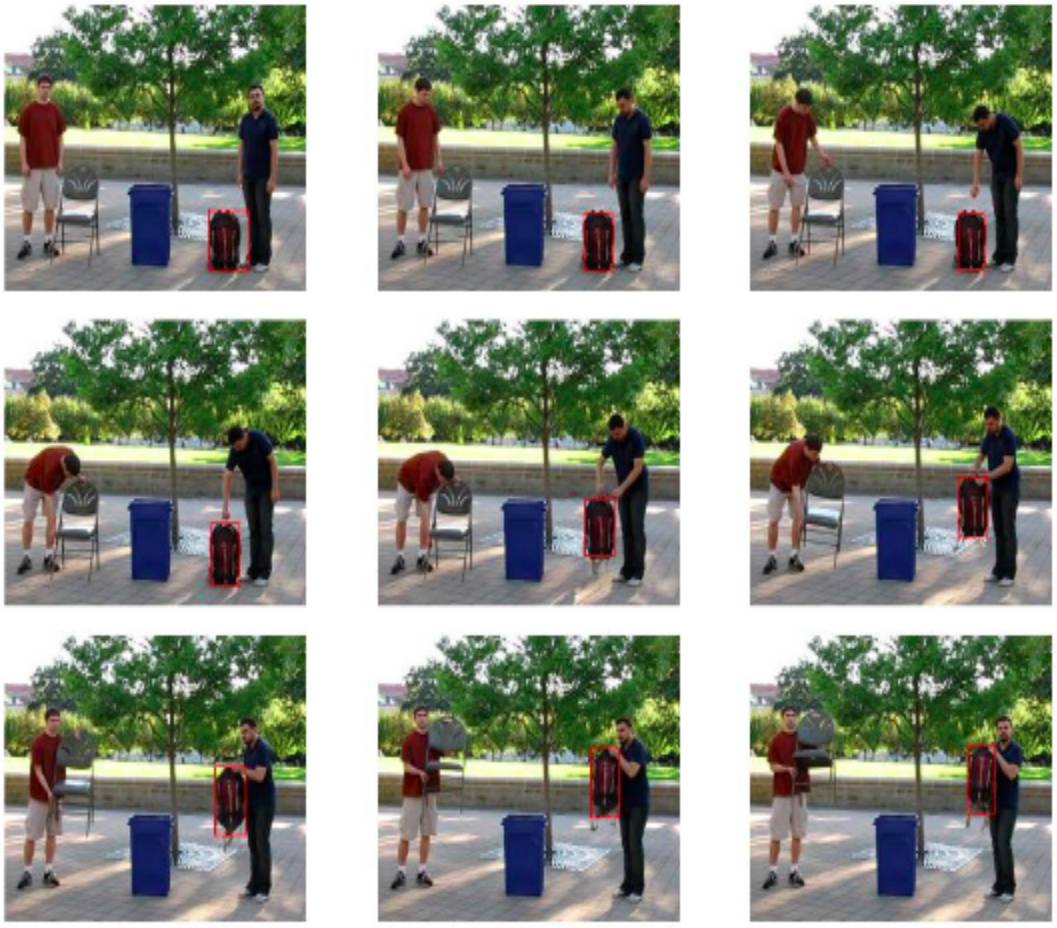


Figure 1. The red rectangles, one per frame of the video, show a *track* for the backpack.

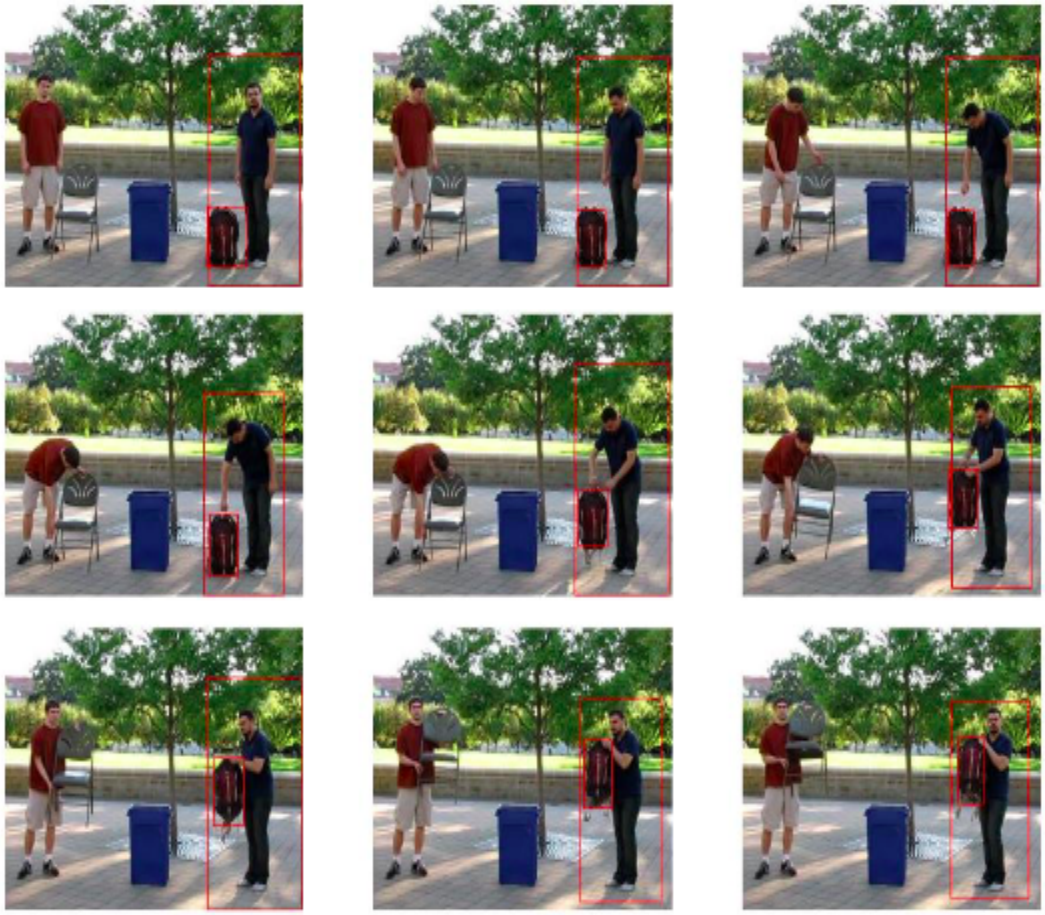


Figure 2. The red rectangles, two per video frame, depict a track for the sentence “the person picks up the backpack”. The sentence track consists of a track for each of its two participants: one for “backpack” and one for “person”.

Given a sentence, we refer to the objects involved as the sentence’s *participants*. Our use of the word participant is similar to the linguistic notion of thematic roles (Baker, 1998), and in particular thematic relations (Jackendoff, 1987). For example, the sentence “the person approaches the red backpack” has two participants: one is “person” and the other is“backpack”. In terms of thematic relations, “person” would be considered the *agent* and “backpack” the

*theme* or *patient*. While we're not interested in categorizing participants, the number of participants in a sentence as well as how they relate to the words in the sentence is central.

Both of these pieces of information can be derived from the dependency parse of a sentence, a formal language for describing the relations between words in a sentence (Mel'čuk, 1988). Take, for instance, the dependency parse of "the person approaches the red backpack", depicted as a parse tree in figure 3. The parse tree shows that the sentence has two participants, one for each noun in the sentence. Moreover, the parse tree tells us how each word relates to the participants. Specifically, it tells us the the track for participant R1 should also be a track for the word "person", the word "the", and the track for the 1st argument, or subject, of the word "approaches". Similarly, the track for participant R2 should be a track for the words "backpack", "red", "the", and the 2nd argument, or object, of the word "approaches".

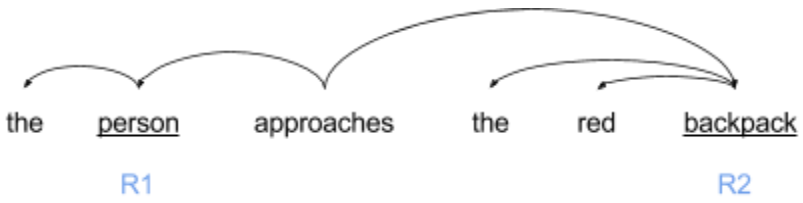



Figure 3. Shows the dependency parse of "the person approaches the red backpack". The sentence has two participants: "person" and "backpack".

## Hidden Markov Models

Our system makes extensive use of Hidden Markov models, or HMMs (Rabiner & Juang, 1986). HMMs are a class of temporal probabilistics models that are widely used for modeling sequences or events that evolve over time. HMMs can be thought of as two parallel sequences: a sequence of states that are not directly observed, and a sequence of observations that are dependent on the underlying states. HMMs make what is known as the Markov assumption,

which states that knowing the state at step $i$ fully characterizes the distribution over states at step $i+1$. Moreover, the distribution over observations at step $i$ is dependent exclusively on the state at step $i$. This strict dependence structure makes it possible to do efficient inference with HMMs, as well as to learn the model's parameters given observed data.

Formally, an HMM can be represented with the parameters $\theta = \{a, b, \pi, \tau\}$. Given states $p$ and $q$, $a(p,q) = p(q|p)$ gives the probability of transitioning from $p$ to $q$, where $\sum_q a(p,q) = 1$. Given observation $x$, $b(p,x) = p(x|p)$ gives the probability of observing $x$ conditioned that the state is $p$, also referred to as the *emission* probability or probability of state $p$ *emitting* observation $x$. $\pi$ is known as the prior state distribution, namely $\pi(p)$ is the probability that a sequence would begin with state $p$ independent of the observations, and $\sum_p \pi(p) = 1$. Similarly, $\tau$ is a terminal distribution such that $\tau(p)$ is the probability that a sequence would end with state $p$, and $\sum_p \tau(p) = 1$.

If we know these parameters $\theta$ we can perform various inference tasks given a sequence of observations, such as scoring the likelihood of state sequences, finding the most likely state sequence, and computing the likelihood that the observations were generated by the process modeled by the parameters. Say we have a sequence of observations $x = (x_1, x_2, \cdots, x_n)$ and a sequence of states $y = (y_1, y_2, \cdots, y_n)$. The joint likelihood of the observations and state sequence is

$$p(x, y|\theta) = \pi(y_1) \cdot \prod_{i=2}^{n} a(y_{i-1}, y_i) \cdot \prod_{i=1}^{n} b(y_i, x_i) \cdot \tau(y_n)$$

Then the most likely state sequence given the observations is $y^* = arg\max_y p(x, y|\theta)$. We can solve for this sequence $y^*$ in polynomial time using the Viterbi algorithm (Viterbi, 1967).

Viterbi is a recursive algorithm that lends itself to a dynamic programing implementation. We now give a brief overview of the algorithm. Given a fixed observation sequence $x = (x_1, \cdots, x_n)$ and a state sequence $y = (y_1, \cdots, y_n)$, we define a truncated version of the joint probability $p(x, y)$ that considers only the first $k$ states and observations. Namely, take

$$r(y_1, y_2, \cdots, y_k) = \pi(y_1) \cdot \prod_{i=2}^{k} a(y_{i-1}, y_i) \cdot \prod_{i=1}^{k} b(y_i, x_i)$$

Now let $S(k, v)$ be the set of state sequences $(y_1, y_2, \cdots, y_k)$ of length $k$ that end with state $y_k = v$. The Viterbi recurrence, which gives the highest score for any state sequence of length $k$ ending in state $v$, is then

$$V(k, v) = \max_{(y_1, \cdots, y_k) \in S(k,v)} r(y_1, \cdots, y_k) = \max_u \{ V(k-1, u) \cdot a(u, v) \cdot b(v, x_k) \}$$

where $u$ is a potential state for step $k - 1$. The base case of the recurrence is $V(1, v) = \pi(v) \cdot b(v, x_1)$ for any state $v$. Then the best joint probability we can achieve given the observation sequence $x$ is

$$\max_{y=(y_1, \cdots, y_n)} p(x, y|\theta) = \max_u \{ V(n, u) \cdot \tau(u) \}$$

Note that this gives us the best score, not the highest-scoring sequence $y^*$. However, getting the sequence is a straightforward process known as *backtracking*. In the Viterbi recurrence, we substitute the $\max$ operation with an $arg\max$ operation. Then we have that

$$y^*_n = arg\max_v \{ V(n, v) \cdot \tau(v) \}$$

With the last state $y^*_n$ fixed, we can find $y^*_{n-1} = arg\max_v \{ V(n-1, v) \cdot a(v, y^*_n) \}$. More generally, once we have $y^*_k$ we see that

$$y^*_{k-1} = arg\max_v \{ V(k-1, v) \cdot a(v, y^*_k) \}$$

The backtracking process can also be done in polynomial time and finds the most likely state sequence $y^*$ given the observation sequence $x$.

## The Sentence Tracker

We now describe the sentence tracker, which we based on (Siddharth et al., 2014). Given a sentence and a video with a predefined set of object proposals, the sentence tracker encodes the sentence's semantics and can score tracks through a video based on their likelihood of portraying the sentence. The sentence tracker is a collection of HMMs. Specifically, there is a *tracker HMM* for each participant in the sentence, and a *word HMM* for each word in the sentence.

A tracker HMM is word agnostic, and scores a single-participant track based on the *temporal coherence* of the detections. Temporal coherence is a way of capturing physical constraints about the motion of objects in the real world; namely, objects do not teleport but rather move smoothly through space, and generally objects don't sporadically change in size. These two principles are captured by two kinds of temporal coherence: coherence of position, and coherence of extent.

In the terminology we established, the states a tracker HMM can take on at step $k$ are the indices of the object proposals for frame $k$, but there is no observation sequence. Thus, a state sequence is scored exclusively based on the transition function $a(\cdot,\cdot)$, which we describe next.

In order to encode the temporal coherence constraints, the tracker HMM uses the average optical flow and size of detections. For coherence of position, it uses the average flow within a detection to forward-project it onto the next frame and computes the distance between

the center of this projected detection and the center of the actual detection in the next frame -- the smaller the distance the higher the score. Additionally, the tracker gives a score based on the similarity of the size of adjacent detections, or coherence of extent. Then, given detection $i_k$ for frame $k$ and detection $j_{k+1}$ for frame $k+1$, the transition function is given by

$$a_{tracker}(i_k, j_{k+1}) = (1 - \frac{dist(i_k, j_{k+1})}{dist_{max}})^C \cdot (1 - \frac{|area(i_k) - area(j_{k+1})|}{area(i_k) + area(j_{k+1})})$$

where $dist(i_k, j+k+1)$ is the Euclidean distance between the center of detection $j_{k+1}$ and the forward-projected center of $i_k$, $dist_{max}$ is the diagonal length of the frame, $area(i_k)$ and $area(j_{k+1})$ are the areas in square pixels of the two detections, and $C$ is a constant. We use $C = 80$. Given a track $t = (d_1, d_2, \cdots, d_n)$ for an $n$-frame video, the tracker HMM assigns it a score of

$$score_{tracker}(t) = \prod_{k=1}^{n-1} a_{tracker}(d_k, d_{k+1})$$

Effectively, the tracker HMM favors tracks that are smooth across frames in both displacement and size.

A word HMM encodes the appearance and dynamics of a word, be it a noun, adjective, verb, or preposition. It scores a track based on how well it portrays the word it models. Word HMMs have $N$ states, so at step $k$ they can be in any state $i \in \{1, 2, \cdots, N\}$. The observation at step $k$ is the image features, optical flow, and relative positions of the $k^{\text{th}}$-frame detections in the track. Each state has an emission model, implemented as logistic regression, which scores observations. Additionally, we constrain any state sequence of the word HMMs to begin at state 1 and terminate at state $N$, and, for all $i$, state $i$ can either self-loop or proceed to state $i+1$. Namely,

$$\pi(1) = 1, \quad \tau(N) = 1, \quad a(i, j) = 0 \text{ if } j \notin \{i, i+1\}$$

Since each word $w$ has a fixed number of states that it can be in at any step, we represent the transition function $a(\cdot,\cdot)$ as a row-normalized, banded-diagonal matrix $A^w$, where $A^w{}_{ij} = a(i,j)$. Note that nouns and adjectives are trivial one-state HMMs that effectively just consist of an emission model.

As we mentioned earlier, each word has some number of arguments, or objects it refers to. Thus, a word HMM observes as many detections in a frame as its word has arguments. For example, a noun or adjective has a single argument so the HMM encoding such a word would observe a single detection per frame, whereas a transitive verb like 'pickup' has two arguments -- the subject doing the action and the object being picked up -- and the HMM would observe the two corresponding detections at each frame.

The emission models, which score detections, take as input the VGG (Simonyan & Zisserman, 2014) features of the argument detections, as well as the average flow vector for each detection and the position of each detection center (relative to the first argument detection). Additionally, we include the Euclidean distance and shortest horizontal distance between each detection and the first argument detection, along with the relative flow vector between each detection and the first argument detection. The output of an emission model is a likelihood. Given detections $d = (d_1, d_2, \cdots, d_m)$ the emission score of word $w$ in state $p$ is of the form

$$b_w(p,d) = \sigma(\ \delta_{wp}\ +\ \sum_{i=1}^{m} c_{wpi} \cdot \varphi(d_1, d_i)\ )$$

where $\sigma$ is the logistic function, $c_{wpi}$ vectors of coefficients, $\delta_{wp}$ a bias factor, and $\varphi(d_i, d_j)$ the feature vector representation of detection $d_j$ (relative to detection $d_i$ when it comes to the position features). We define $\Phi(d)$ to be the concatenation of all $\varphi(d_1, d_i)$, and $c_{wp}$ to be the concatenation of all $c_{wpi}$. Then we have that $b_w(p,d) = \sigma(\ \delta_{wp}\ +\ c_{wp} \cdot \Phi(d)\ )$.

Then, given a sequence of states $y = (y_1, y_2, \cdots, y_n)$ and detections $x = (x_1, x_2, \cdots, x_n)$, with $x_i = (d^i{}_1, d^i{}_2, \cdots, d^i{}_m)$ for all $i$, the HMM for word $w$ gives the score

$$score_w(y,x) = \pi(y_1) \cdot \prod_{i=2}^{n} A^w{}_{y_{i-1}y_i} \cdot \prod_{i=1}^{n} b_w(y_i, x_i) \cdot \tau(y_n)$$

We see that the word HMM's score gives the joint probability of the state sequence satisfying the temporal dynamics of a word and that the detections depict the given progression of the word.

Now that we understand its building blocks we can finish describing the sentence tracker. The sentence tracker uses a sentence's dependency parse in order to combine the scores of the tracker and word HMMs. Given a track for the full sentence, each participant's track is scored by a tracker HMM and by the word HMM of each word for which the participant is an argument.

For example, take the sentence "the person approaches the red backpack." As shown on figure 3, we have two participants: "person" and "backpack". The track for participant R1 will be scored by a tracker HMM, the "person" HMM, and the "approach" HMM (as the subject argument), while the track for participant R2 will be scored by a tracker HMM, the "backpack" HMM, the "red" HMM, and the "approach" HMM (as the object argument).

More generally, say the sentence tracker encodes the sentence $s = (w_1, w_2, \cdots, w_L)$. The sentence $s$ has $R$ participants and, for each $l \in \{1, \cdots, L\}$, the word $w_l$ takes arguments $(r^l{}_1, r^l{}_2, \cdots, r^l{}_{m_l})$, where $m_l \leq R$ and each $r^l{}_i$ is a participant in $\{1, \cdots, R\}$. Given an $n$-frame video, let $T = \{t_1, t_2, \cdots, t_R\}$ be a track for the sentence, where $t_r$ is the track for participant $r$ with detections $(d^r{}_1, d^r{}_2, \cdots, d^r{}_n)$, one detection per frame. Let $Y = \{y^1, y^2, \cdots, y^L\}$ be the set of word-state sequences, where $y^l = (y^l{}_1, y^l{}_2, \cdots, y^l{}_n)$ is the state sequence for word $w_l$. Then the joint score given by the sentence tracker to the track $T$ and state sequences $Y$ is

$$score_{ST}(T,Y) = \prod_{r=1}^{R} score_{tracker}(t_r) \cdot \prod_{l=1}^{L} score_{w_l}(y^l, x^l)$$

where $x^l = (x^l{}_1, x^l{}_2, \cdots, x^l{}_n)$ is the sequence of observations for word $w_l$. Specifically, each observation $x^l{}_k = (d^{r^l{}_1}{}_k, d^{r^l{}_2}{}_k, \cdots, d^{r^l{}_{m_l}}{}_k)$ consists of the argument detections for word $w_l$ in frame $k$.

As described, scoring a sentence track $T$ and state sequences $Y$ can be factored into scoring the individual participant tracks with tracker HMMs and scoring the state sequences, along with the appropriate detections, with the corresponding word HMMs. We can, equivalently, think of the sentence tracker as an HMM itself. Specifically, given track $T$ and state sequences $Y$ we can derive a super-state sequence $S = (s_1, s_2, \cdots, s_n)$ such that for all frames $i$

$$s_i{}^{participant_r} = d^r{}_i \quad \forall r \in \{1, \cdots, R\}$$

$$s_i{}^{word_l} = y^l{}_i \quad \forall l \in \{1, \cdots, L\}$$

So each state $s_i$ in the state sequence $S$ encodes a detection for each participant in the sentence and a state for each word in the sentence. Then, the score for the state sequence $S$ is

$$score_{ST}(S) = \pi_{ST}(s_1) \cdot \prod_{i=2}^{n} a_{ST}(s_{i-1}, s_i) \cdot \prod_{i=1}^{n} b_{ST}(s_i) \cdot \tau_{ST}(s_n)$$

$$= score_{ST}(T,Y)$$

where

$$\pi_{ST}(s) = \prod_{l=1}^{L} \pi(s^{word_l})$$

$$\tau_{ST}(s) = \prod_{l=1}^{L} \tau(s^{word_l})$$

$$a_{ST}(s_i, s_j) = \prod_{r=1}^{R} a_{tracker}(s_i{}^{participant_r}, s_j{}^{participant_r}) \cdot \prod_{l=1}^{L} A^{w_l}{}_{s_i{}^{word_l}\, s_j{}^{word_l}}$$

$$b_{ST}(s) = \prod_{l=1}^{L} b_{w_l}( s^{word_l}, (s^{participant_{r^l 1}}, s^{participant_{r^l 2}}, \cdots, s^{participant_{r^l m_l}}) )$$

Thus, we can again use Viterbi to find the track $T^*$ and state sequences $Y^*$ that jointly maximize the tracker scores and word scores. Such a sentence track $T^*$ would be one that consists of a track for each participant in the sentence, where each participant's track is both temporally coherent and satisfies its relations to the words in the sentence, as derived from the sentence's dependency parse.

Behind the scenes, the sentence tracker builds a *lattice* of nodes, where there is a column of nodes for each frame. Each node represents one of the sentence tracker's super-states for that frame, encoding a detection for each participant and a state for each word, and a column is the enumeration of all possible nodes for that frame. Each node in a column is connected to every node in the next column. Effectively, a sequence of super-states $S = (s_1, s_2, \cdots, s_n)$ is equivalent to a path through the lattice that goes through exactly one node per column.

Performing Viterbi is equivalent to finding the highest-scoring path through the lattice. In the following sections we discuss how this lattice can be used to learn the parameters for the word HMMs, using the forward-backward algorithm and an adaptation of the Baum-Welch algorithm (Baum et al.,1970).

## 3.4 Pruning the Lattice

As presented above, the sentence tracker builds a lattice of the cross-product of all possible participant-to-detection and word-to-state mappings. The number of nodes in the lattice can grow very large, in particular when there are multiple participants in the sentence and many possible detections. For videos with resolutions around 256x192, we found that often up to 250

proposals per frame are needed to properly cover all objects of interest. So even for a two-participant sentence there are $> 250^2$ nodes per frame and each takes at least as many operations to compute.

Fortunately, the lattice is quite sparse. Many of the nodes have a probability of zero, and even among the non-zero nodes the probability mass is generally concentrated on very few nodes. We can thus use some heuristics to prune the lattice and improve our runtime by orders of magnitude.

The optimization we focused on was to pre-filter the object proposals for a video given a sentence.The main idea is that we often have knowledge of which proposals are likely to be relevant to a sentence and which are likely to be useless, so we can leverage this to pair down the proposals to a small relevant set. For our experiments with the system we are assuming pre-trained models for nouns, so we can use the nouns to prefilter the proposals. If we are given the sentence “the person approaches the red backpack”, we use the “person” and “backpack” models to choose the top K proposals per frame for each participant.

The simplest approach would be to take the K proposals with the best “person” score and the top K proposals with the best “backpack” score. However, we found that some small modifications can drastically improve the quality of the selected proposals. First, we let a proposal’s score be the product of the word score and a “mini-tracker” score. This tracker score is the max over the temporal coherence scores between the given proposal and all possible proposals from the previous frame, each weighted by its corresponding word score. This additional score helps select proposals that are more consistent from frame to frame and filter out proposals that are only sporadically prefered by the word models.

Once we have the top K proposals per participant per frame, we use forward-projection to fill in gaps between frames, and then apply non-maximal suppression (Blaschko, 2011) to

select the best, non-overlapping proposals. Forward-projection involves taking each of the top K proposals, projecting it into the next frame using its average optical flow, and adding the proposal closest to this projection to the set selected for that next frame. We ultimately use the top 4 proposals per participant: we take the 12 highest-scoring proposals and their forward-projections, and then narrow back down to 4 using non-maximal suppression. NMS is crucial when there are multiple instances of “person” or “backpack” in the the scene, since we can’t know a priori to which the sentence refers.

## 3.5 Learning Word HMM Parameters

Given a collection of videos, each paired with some positive and negative sentences describing events and actions in the video that are true and false, respectively, our objective is to learn the most discriminative parameters for the word HMMs. In theory, our approach should be able to learn all the words given the right examples, but we only look at the case where the nouns are known and attempt to learn verbs, adjectives, and prepositions. In order to learn the word parameters we use the expectation-maximization (EM) algorithm (Moon, 1996).

First we describe how EM works on HMMs, followed by a discriminative method for learning the HMM parameters given positive and negative examples. Finally, we present our algorithm as an extension of these methods.

### EM for HMM

Expectation-maximization (Moon, 1996) is an algorithm widely used for learning parameters of probabilistic models that have hidden variables, as is the case with HMMs.

If one has access to both observation sequences and the corresponding hidden state sequences, then it is trivial to learn the maximum likelihood parameters for an HMM. Namely,

the optimal value for the transition probability from state $u$ to state $v$ is simply the number of times that this transition occurs in the data divided by the number of times there is *any* transition from state $u$. Formally,

$$a(u,v) = \frac{count(u \to v)}{\sum_q count(u \to q)}$$

where $count(u \to v)$ is the number of times that the transition from state $u$ to state $v$ appears in the data.

Finding the optimal parameters for $b(u,x)$ is straightforward but depends on how the function $b$ is modeled. At a high level, we can simply aggregate all observations $\{x\}$ that are emitted by the hidden state $u$ in the data, and find the best parameters to fit that distribution.

Since the state sequences are generally not known, we can employ the Baum-Welch algorithm (Baum et al., 1970), the HMM-specific variant of EM, to simultaneously estimate the model parameters and the hidden state sequences. The main idea is to alternate between estimating what the hidden state sequences are and computing the optimal parameters using the estimated state sequences. Rather than selecting the best state sequence for each observation sequence, the algorithm implicitly computes a posterior probability distribution over all possible state sequences, and uses the candidate state sequences as weighted examples during the parameter estimation step. It has been shown that, with enough alternations between state estimation and parameter estimation, the algorithm is guaranteed to converge onto a locally optimal parameter set. We describe Baum-Welch below.

Suppose we have some HMM with parameters $\theta = \{a, b, \pi, \tau\}$ and states $\{1, 2, \cdots, N\}$. We are given $m$ observations sequences $X = (x^1 x^2, \cdots, x^m)$, where $x^i = (x^i{}_1, x^i{}_2, \cdots, x^i{}_n)$. We want to find the parameters $\theta$ that best model this data; specifically, we want the parameters that

assign the highest likelihood to the observation sequences. We first initialize the parameters, which is usually done randomly. We'll denote the initial parameters as $\theta^0$. The idea is that, given parameters $\theta^t$, we will compute the posterior probability over state sequences and then use these estimates to compute the new parameter set $\theta^{t+1}$. State sequence posteriors are computed in a phase called the *E-step*, while the parameters are re-estimated in the *M-step*. This alternation between E-step and M-step is repeated until the parameters converge or after some maximum number of iterations.

We illustrate this process by estimating the transition probabilities, but estimating the other parameters is analogous. As mentioned above, in order to estimate the transition probabilities we need the number of times that each state transition occurs in the data. Since we don't know the state sequences, we will estimate these counts during the E-step. The *expected* count for transition $u \to v$ is

$$count_E(u \to v) = \sum_{i=1}^{m} \sum_{y} p(y|x^i, \theta^{t-1}) \cdot count(y, u \to v)$$

where $y$ is a possible hidden state sequence for observation sequence $x^i$, $count(y, u \to v)$ is the number of times a transition $u \to v$ occurs in state sequence $y$, and $p(y|x^i, \theta^{t-1})$ is the posterior probability of states $y$ given the observations $x^i$ and the current parameters $\theta^{t-1}$.

Next, we simply use these expected counts as if they were hard counts, and find the maximum likelihood transition probabilities

$$a(u, v) = \frac{count_E(u \to v)}{\sum_{q=1}^{N} count_E(u \to q)}$$

This parameter re-estimation using the expected counts constitutes the M-step.

The challenge here is computing the sum over possible state sequences in the E-step:

$$\sum_{y} p(y|x^i, \theta^{t-1}) \cdot count(y, u \to v)$$

Since there are exponentially many potential state sequences it is simply intractable to actually enumerate them. Instead, we use the *forward-backward* algorithm, a dynamic programming algorithm for efficiently calculating the posterior distribution over states for two adjacent steps in a given observation sequence. Namely, forward-backward computes

$$p(y_j = u,\ y_{j+1} = v|x, \theta) \ = \sum_{y:\ y_j = u, y_{j+1} = v} p(y|x, \theta)$$

for any step $j$ and states $u, v$ given the observation sequence $x$ and parameters $\theta$. This is the marginal posterior probability that the state at step $j$ is $u$ and the state at the next step is $v$, conditioned on the observations and current parameters. Then we have the equivalence

$$\sum_{y} p(y|x^i, \theta^{t-1}) \cdot count(y, u \to v) \ = \ \sum_{j=1}^{n-1} p(y_j = u,\ y_{j+1} = v|x^i, \theta^{t-1})$$

We now present the forward-backward algorithm. Let $x = (x_1, x_2, \cdots, x_n)$ be an observation sequence. We define the *forward* probability function

$$\alpha_u(j) = p(x_1, x_2, \cdots, x_{j-1}, y_j = u|\theta)$$

for any step $j \in \{1, \cdots, n\}$ and any state $u$. This is the probability that the first $j - 1$ observations are $x_1, x_2, \cdots, x_{j-1}$ and that the state at the $j^{th}$ step is $u$, independent of the observation at step $j$.

Similarly, we define the *backward* probability function

$$\beta_u(j) = p(x_j, \cdots, x_n|y_j = u, \theta)$$

for all steps $j \in \{1, \cdots, n\}$ and states $u$. $\beta_u(j)$ represents the probability that the $j^{th}$ thru $n^{th}$ observations are $x_j, x_{j+1}, \cdots, x_n$ given that the state at step $j$ is $u$.

Note that given these two definitions we have the equality

$$p(y_j = u,\ y_{j+1} = v|x,\theta) = \frac{1}{Z}\alpha_u(j) \cdot a(u,v) \cdot b(u,x_j) \cdot \beta_v(j+1)$$

where $Z = p(x_1, x_2, \cdots, x_n|\theta) = \sum_q \alpha_q(i) \cdot \beta_q(i)$ for any $i \in \{1, \cdots, n\}$ is the total probability of the observation sequence given our parameters.

What remains now is computing $\alpha$ and $\beta$. As with the Viterbi algorithm, we define these functions recursively. For any state $u$ and all steps $j \in \{2, 3, \cdots, n\}$ we have that

$$\alpha_u(j) = \sum_v \alpha_u(j-1) \cdot a(v,u) \cdot b(v,x_j)$$

with the base case $\alpha_u(1) = \pi(u)$.

Similarly, for all states $u$ and steps $j \in \{1, 2, \cdots, n-1\}$ we have

$$\beta_u(j) = \sum_v a(u,v) \cdot b(u,x_j) \cdot \beta_v(j+1)$$

with the base case $\beta_u(n) = b(u,x_n) \cdot \tau(u)$. With a dynamic programming implementation both $\alpha$ and $\beta$ can be evaluated in polynomial time.

## Discriminative Training for HMM

As described so far, Baum-Welch allows us to learn the HMM parameters that best explain the observation sequences used during training. However, we are interested in learning HMMs that can discriminate observation sequences as positive or negative -- for instance, we want our 'pickup' HMM to give a high score to the tracks for a person picking up a backpack yet give a low score to tracks for a person *putting down* a backpack. In order to achieve this we use a discriminative variant of the Baum-Welch algorithm.

We base our approach on the discriminative training method introduced by Collins (2002), in which the correct state sequence for an HMM is compared to the maximum likelihood sequence as scored by the current parameter settings, and discrepancies between the

sequences are penalized. This method was designed for learning HMM parameters given observation sequences paired with the desired state sequences, and it works as follows. We will refer to correct state sequences included in the training set as the *positive* state sequences, and all competing state sequences as the *negative* sequences.

Let $X = \{x^1, x^2, \cdots, x^m\}$ be a set of observation sequences and $Y = \{y^1, y^2, \cdots, y^m\}$ the corresponding positive state sequences. Define $\Phi(x, y)$ to be a vector of features extracted from an observation sequence $x$ and state sequence $y$, and let $\alpha$ be a vector of coefficients -- the parameters we're trying to learn. A pair $x, y$ is assigned the score

$$\alpha \cdot \Phi(x, y) = \sum_{s=1}^{d} \alpha_s \cdot \Phi_s(x, y)$$

or the inner product of the feature vector and coefficients, where $d$ is the number of features and coefficients.

The algorithm makes $T$ passes over the dataset, and for each pair of observation sequence and state sequence $x, y$ it computes the highest-scoring sequence $\hat{y} = arg \max_{y'} \alpha \cdot \Phi(x, y)$. Any discrepancies between the feature vectors for $y$ and $\hat{y}$ are added to the coefficients vector $\alpha$. Namely, for each example $i \in \{1, 2, \cdots, m\}$ the algorithm performs the following update:

$$\forall s \in \{1, 2, \cdots, d\} \quad \alpha_s \leftarrow \alpha_s + \Phi_s(x^i, y^i) - \Phi_s(x^i, \hat{y}^i)$$

The effect of these updates is that the positive state sequence $y$ will tend to score at least as high as any competing negative state sequence $\hat{y}$. One of the main results from (Collins, 2002) is that if there exists some parameters that score the positive state sequences higher than all negative sequences, then the algorithm will converge on such parameters. They also show that in the case where the positive state sequences are not separable from some negative

sequences, the algorithm, with minor modifications, will converge on parameters that make the fewest mistakes.

In our setup, however, we don't have labeled tracks and state sequences for training our HMMs. All we have are videos and sentences describing them. More specifically, we have sentences that are true about the videos they describe -- which we refer to as *positive sentences* -- and we have sentences that are false about the videos they describe -- *negative sentences*. For example, given a video of a person picking up a red backpack we might have the positive sentences "the person picks up the backpack" and "the backpack is red"; we could also have negative sentences for this video, such as "the person puts down the backpack", "the backpack picks up the person", or "the backpack is blue".

The positive and negative sentences are useful because they allow us to generate correct tracks and competing tracks for a given word, so that we can use methods from (Collins, 2002). For instance, we know that if the sentence "the person puts down the backpack" is false about a video, then any track that our sentence tracker selects as portraying this sentence will be an incorrect track. On the other hand, tracks for "the person picks up the backpack" will not all be correct, but a correct track will be available for the sentence tracker to select. The idea is that the joint likelihood $score_{ST}(T, Y)$ for a track $T$ and any word state sequences $Y$ for a particular sentence should be high when $T$ portrays the sentence, but low then it doesn't.

To deal with the issue that even for positive sentences we don't know the correct track, we wrap the update rule from (Collins, 2002) in the Baum-Welch algorithm. In the following section we describe how we modify the E-step and M-step of Baum-Welch to incorporate positive and negative examples. We also describe how we factor the posterior probabilities computed by the forward-backward algorithm for the sentence tracker states into the marginal posteriors for the individual word HMM states.

## EM for Sentence Tracker

As stated above, we want to learn the most discriminative word HMM parameters given videos labeled with positive and negative sentences. In this section we describe how we derive posterior marginals for the word HMM states from the posterior marginals computed for the sentence tracker states, and how we incorporate the posteriors from positive and negative sentences to learn parameters in a discriminative manner.

At a high level, our algorithm is classical Baum-Welch applied to the sentence tracker. During the E-step we iterate over the video-sentence pairs and run the forward-backward algorithm, which computes the marginal posteriors for the sentence tracker states, or the nodes in the corresponding sentence tracker lattice. Namely, for every $j, u, v$, we compute the posterior probability that the node at frame $j$ is $u$ and the node at frame $j+1$ is $v$. However, instead of accumulating these into transition counts for the sentence tracker states, we marginalize the posterior at the level of the corresponding word HMM states.

Recall that lattice nodes encode participant-to-detection and word-to-state assignments. We use these lattice-level posteriors to find the expected counts for word-state transitions and assign soft word-state labels to observations. Specifically, given $n$-frame video $v$, sentence $s$ and parameters $\theta$, the expected count of transitions from state $p$ to state $q$ for word $w$, restricted to $v$ and $s$, are

$$count_{v,s}(w, p \to q) = \sum_{i \in S_s(w)} \sum_{j=1}^{n-1} p(y_j = u, y_{j+1} = u' | v, s, \theta)$$

where $S_s(w)$ is the set of indices $i$ such that $w$ is the $i^{th}$ word in the sentence $s$, and $u$ and $u'$ are lattice nodes such that $u^{word_i} = p$ and $u'^{word_i} = q$. Recall that $p(y_j = u, y_{j+1} = u' | v, s, \theta)$, the

probability that the $j^{th}$ node in the lattice path is $u$ and the next node is $u'$, can be computed by the forward-backward algorithm.

Similarly, we want to generate weighted observation examples for the emission models. An example for state $p$ of an $m$-argument word $w$ would be the feature vector $\Phi(d)$ that we defined earlier for detections $d = (d_1, \cdots, d_m)$ from frame $k$ of video $v$. Then the weight of this example for state $p$ of word $w$ would be

$$weight_{v,s}(w,p,d,k) = \sum_{i \in S(w)} \sum_{u} p(y_k = u | v, s, \theta)$$

where $u$ is a lattice node from frame $k$ such that:

- $u^{word_i} = p$
- $\forall i \in \{1, \cdots, m\}$, $u^{participant_{r_i}} = d_i$, where $r_i$ is the participant in the $i^{th}$ argument position of word $w$

Note that we can compute $p(y_k = u | v, s, \theta)$ with little additional work once we have the two-state posteriors, since $p(y_k = u | v, s, \theta) = \sum_{u'} p(y_k = u, y_{k+1} = u' | v, s, \theta) = \sum_{u'} p(y_{k-1} = u', y_k = u | v, s, \theta)$.

We then aggregate these counts and examples across all the video-sentence pairs. As in (Collins, 2002), when we aggregate the word-state transition counts, for each word, we subtract the counts from negative sentences from the counts from positive sentences. Namely, let the final counts for word $w$ from state $p$ to state $q$ be

$$count_{net}(w, p \to q) = \sum_{(v,s) \in S^P} count_{v,s}(w, p \to q) - \sum_{(v,s) \in S^N} count_{v,s}(w, p \to q)$$

where $S^P$ and $S^N$ are the sets of positive and negative video-sentence pairs, respectively.

When aggregating the weighted observation examples, we simply label examples from positive sentences as positive examples and examples from negative sentences as negative examples.

During the M-step we use the counts and weighted examples to re-estimate the word HMM parameters. When estimating the state transition matrix $A^w$ for word $w$, we do two phases of counts normalization. First, we normalize each row with a small smoothing term $\alpha$ for numerical stability:

$$A^w{}_{p,q} = \frac{count_{net}(w, p \rightarrow q) + \alpha}{N \cdot \alpha + \sum_h count_{net}(w, p \rightarrow h)}$$

where $N$ is the number of states of $w$. Note that after this normalization we violate our banded-diagonal matrix constraint which states that $A^w{}_{p,q} = 0 \;\; if \; q \notin \{p, p+1\}$ . We now enforce our constraint by zeroing all entries $A^w{}_{p,q}$ where $q \in \{p, p+1\}$ and then renormalize without a smoothing term. This two-step estimation is a form of coordinate descent (Breheny & Huang, 2011), where our parameters start on the manifold of valid parameters, make a small step along the strongest gradient direction, and then reproject onto the manifold of valid parameters.

As for the emission model parameters, for each state we do stochastic adaptive gradient descent (Duchi et al. 2011) using all the corresponding positive and negative examples, each with its computed weight. For each EM iteration $i$ we do $min(8, \; i + 2)$ epochs of gradient descent. The reasoning behind this is that at each iteration we become progressively more confident about which are the positive and negative examples, but we never want to overfit to the current distribution over the examples. We avoid performing too many gradient epochs in the first iterations and progressively add iterations as the tracks become more and more discriminative, concentrating more weight on the positive examples and spreading probability mass over the negative examples.

# 4 Experiments

We want to show our system can learn words that are functions of appearance, spatial relations, motion, and patterns of these features over time. In order to test this we attempted to learn words that require one or more of these qualities and tried to disambiguate scenes using the learned words. We attempted to learn six pairs of words:

- pickup / put down
- approach / leave
- open / close (a book)
- walking / skipping
- moving / still
- red / blue

We now describe the experimental setup, in terms of training examples and initialization parameters used, and report the results.

## 4.1 Setup

We trained each pair of words separately from the other pairs. Additionally, we trained ‘red’, ‘blue’, ‘pickup’, ‘put down’, and ‘approach’ jointly. All nouns used in the training and testing sentences for these experiments were pre-trained and were frozen during the experiments.

The six transitive verbs were trained on sentences like “the person picks_up the backpack”. We ignore determiners and reduce transitive verbs to their infinitive, so the above sentence would actually be presented to the system as “person pickup backpack”. Both positive and negative sentences had this form. Some example video-sentence pairs for these verbs are shown in figure 4. We set the number of states for these word HMMs to $N = 4$.

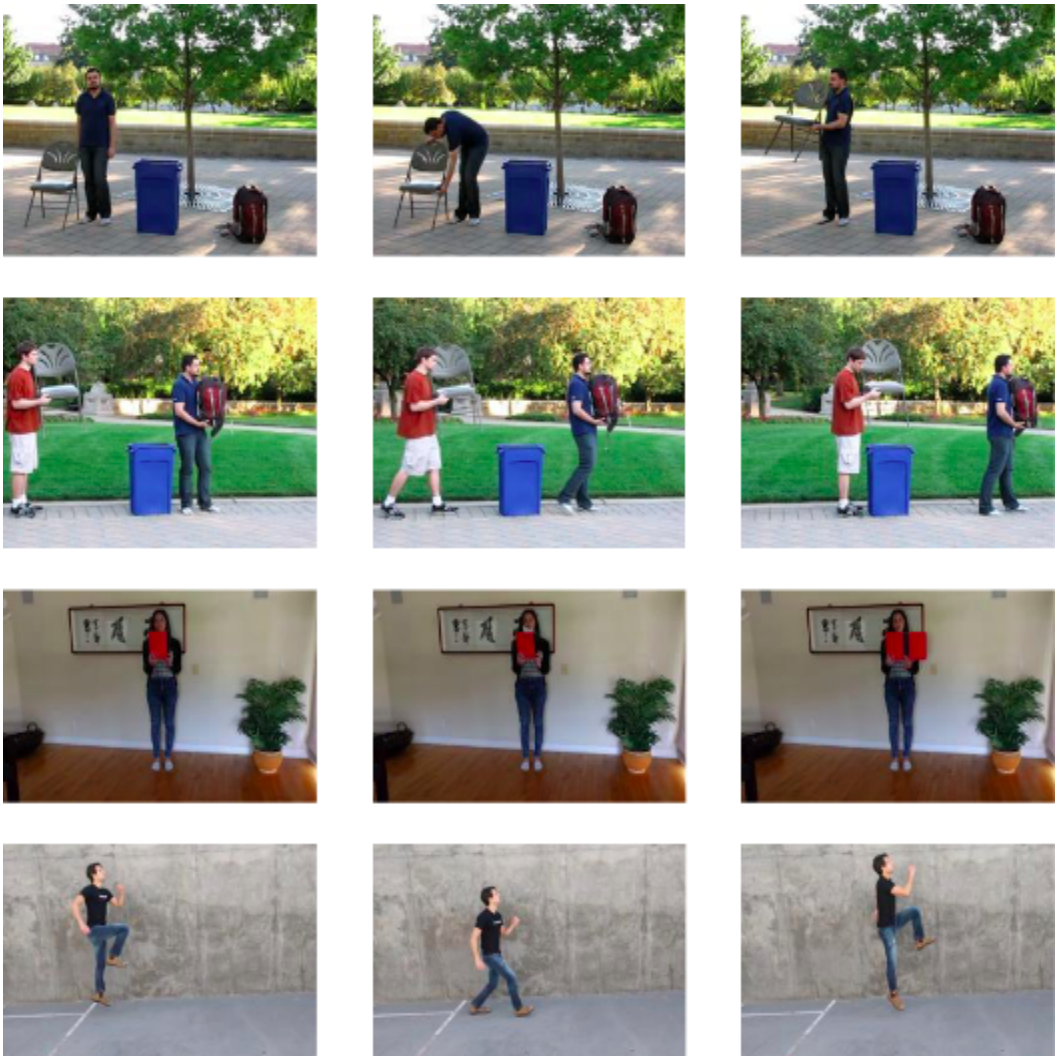


Figure 4. Each row of images depicts frames from a sample video that we paired with sentences during training. For instance, the top row video might have been paired with the positive (true) sentence “person pickup chair” or the negative (false) sentences “person pickup backpack” and “person put_down chair”. The second video from the top might be paired with positive sentences like “red person approach bin”, “person leave bin”, “blue bin” or “moving person”, and with negative sentences like “backpack approach bin” or “still person”. The third video from the top was used for learning ‘open’ and ‘close’, using the positive sentence “person open book” and negative sentence “person close book”. The bottom video depicts “skipping person”.

As for the six adjectives, we use sentences of the form “skipping person” or “blue bin”. Additionally, for learning ‘red’ and ‘blue’ we used sentences of the form ‘red person pickup blue bin’ using a pre-trained, working model for ‘pickup’. Examples are shown in figure 4. For ‘walking’, ‘skipping’, and ‘moving’ we used $N = 3$ since these words reflect dynamic events that often evolve over time, whereas for ‘red’, ‘blue’ and ‘still’ we set $N = 1$.

In addition to the experiments described above, we also attempted to jointly learn ‘red’, ‘blue’, ‘pickup’, ‘put down’ and ‘approach’. This simply involved combining the positive sentences and negative sentences of all five words into joint positive and negative sentence sets, respectively. Moreover, the positive sentences for the transitive verbs were over-specified by adding ‘red’ and ‘blue’ to the appropriate nouns. Figure 4 depicts some example video-sentence pairs.

For all words $w$ we initialized the state transition probabilities to $A^w_{i,i} = 0.99$ and $A^w_{i,\,i+1} = 0.01$ for all $i$. Moreover, the coefficients for the observation emission models were all initialized independently at random by sampling from a zero mean normal with small ($< 1$) variance.

## 4.2 Results

We now present the test cases used for each experiment and the corresponding results.

We tested ‘pickup’ by showing the system a video portraying two people, one picking up a chair and on picking up a backpack, and asking the system to give the best track for “person pickup chair” and the best track for “person pickup backpack”. The desired outcome is that the person track for each sentence is the one for the individual interacting with the correct object. ‘Pickup’ correctly disambiguated the the two test videos used.

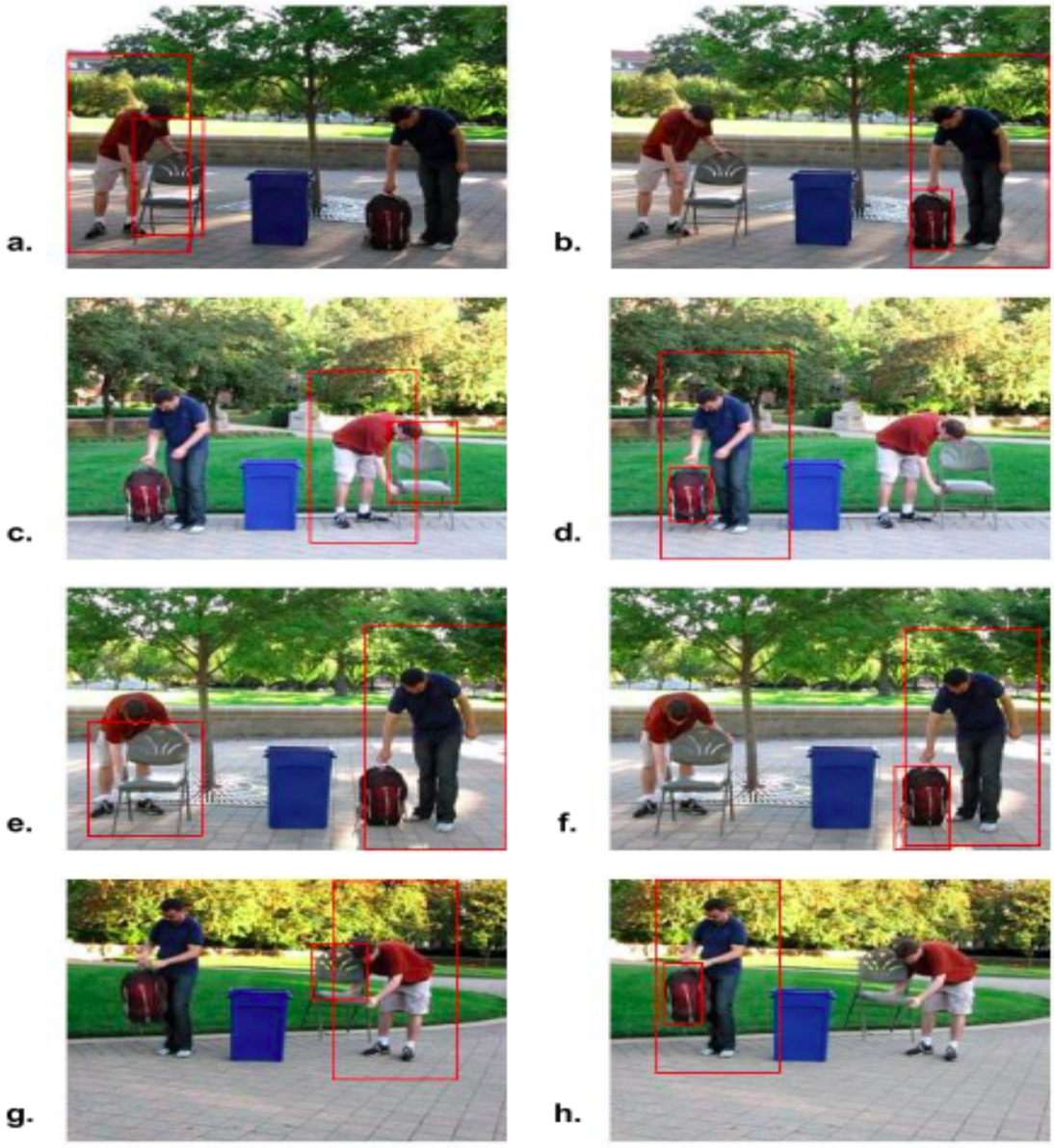



Figure 5. Shows the tracks selected for the four ‘pickup’ test cases and the four ‘put down’ test cases. **a.** “person pickup chair” **b.** “person pickup backpack” **c.** “person pickup chair” **d.** “person pickup backpack” **e.** “person put_down chair” **f.** “person put_down backpack” **g.** “person put_down chair” **h.** “person put_down backpack”.

We tested ‘put down’ with two videos analogous to those used for ‘pickup’. The learned word model managed to correctly disambiguate one of the two videos but not the other. Figure 5 shows the tracks produced for the ‘pickup’ and ‘put down’ tests.

In order to test ‘approach’ and ‘leave’ we presented the system with a video portraying two people and a bin, where one person approaches the bin while the other walks away from it. We asked the system for the best track for “person approach bin” and the best track for “person leave bin”. As shown in figure 6, both words correctly disambiguated the scene.

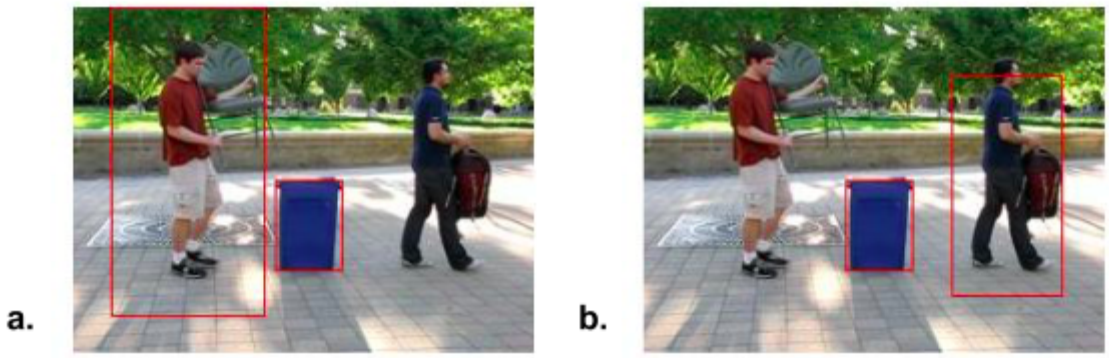



Figure 6. Shows the tracks selected for the ‘approach’ versus ‘leave’ test case.

**a.** “person approach bin” **b.** “person leave bin”.

We tested ‘open’ and ‘close’ with videos showing two people, one opening a book while the other closes a book. The learned models for ‘open’ and ‘close’ failed to disambiguate the videos. The results are shown in figure 7.

We used analogous test cases used for the six adjectives. For ‘walking’ versus ‘skipping’ we presented the system with 8 videos containing two people, one walking and the other skipping. The model for ‘walking’ correctly disambiguated 7 out of 8 test videos, while the model for ‘skipping’ correctly disambiguated 5 out of the 8 videos (see figure 8).

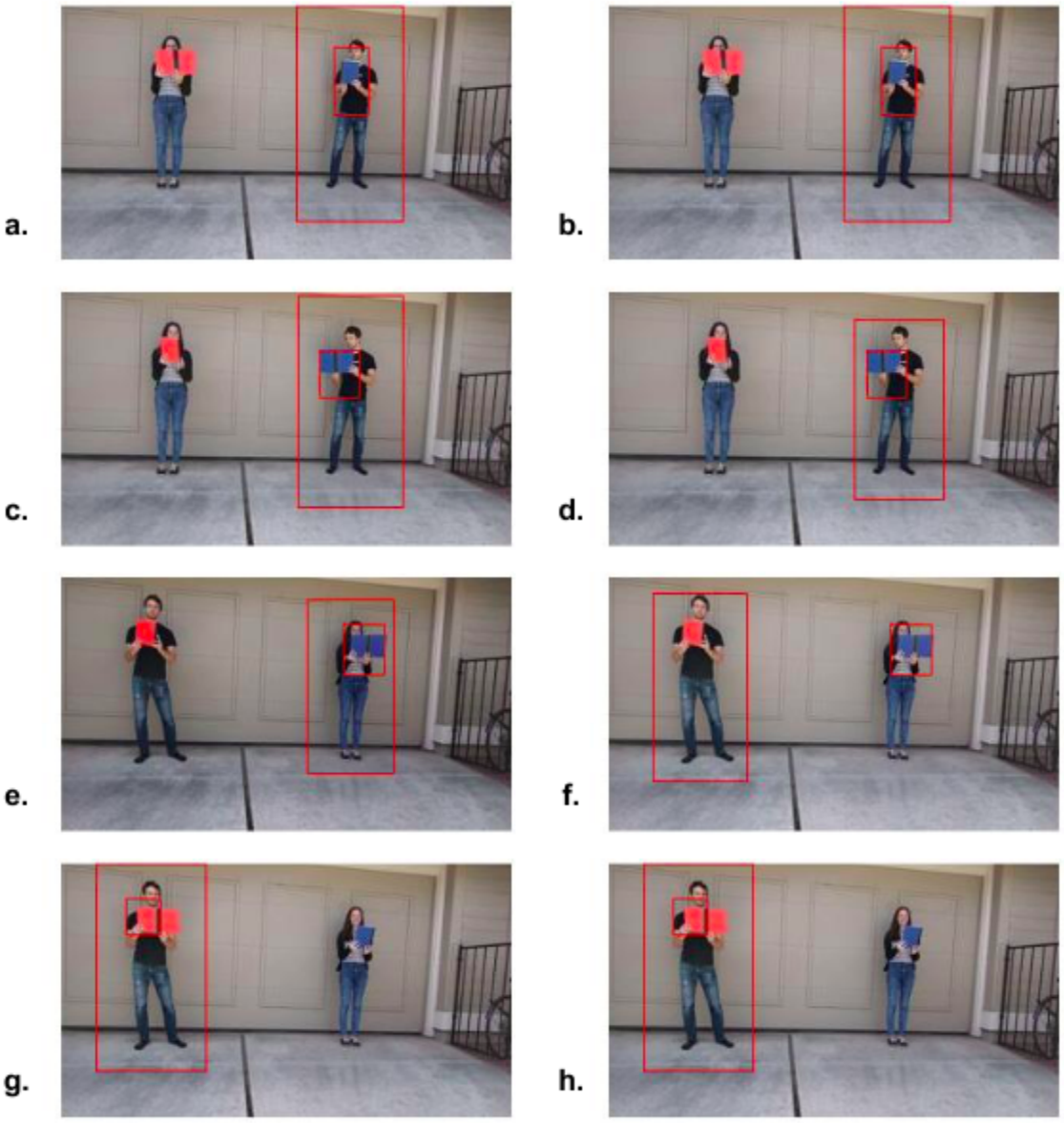
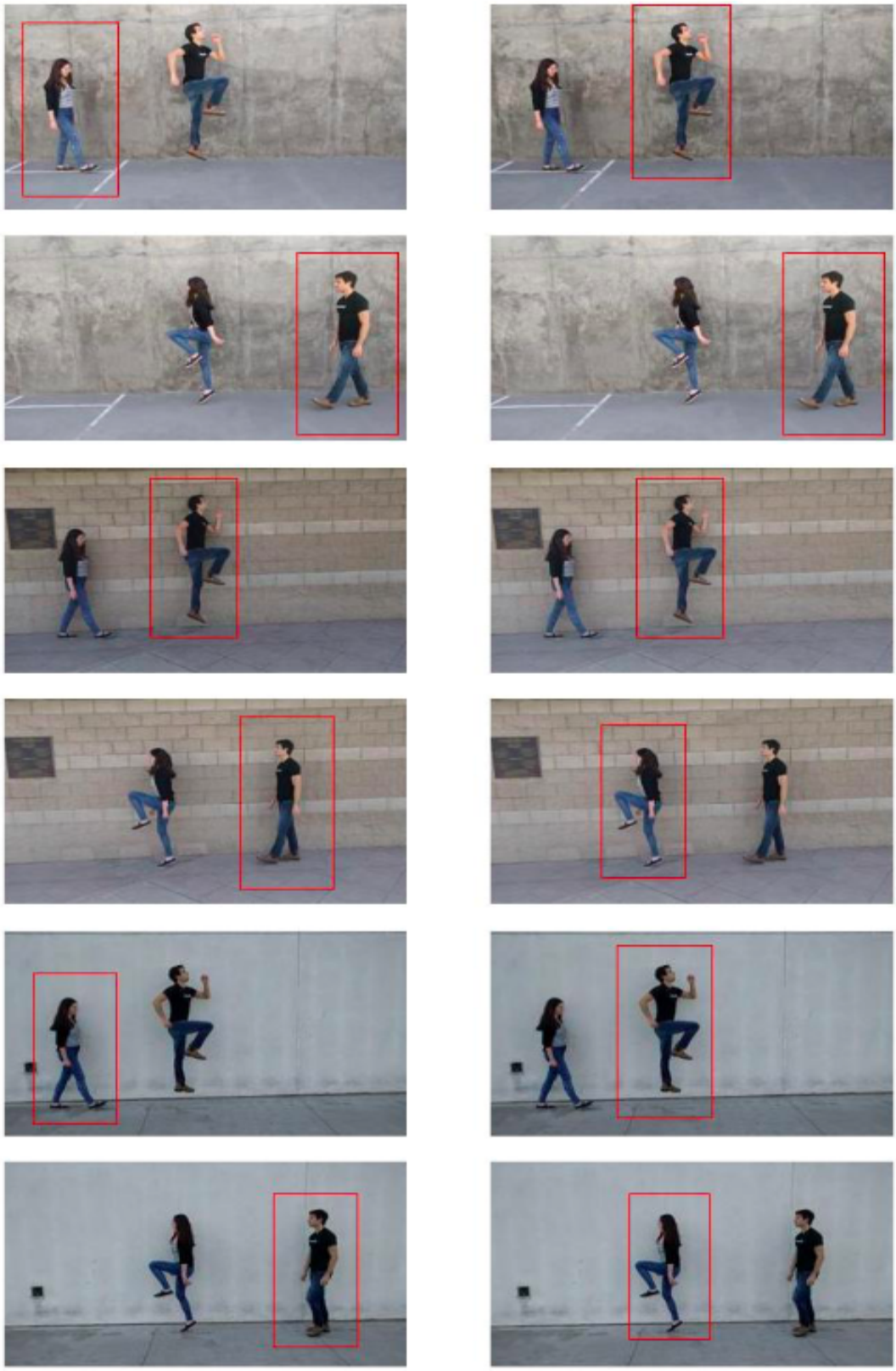



Figure 7. Shows the tracks selected for the ‘open’ versus ‘close’ test cases. The first and fourth videos depict the person on the left opening a book and the person on the right closing a book. The second and third videos depict the person on the left closing a book and the person on the right opening a book. The left column of images (**a**, **c**, **e**, **g**) shows the tracks selected for “person open book”, while the right column (**b**, **d**, **f**, **h**) shows the tracks selected for “person close book”.



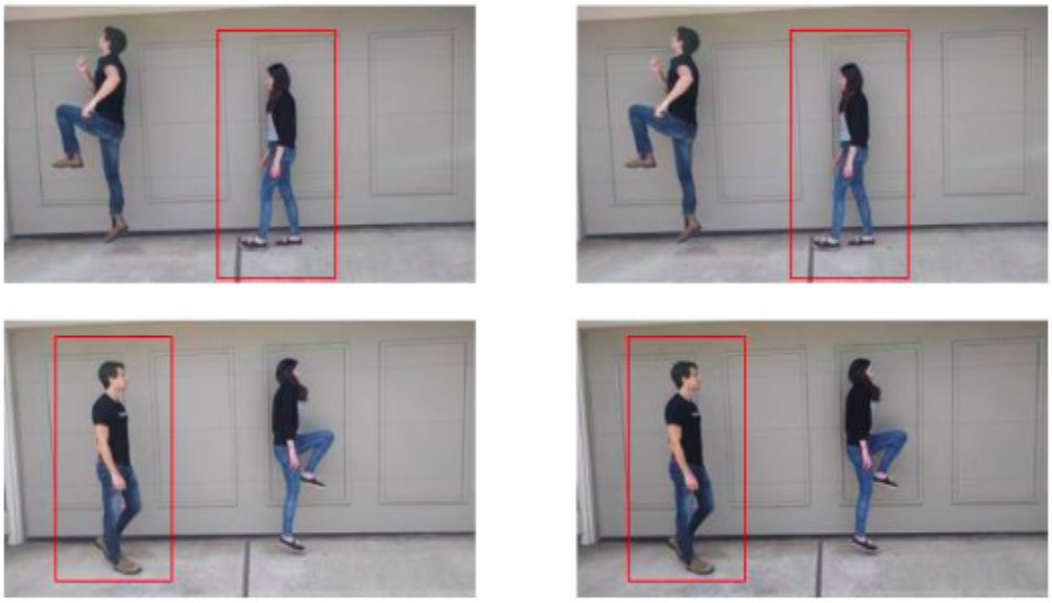


Figure 8. Shows the tracks selected for the ‘walking’ versus ‘skipping’ test cases. The eight images in the left column show the tracks for “walking person”, and the eight images in the right column show the tracks for “skipping person”.

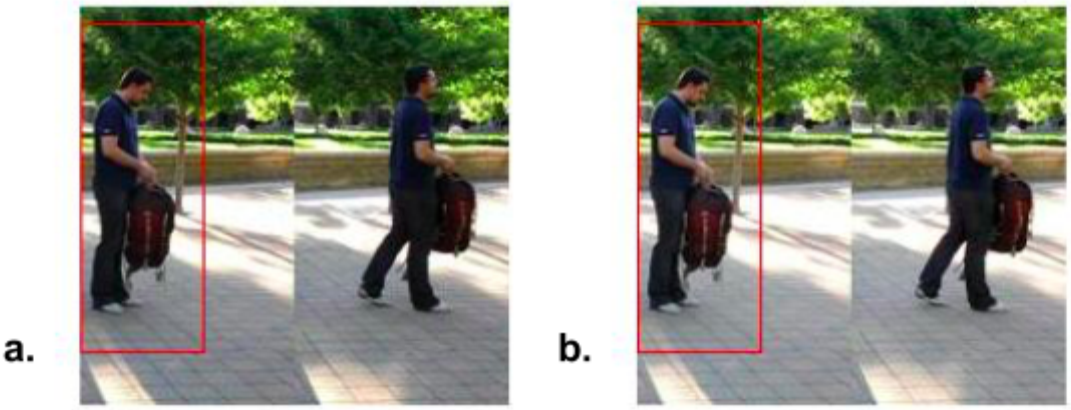



Figure 9. Shows the tracks selected for the ‘moving’ versus ‘still’ test case. **a.** Shows the track selected for “moving person” and **b.** shows the track for “still person”.

For testing ‘moving’ versus ‘still’ we created an artificial video by taking a video of a person walking across the scene, snapshotting the first frame, and stitching this frame to the original video such that the new video depicts two identical individuals, one frozen still throughout the video and the other walking. The track for “still person” should select the motionless individual while the track for “moving person” should select the individual walking. However, as shown in figure 9, both models selected the motionless person track.

In order to test ‘red’ and ‘blue’ we used videos showing a two instances of a given object class, where one instance is blue and the other is red. Specifically, we used to videos each portraying a person wearing a blue shirt and a person wearing a red shirt, as well as two videos with a blue book and a red book. In the two videos with people, when we asked for the “red person” and “blue person” tracks the system correctly disambiguated the videos. However, the models were unable to disambiguate between the blue and red books. It is important to note that the ‘red’ and ‘blue’ models were trained on video-sentence pairs that included the individuals present in the test cases, but no books were observed during training. This suggests that the models did learn some function of appearance that was able to disambiguate between the blue- and red-wearing people, but this function did not necessarily map to the colors red and blue and was therefore unable to generalize to the unseen object class. We imagine that the models would generalize better if trained on a more diverse set of object classes. The results for these tests are shown in figure 10.

Finally, we report the results for the joint learning of ‘red’, ‘blue’, ‘pickup’, ‘put down’ and ‘approach’. The models for the individual words were tested identically to the cases where trained separately, and the results were nearly identical as well. The ‘blue’, ‘pickup’, ‘put down’ and ‘approach’ models performed exactly as they did when trained separately, but the ‘red’ model failed to identify the correct person in one of the two test cases.

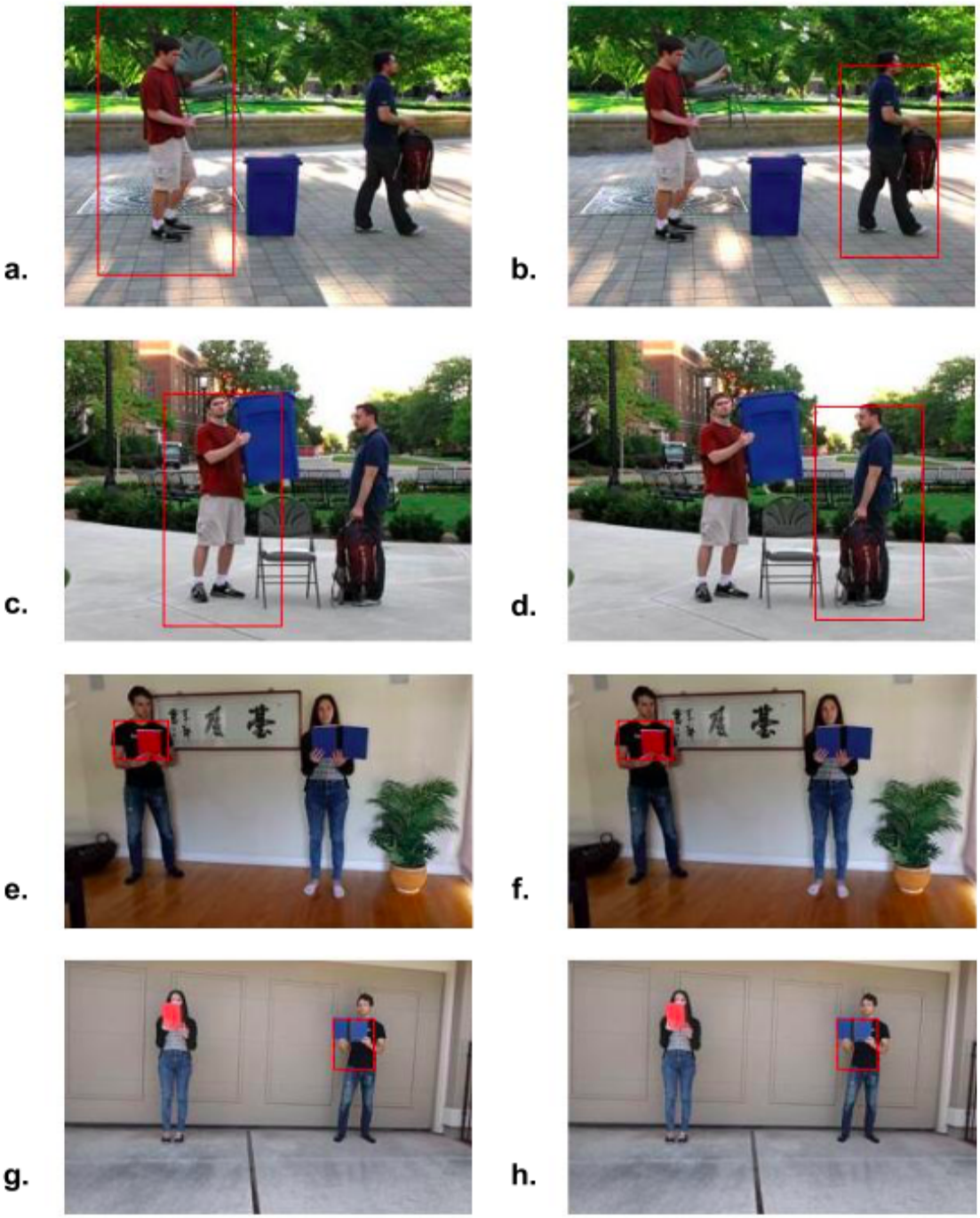



Figure 10. Shows the tracks selected for the ‘red’ versus ‘blue’ test cases. **a.** “red person” **b.** “blue person” **c.** “red person” **d.** “blue person” **e.** “red book” **f.** “blue book” **g.** “red book” **h.** “blue book”.

Overall, the results show that the system can effectively leverage appearance, spatial relations, and temporal dynamics when learning words. The ‘pickup’ model, for instance, shows that the models require for the subject and object to be in contact. In fact, we also trained this model while omitting the VGG features and it performed identically during testing. The color models necessarily rely on appearance features to disambiguate between objects, and the learned distinction between ‘approach’ and ‘leave’ shows that the models can leverage the timeline of an event -- ‘approach’ and ‘leave’ differ primarily in the fact that one consists of starting far and ending close, while the other starts close and ends far. Moreover, the five-word experiment shows that our method still works when training sentences contain multiple unknown words. In fact, we believe that our approach can work even when the nouns are unknown, learning all words simultaneously.

At the same time the system is not without shortcomings and has substantial room for improvement. From our experience it seems that the models are quite sensitive to the positive and negative examples used. For example, the videos used to learn ‘pickup’, ‘put down’, ‘approach’, ‘leave’, ‘red’ and ‘blue’ were biased towards showing the person wearing blue more often than the person wearing red. As a results, we initially found all models selecting the blue person in test cases regardless of the sentence being tracked. We realized that part of the issue was that the models would overfit to the appearance features, since these vastly outnumbered the non-appearance features. We resolved this issue by scaling down the appearance features relative to the non-appearance features, but the issue is worth noting.

Additionally, while the uniform feature representation we utilized proved to be expressive enough to learn many of the concepts we tested, it cannot completely characterize many complex events. For instance, when initially trying to learn ‘approach’ we provided negative examples such as “bin approach person” (paired with a video of a person approaching a bin) to

indicate that the relation is not necessarily symmetric. However, the model was unable to learn a useful function with such examples. It is not to say that it's impossible for the features to represent such an event, but it warns against over-specifying.

In summary, the approach we present shows promise. We demonstrated its ability to learn words that capture a variety of useful and interesting properties, and, more importantly, we showed it can be done with little supervision.

# 5 Future Work

This work has shown the feasibility of our approach, but there is much more to be done in demonstrating the extent of what can be learned as well as optimizing performance.

Much like convolutional neural networks such as VGG (Simonyan & Zisserman, 2014) and OverFeat (Sermanet et al., 2013) provide a building block for more easily learning linear classifiers for new nouns, we think a universal network for learning verbs can help with learning new verbs beyond those we experiment with in this research. We imagine there are features that are strong cues for many two-argument words from verbs to prepositions -- for example, whether the two objects are in contact is crucial for a huge number of transitive verbs. A network that extracted these salient features might make learning linear classifiers for such words significantly easier.

For future work with our system, we envision a network a few layers deep that takes as input the VGG, optical flow, and position features for both arguments of a word, and the linear classifier for a word plugs in as a final layer on top of this network. The network would be shared by all two-argument words, and as such it would learn a feature representation most useful for learning this class of words. Learning the parameters for this network would work seamlessly with our current learning algorithm, since it is just as easy to perform the gradient descent through end-to-end, through the linear classifiers and the entire network.

While we only experimented with learning words assuming pre-trained models for the nouns in the training sentences, we designed our system hoping it could ultimately be used to learn the nouns as well. The reason for not attempting this was the computational complexity of our core algorithms, since without knowing the nouns we would be unable to effectively prune the

sentence tracker lattice. However, we think there are promising avenues for bypassing this obstacle.

One potential approach would be to divide the learning into two phases: first learning all the relevant nouns, and only then learning the remaining words as we demonstrate in this paper. With enough diversity in the video corpus, particularly a variety of backgrounds, it is not unreasonable to consider that the co-occurrence of nouns in sentences with their corresponding image patches in the videos would be enough to learn many noun models, without the need to leverage the other words in the sentences.

Another approach would be to learn all the words together, but to start with very low-resolution versions of the videos and iteratively work back up to full resolution. The idea is that with small enough frames we can roughly capture all the relevant objects with a sufficiently small number of object proposals such that we can avoid pruning the sentence tracker lattice. Then, given the noun models learned at low resolution, we can prune the lattice for a slightly higher resolution. We can re-learn all the words at the higher resolution, refining our noun models, and continue with this iterative process until we reach the original resolution. In this approach we can leverage the constraints imposed by sentence structure to learn the nouns, as we do with the other words. The main issue is that there will be nouns that are simply too hard to discern at low resolutions.

As for the system's performance, there are many hyperparameters and structural parameters that can be tuned, and we simply didn't have the bandwidth to empirically find the best (or even *good*) settings for many of them. Here we outline some of the salient configuration elements that could be explored.

While our learning algorithm solves for optimal transition probabilities for the word HMM states, we actually experimented with freezing the transition probabilities at their values from initialization. We did not witness evidence for whether or not this makes a significant difference, one way or the other, during training or evaluation. However, there are likely words for which this choice is consequential, so we think it might be worth exploring along with other choices of initialization.

Similarly, we made a design decision for learning the emission models. Specifically, at each EM iteration we perform gradient descent starting with the final parameters from the previous iteration. We could have, alternatively, started gradient descent for each iteration with zeroed parameters.

In terms of hyperparameters, we know that the scaling of the VGG features with respect to the non-VGG features is important. When we began experimenting we found that the system would often overfit to the appearance of events at the cost of spatial relations, preventing it from learning transitive verbs like ‘pickup’ or ‘approach’. We hypothesized that the reason for this was that the VGG features, corresponding to appearance, outnumbered the non-VGG features 1000 to 1. Scaling the features to offset this fixed the issue, but we didn’t experiment with many scalings.

The weight given to the tracker HMM score, the learning rate used during gradient descent, and the number of gradient epochs all impact the system’s performance and deserve more careful tuning.

We think our approach shows promise based on the experiments performed to date. However, there is much work to be done in order to test its limits, and there is quite a bit of room for

optimization that we simply did not have the time to do. We are excited to see how far this can go.

# Acknowledgments

I would like to thank Andrei Barbu, who helped supervise this project. His guidance was instrumental in getting over the many roadblocks we hit along the way.

# References

[1] Alexe, Bogdan, Thomas Deselaers, and Vittorio Ferrari. "Measuring the objectness of image windows." *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 34.11 (2012): 2189-2202.

[2] Ali, Saad, and Mubarak Shah. "Human action recognition in videos using kinematic features and multiple instance learning." *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 32.2 (2010): 288-303.

[3] Altun, Yasemin, Ioannis Tsochantaridis, and Thomas Hofmann. "Hidden markov support vector machines." *ICML*. Vol. 3. 2003.

[4] Baker, Mark C. "Thematic roles and syntactic structure." *Elements of grammar*. Springer Netherlands, 1997. 73-137.

[5] Baum, Leonard E., et al. "A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains." *The annals of mathematical statistics* 41.1 (1970): 164-171.

[6] Bilen, Hakan, Marco Pedersoli, and Tinne Tuytelaars. "Weakly supervised object detection with posterior regularization." *British Machine Vision Conference*. 2014.

[7] Blaschko, Matthew B. "Branch and bound strategies for non-maximal suppression in object detection." *Energy Minimization Methods in Computer Vision and Pattern Recognition*. Springer Berlin Heidelberg, 2011.

[8] Breheny, Patrick, and Jian Huang. "Coordinate descent algorithms for nonconvex penalized regression, with applications to biological feature selection." *The annals of applied statistics* 5.1 (2011): 232.

[9] Chavali, Neelima, et al. "Object-Proposal Evaluation Protocol is' Gameable'."*arXiv preprint arXiv:1505.05836* (2015).

[10] Collins, Michael. "Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms." *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*. Association for Computational Linguistics, 2002.

[11] Crandall, David J., and Daniel P. Huttenlocher. "Weakly supervised learning of part-based spatial models for visual object recognition." *Computer Vision–ECCV 2006*. Springer Berlin Heidelberg, 2006. 16-29.

[12] Dalal, Navneet, and Bill Triggs. "Histograms of oriented gradients for human detection." *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*. Vol. 1. IEEE, 2005.

[13] Duchi, John, Elad Hazan, and Yoram Singer. "Adaptive subgradient methods for online learning and stochastic optimization." *The Journal of Machine Learning Research* 12 (2011): 2121-2159.

[14] Everingham, Mark, et al. "The pascal visual object classes challenge: A retrospective." *International Journal of Computer Vision* 111.1 (2015): 98-136.

[15] Ganapathiraju, Aravind, Jonathan Hamaker, and Joseph Picone. "Hybrid SVM/HMM architectures for speech recognition." *INTERSPEECH*. 2000.

[16] Horn, Berthold K., and Brian G. Schunck. "Determining optical flow." *1981 Technical symposium east*. International Society for Optics and Photonics, 1981.

[17] Jackendoff, Ray. "The status of thematic relations in linguistic theory."*Linguistic inquiry* 18.3 (1987): 369-411.

[18] Liu, Ce. *Beyond pixels: exploring new representations and applications for motion analysis*. Diss. Massachusetts Institute of Technology, 2009.

[19] Manen, Santiago, Matthieu Guillaumin, and Luc Gool. "Prime object proposals with randomized prim's algorithm." *Proceedings of the IEEE International Conference on Computer Vision*. 2013.

[20] Mel'čuk, Igor' Aleksandrovič. *Dependency syntax: theory and practice*. SUNY press, 1988.

[21] Moon, Tood K. "The expectation-maximization algorithm." *Signal processing magazine, IEEE* 13.6 (1996): 47-60.

[22] Nguyen, Minh Hoai, et al. "Weakly supervised discriminative localization and classification: a joint learning process." *Computer Vision, 2009 IEEE 12th International Conference on*. IEEE, 2009.

[23] Prest, Alessandro, et al. "Learning object class detectors from weakly annotated video." *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE, 2012.

[24] Prim, Robert Clay. "Shortest connection networks and some generalizations."*Bell system technical journal* 36.6 (1957): 1389-1401.

[25] Rabiner, Lawrence R., and Biing-Hwang Juang. "An introduction to hidden Markov models." *ASSP Magazine, IEEE* 3.1 (1986): 4-16.

[26] Rosenblatt, Frank. "The perceptron: a probabilistic model for information storage and organization in the brain." *Psychological review* 65.6 (1958): 386.

[27] Russakovsky, Olga, et al. "Imagenet large scale visual recognition challenge." *International Journal of Computer Vision* 115.3 (2015): 211-252.

[28] Sermanet, Pierre, et al. "Overfeat: Integrated recognition, localization and detection using convolutional networks." *arXiv preprint arXiv:1312.6229* (2013).

[29] Siddharth, Narayanaswamy, Andrei Barbu, and Jeffrey Mark Siskind. "Seeing What You're Told: Sentence-Guided Activity Recognition in Video." *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*. IEEE, 2014.

[30] Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." *arXiv preprint arXiv:1409.1556* (2014).

[31] Thurau, Christian, and Václav Hlaváč. "Pose primitive based human action recognition in videos or still images." *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*. IEEE, 2008.

[32] Valstar, Michel F., and Maja Pantic. "Combined support vector machines and hidden markov models for modeling facial action temporal dynamics." *Human–Computer Interaction*. Springer Berlin Heidelberg, 2007. 118-127.

[33] Viterbi, Andrew J. "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm." *Information Theory, IEEE Transactions on* 13.2 (1967): 260-269.

[34] Wang, Chong, et al. "Weakly supervised object localization with latent category learning." *Computer Vision–ECCV 2014*. Springer International Publishing, 2014. 431-445.