Learning from Narrated Instruction Videos

Jean-Baptiste Alayrac, Piotr Bojanowski, Nishant Agrawal, Josef Sivic, Ivan Laptev, and Simon Lacoste-Julien

Abstract—Automatic assistants could guide a person or a robot in performing new tasks, such as changing a car tire or repotting a plant. Creating such assistants, however, is non-trivial and requires understanding of visual and verbal content of a video. Towards this goal, we here address the problem of automatically learning the main steps of a task from a set of narrated instruction videos. We develop a new unsupervised learning approach that takes advantage of the complementary nature of the input video and the associated narration. The method sequentially clusters textual and visual representations of a task, where the two clustering problems are linked by joint constraints to obtain a single coherent sequence of steps in both modalities. To evaluate our method, we collect and annotate a new challenging dataset of real-world instruction videos from the Internet. The dataset contains videos for five different tasks with complex interactions between people and objects, captured in a variety of indoor and outdoor settings. We experimentally demonstrate that the proposed method can automatically discover, learn and localize the main steps of a task in input videos.

Index Terms—Step discovery, Narrated instruction videos, unsupervised learning.

1 Introduction

Millions of people watch narrated instruction videos[1] to learn new tasks such as assembling IKEA furniture or changing a flat car tire. Many of such tasks have large amounts of videos available on-line. For example, querying for "how to change a tire" results in more than 300,000 hits on YouTube. Most of these videos, however, are made with the intention to teach other people to perform the task and do not provide direct supervisory signal for automatic learning algorithms. Developing unsupervised methods that could learn tasks from myriads of instruction videos on the Internet is therefore a key challenge. Such automatic cognitive ability would enable constructing virtual assistants and smart robots that learn new skills from the Internet to, for example, help people achieve new tasks in unfamiliar situations.

In this work, we consider instruction videos and develop a method that learns a sequence of steps, as well as their textual and visual representations, required to achieve a certain task. For example, given a set of narrated instruction videos demonstrating how to change a car tire, our method automatically discovers consecutive steps for this task such as *loosen the nuts of the wheel*, *jack up the car*, *remove the spare tire* and so on as illustrated in Figure 1. In addition, the method learns the visual and linguistic variability of these steps from natural videos.

Discovering key steps from instruction videos is a highly challenging task. First, linguistic expressions for the same step can have high variability across videos, for example: "...Loosen up the wheel nut just a little before you start jacking the car..." and "...Start to loosen the lug nuts just enough to make them easy to turn by hand...". Second, the visual appearance of each step may vary greatly because of differences in viewpoints, lighting, the poses, clothing and motion of people, types of manipulated objects and other factors. Finally, the overall structure of instruction videos may vary due to possible changes in the type and the order of steps.

To address these challenges, in this paper we develop an unsupervised learning approach that takes advantage of the complementarity of the visual signal in the video and the corresponding natural language narration to resolve their ambiguities. We assume that videos of the same task share the same sequence of ordered steps (also called script in the NLP literature [34]), however, the type and temporal locations of individual steps are unknown and should be discovered from the data. This is in contrast to other existing methods for modeling instruction videos [24] that assume a script (recipe) is known and fixed in advance. We address the problem by first performing temporal clustering of text followed by clustering in video, where the two clustering tasks are linked by joint constraints. The complementary nature of the two clustering problems helps to resolve ambiguities in the two individual modalities. For example, two video segments with very different appearance but depicting the same step can be grouped together if they share similar narrations. Conversely, two video segments described with very different expressions, for example, "jack up the car" and "raise the vehicle" can be identified as belonging to the same instruction step because they have similar visual appearance. The output of our method is the script listing the discovered steps of the task as well as the temporal location of each step in the input videos. We validate our method on a new dataset of instruction videos composed of five different tasks[2] with a total of 150 videos and about 800,000 frames.

2 Related work

This work relates to unsupervised and weakly-supervised learning methods in computer vision and natural language processing. Particularly related to ours is the work on learning script-like knowledge from natural language descriptions [7], [13], [34]. These methods aim to discover typical events (steps) and their

- *Jean-Baptiste Alayrac, Piotr Bojanowksi, Nishant Agrawal, Josef Sivic and Ivan Laptev are with INRIA, Département d'Informatique de l'ENS, CNRS, PSL Research University, 75005, Paris, France.*
- *Simon Lacoste-Julien is in the CS & OR Department of Université de Montréal, Montréal, Canada.*

[1] Some instruction videos on YouTube have tens of millions of views, e.g. www.youtube.com/watch?v=J4-GRH2nDvw.

[2] Changing car tire, Perform cardiopulmonary resuscitation (CPR), Jump a car, Repot a plant, Make coffee

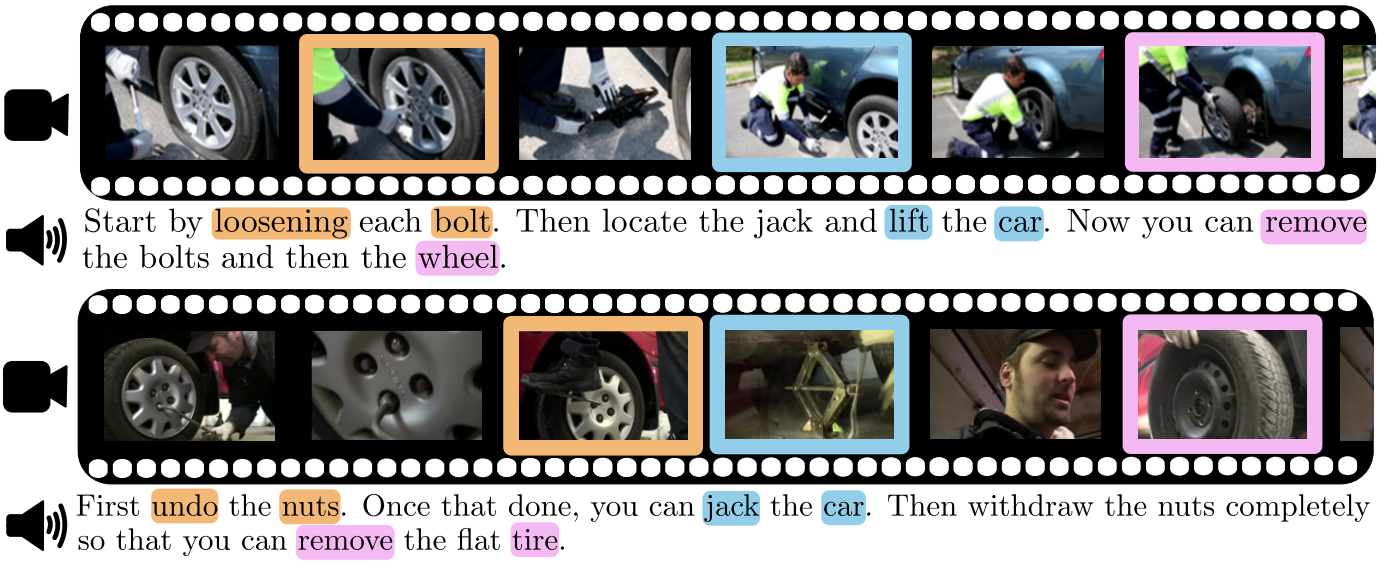


Fig. 1: Given a set of narrated instruction videos demonstrating a particular task, we wish to automatically discover main steps to achieve the task and to associate each step with its corresponding narration and a temporal interval in each video. Here two videos of changing a car tire are illustrated by corresponding frames and excerpts of narrations. Steps of the same type are highlighted by the same color. Note the large variations in narrations and appearance of corresponding steps across videos.

order for particular scenarios (tasks)[3] such as "cooking scrambled egg", "taking a bus" or "making coffee". While [7] uses large-scale news copora, [34] argues that many events are implicit and are not described in such general-purpose text data. Instead, [13], [34] use event sequence descriptions collected for particular scenarios. Differently to this work, we learn sequences of events from narrated instruction videos on the Internet. Such data contains detailed event descriptions but is not structured and contains more noise compared to the input of [13], [34].

Interpretation of narrated instruction videos has been recently addressed in [24]. While this work analyses cooking videos at a great scale, it relies on readily-available recipes which may not be available for more general scenarios. Differently from [24], we here aim to learn the steps of instruction videos using a discriminative clustering approach. A similar task to ours is addressed in [28] using latent variable structured perceptron algorithm to align nouns in instruction sentences with objects touched by hands in instruction videos. However, similarly to [24], [28] uses laboratory experimental protocols as textual input, whereas here we consider a weaker signal in the form of the real transcribed narration of the video.

In computer vision, unsupervised action recognition has been explored in simple videos [30]. More recently, weakly supervised learning of actions in video using video scripts or event order has been addressed in [4], [5], [6], [11], [21]. Particularly related to ours is the work [5] which explores the known order of events to localize and learn actions in training data. While [5] uses manually annotated sequences of events, we here discover the sequences of main events by clustering transcribed narrations of the videos. Related is also the work of [6] that aligns natural text descriptions to video but in contrast to our approach does not discover automatically the common sequence of main steps. Methods in [29], [33] learn in an unsupervised manner the temporal structure of actions from video but do not discover textual expressions for actions as we do in this work. The recent concurrent work [35] is addressing, independently of our work, a similar problem but with a different approach based on a probabilistic generative model and considering a different set of tasks mainly focussed on cooking activities.

Our work is also related to video summarization and in particular to the recent work on category-specific video summarization [32], [37]. While summarization is a subjective task, we here aim to extract the key steps required to achieve a concrete task that consistently appear in the same sequence in the input set of videos. In addition, unlike video summarization [32], [37] we jointly exploit visual and linguistic modalities in our approach.

3 New Dataset of Instruction Videos

We have collected a dataset of narrated instruction videos for five tasks: *Making coffee*, *Changing car tire*, *Performing cardiopulmonary resuscitation (CPR)*, *Jumping a car* and *Repotting a plant*. The videos were obtained by searching YouTube with relevant keywords. The five tasks were chosen so that they have a large number of available videos with English transcripts while trying to cover a wide range of activities that include complex interactions of people with objects and other people. For each task, we took the top 30 videos with English ASR returned by YouTube. We also quickly verified that each video contains a person actually performing the task (as opposed to just talking about it). The result is a total of 150 videos, 30 videos for each task. The average length of our videos is about 4,000 frames (or 2 minutes) and the entire dataset contains about 800,000 frames.

The selected videos have English transcripts obtained from YouTube's automatic speech recognition (ASR) system. To remove the dependence of results on errors of the particular ASR method, we have manually corrected misspellings and punctuations in the output transcriptions. We believe this step may soon become obsolete given rapid improvements of ASR methods. As we do not modify the content of the spoken language in videos, the transcribed verbal instructions still represent an extremely challenging example of natural language with large variability in

[3]We here assign the same meaning to terms "event" and "step" as well as to terms "script" and "task".

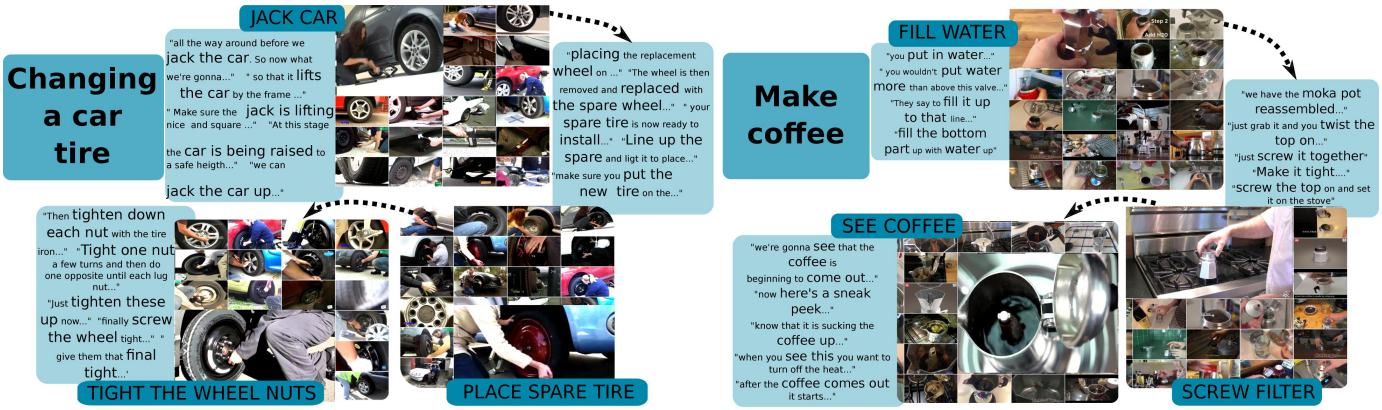


Fig. 2: Illustration of our newly collected dataset of instructions videos. Examples of transcribed narrations together with still frames from the corresponding videos are shown for two (out of 5) tasks: *Changing car tire* and *Making coffee*. The dataset contains challenging real-world videos performed by many different people, captured in uncontrolled settings in a variety of outdoor and indoor environments. Note the large variability of verbal expressions and the terminology in the transcribed narrations as well as the large variability of visual appearance due to viewpoint, used objects, and actions performed in different manner. See our project webpage [2] for more examples.

the used expressions and terminology. Each word of the transcript is associated with a time interval in the video (usually less than 5 seconds) obtained from the closed caption timings.

Figure 2 illustrates two tasks of our newly collected dataset. For each task, we show a subset of 3 events that compose the task. In the following, we refer to these events as *steps* as they are units of a procedure which aims to complete the given *task*. Each step is represented by several sample frames and extracted verbal narrations. Note the large variability of verbal expressions and the terminology in the transcribed narrations as well as the large variability of visual appearance due to viewpoint, used objects, and actions performed in different manner. At the same time, note the consistency of the actions between the different videos and the underlying script of each task.

Manually annotated ground truth. For the purpose of evaluation, we have manually annotated the temporal location in each video of the main steps necessary to achieve the given task. For all tasks, we have defined the ordered sequence of ground truth steps *before* running our algorithm. The choice of these steps was made by an agreement of 2-3 annotators who have watched the input videos and verified the steps on instruction video websites such as http://www.howdini.com or http://www.wikihow.com. Checking the instruction video websites helped us to validate the granularity of the steps. While some steps can be occasionally left out in some videos or the ordering slightly modified, overall we have observed a good consistency in the given sequence of instructions among the input videos and we come back to this in section 3.1. Given the list of steps for each task, we have manually annotated each time interval in each input video to one of the ground truth steps (or no step). The actions of the individual steps are typically separated by hundreds of frames where the narrator transitions between the steps or explains verbally what is going to happen. Note that some steps could be missing in some videos, or could be present but not described in the narration. The narrations and the actual actions in the video often have coarse temporal alignment since the actions are often described before being performed. Our dataset is available at [2]. In the following section we report and discuss the dataset statistics.

3.1 Dataset statistics

To illustrate and quantify different properties of our dataset, we introduce three different scores characterizing (i) the consistency of the step ordering, (ii) the frequency of missing steps and (iii) the frequency of step repetitions. We describe these scores in detail below and then measure them on our new dataset.

Let N be the number of videos for a given task and K the number of steps defined in the ground truth. We assume that the ground truth steps are given in an ordered fashion, meaning the global order is defined as the sequence $(1, \ldots, K)$. For the n-th video, g_n denotes the total number of annotated steps, u_n denotes the number of unique annotated steps, and finally l_n denotes the length of the longest common subsequence between the annotated sequence of steps and the ground truth sequence $(1, \ldots, K)$. Note that, given the terms defined above, we have: $l_n \leq u_n \leq K \leq g_n$. We then define the following scores.

Order consistency error. The *order consistency error* O is the proportion of (non-repeated) steps that are not consistent with the global ordering. In other words, it measures the number of steps that do not fit the global ordering defined in the ground truth sequence divided by the total number of unique annotated steps. More formally, using the l_n and u_n notation for the n-th video defined above the order consistency error is written as:

$$O := 1 - \frac{\sum_{n=1}^{N} l_n}{\sum_{n=1}^{N} u_n}. \quad (1)$$

This score varies between 0 and 1. When the order consistency error is low the videos are consistent with the single ground truth sequence of steps. Note that we report numbers aggregated over all videos instead of reporting the average over all videos. This is to avoid videos with few annotations having a large effect on the resulting overall score. We use this aggregation in all the following metrics.

Missing steps. The *missing steps* score M is the proportion of steps that are visually missing in the videos when compared to the ground truth sequence common to all videos for the task. Using the u_n notation from above, the score is defined as

$$M := 1 - \frac{\sum_{n=1}^{N} u_n}{KN}. \quad (2)$$

Task	Changing tire	Performing CPR	Repoting plant	Making coffee	Jumping cars	**Average**
Order consistency error	0.7%	11%	6%	3%	8%	**6%**
Missing steps	16%	32%	30%	28%	27%	**27%**
Repeated steps	4%	50%	7%	11%	0.4%	**14%**

TABLE 1: Statistics of the newly collected instruction video dataset.

The score varies between 0 and 1. When this score is 0 all steps of the ground truth sequence are depicted in all videos.

Repeated steps. The *repetition score* R is defined as the proportion of steps that are repeated:

$$R := 1 - \frac{\sum_{n=1}^{N} u_n}{\sum_{n=1}^{N} g_n}. \quad (3)$$

The score varies between 0 and 1. When the score is 0 none of the ground truth steps are repeated in the dataset.

Results. Table 1 shows the above scores measured for the five tasks of our instruction videos dataset. Interestingly, we observe relatively low order consistency errors over the five tasks with an average error of only 6%. We believe this can be explained by the goal of instruction videos to give clear, concise and comprehensible audio-visual instructions on how to achieve a given task. On average the videos are missing 27% of the steps, which is relatively high. We believe this illustrates the difficulty of defining the right granularity of the ground truth steps for each task as some optional or implicit steps might be omitted in some videos. Finally, on average 14% of the annotated steps are repeated multiple times in the video. However, examining in detail the per-class results, we observe that this relatively high average score is mainly due to the *Performing CPR* task. CPR is indeed characterized by repetitions of the same steps, namely the alternation between compressions and giving breath. Overall, the relatively high frequency of the missing and repeated steps illustrate the difficulty of our problem. The relatively low order consistency error will be used as an advantage in our method that will be described next.

4 Modelling Narrated Instruction Videos

We are given a set of N instruction videos all depicting the same task (such as "changing a tire"). The n-th input video is composed of a video stream of T_n segments of frames $(x_t^n)_{t=1}^{T_n}$ and an audio stream containing a detailed verbal description of the depicted task. We suppose that the audio description was transcribed to raw text and then processed to a sequence of S_n text tokens $(d_s^n)_{s=1}^{S_n}$. Given this data, we want to automatically recover the sequence of K main steps that compose the given task and locate each step within each input video and text transcription.

We formulate the problem as two clustering tasks, one in text and one in video, applied one after another and linked by joint constraints on two modalities. In the first stage, we cluster the text transcripts into a sequence of K main steps to complete the given task. Empirically, we have found (see results in Section 5.3) that it is possible to discover the sequence of the K main steps for each task with high precision. However, the text itself gives only a poor localization of each step in each video. Therefore, in the second stage we accurately localize each step in each video by clustering the input videos using the sequence of K steps extracted from text as constraints on the video clustering. To achieve this, we use two types of constraints between video and text. First, we assume that both the video and the text narration follow the same sequence of steps. This results in a global ordering constraint on the recovered clustering. Second, we assume that people perform the action approximately at the same time that they talk about it. This constraint temporally links the recovered clusters in text and video. The important outcome of the video clustering stage is that the K extracted steps get propagated by visual similarity to videos where the text descriptions are missing or ambiguous.

We first describe the text clustering in Section 4.1 and then introduce the video clustering with constraints in Section 4.2.

4.1 Clustering transcribed verbal instructions

The goal here is to cluster the transcribed verbal descriptions of each video into a sequence of *main steps* necessary to achieve the task. This stage is important as the resulting clusters will be used as constraints for jointly learning and localizing the main steps in video. We assume that the important steps are common to many of the transcripts and that the sequence of steps is (roughly) preserved in all transcripts. Hence, following [34], we formulate the problem of clustering the input transcripts as a multiple sequence alignment problem. However, in contrast to [34] who cluster manually provided descriptions of each step, we wish to cluster transcribed verbal instructions. Hence our main challenge is to deal with the variability in spoken natural language. To overcome this challenge, we take advantage of the fact that completing a certain task usually involves interactions with objects or people and hence we can extract a more structured representation from the input text stream.

More specifically, we represent the textual data as a sequence of *direct object relations*. A direct object relation d is a pair composed of a verb and its direct object complement, such as "remove tire". This representation contains relevant information about the interaction (*verb*) between the demonstrator and the surrounding environment (*direct object complement*), while being compact. Such a direct object relation can be extracted from the dependency parser of the input transcribed narration [10]. We denote the set of all different direct object relations extracted from all narrations as $\mathcal{D}$, with cardinality D. For the n-th video, we thus represent the text signal as a sequence of direct object relation tokens: $d^n = (d_1^n, \ldots, d_{S_n}^n)$, where the length S_n of the sequence varies from one video clip to another. This step is key to the success of our method as it allows us to convert the problem of clustering raw transcribed text into an easier problem of clustering sequences of direct object relations. The goal is now to extract from the narrations the most common sequence of K main steps to achieve the given task. To achieve this, we first find a globally consistent alignment of the direct object relations that compose all text sequences by solving a multiple sequence alignment problem. Second, we pick from this alignment the K most globally consistent clusters across videos.

Multiple sequence alignment model. We formulate the first stage of finding the common alignment between the input sequences of direct object relations as a multiple sequence alignment problem with the *sum-of-pairs score* [39]. In details, a global

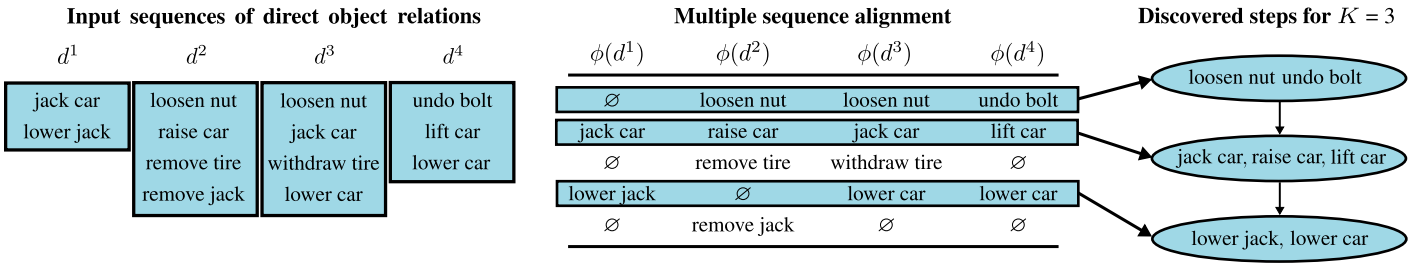


Fig. 3: **Clustering transcribed verbal instructions. Left:** The input raw text for each video is converted into a sequence of direct object relations. Here, an illustration of four sequences from four different videos is shown. **Middle:** Multiple sequence alignment is used to align all sequences together. Note that different direct object relations are aligned together as long as they have the same sense, e.g. "loosen nut" and "undo bolt". **Right:** The main instruction steps are extracted as the $K = 3$ most common steps in all the sequences.

alignment can be defined by re-mapping each input sequence d^n of tokens to a global common template of L slots, for L large enough. We let $(\phi(d^n))_{1\leq l\leq L}$ represent the (increasing) re-mapping for sequence d^n at the new locations indexed by l: $\phi(d^n)_l$ represents the direct object relation put at location l, with $\phi(d^n)_l = \varnothing$ if a slot is left empty (denoting the insertion of a gap in the original sequence of tokens). See the middle of Figure 3 for an example of re-mapping. The goal is then to find a global alignment that minimizes the following sum-of-pairs cost function:

$$\sum_{(n,m)} \sum_{l=1}^{L} c(\phi(d^n)_l, \phi(d^m)_l), \tag{4}$$

where $c(d_1, d_2)$ denotes the cost of aligning the direct object relations d_1 and d_2 at the same common slot l in the global template. The above cost thus denotes the sum of all pairwise alignments of the individual sequences (the outer sum), where the quality of each alignment is measured by summing the cost c of matches of individual direct object relations mapped into the common template sequence. We use a negative cost when d_1 and d_2 are similar according to the distance in the WordNet tree [12], [27] of their verb and direct object constituents, and positive if they are dissimilar (details are given in Section 5). As the verbal narrations can talk about many other things than the main steps of a task, we set $c(d, d') = 0$ if either d or d' is $\varnothing$. An illustration of clustering the transcribed verbal instructions into a sequence of K steps is shown in Figure 3.

Multiple sequence alignment as a quadratic program. Optimizing the cost (4) is NP-hard [39] because of the combinatorial nature of the problem. The standard solution from computational biology is to apply a heuristic algorithm that proceeds by incremental pairwise alignment using dynamic programming [22]. In contrast, we reformulate the multiple sequence alignment problem given by (4) as an integer quadratic program (IQP) with combinatorial constraints. To the best of our knowledge, this is a new formulation of the multiple sequence alignment (MSA) problem for which we can apply the Frank-Wolfe algorithm [16] and consistently obtain in our setting better values of the MSA objective (4) than the current state-of-the-art MSA heuristic algorithms. To be able to transform the MSA problem into an IQP, we first correctly encode the important quantities of our problem in a matrix form. There are three relevant quantities: (i) the cost of aligning together two different direct object relations (encoded in C_o), (ii) the content of each individual sequences (encoded in Y_n) and (iii) the mapping of each sequence to the common template (encoded in U_n). We now give details about these different variables and how to combine them in order to get the final IQP formulation (6).

Cost matrix C_o. We summarize the cost of aligning non-empty direct object relations by the matrix $C_o \in \mathbb{R}^{D\times D}$. $(C_o)_{ij}$ is equal to the cost of aligning the i-th and the j-th direct object relation from the dictionary together.

Individual sequences Y_n. We encode the identity of a direct object relation with a D-dimensional indicator vector. The text sequence n can then be represented by an indicator matrix $Y_n \in \{0,1\}^{S_n\times D}$. The j-th row of Y_n indicates which direct object relations is evoked at the j-th position.

Mapping of the sequence U_n. Similarly, the token re-mapping $(\phi(d^n))_{1\leq l\leq L}$ can be represented as a $L \times D$ indicator matrix where each row l encodes which token is appearing in slot l (and a whole row of zero is used to indicates an empty $\varnothing$ slot). This re-mapping can be constructed from the following two pieces of information: the input sequence Y_n and the mapping of each element of the sequence to the global template. We represent the latter by the decision matrix $U_n \in \{0,1\}^{S_n\times L}$ that gives which element of the sequence s is re-mapped to which global template slot l. We thus have $\phi(d^n) = U_n^T Y_n$ (as a $L\times D$ indicator matrix).

Quadratic cost. Given this encoding, the cost matrix C_o, and the fact that the alignment of empty slots has zero cost, we can rewrite the MSA problem that minimizes the sum-of-pairs objective (4) as follows:

$$\begin{aligned} \underset{U_n, n\in\{1,\dots,N\}}{\text{minimize}} \quad & \sum_{(n,m)} \mathrm{Tr}(U_n^T Y_n C_o Y_m^T U_m) \\ \text{subject to} \quad & U_n \in \mathcal{U}_n,\ n = 1,\dots,N. \end{aligned} \tag{5}$$

In the above equation, the trace (Tr) is computing the cost of aligning sequence m with sequence n (the inner sum in (4)). Moreover, $\mathcal{U}_n$ is a constraint set that encodes the fact that U_n has to be a valid (increasing) re-mapping.[4] We can then eliminate the video index n by simply stacking the assignment matrices U_n in one matrix U of size $S \times L$. Similarly, we denote Y the $S \times D$ matrix which is obtained by the concatenation of all the Y_n matrices. Finally, we can rewrite the equation (5) as a quadratic program over the (integer) variable U:

$$\underset{U}{\text{minimize}}\ \mathrm{Tr}(U^T B U),\ \text{subject to}\ U \in \mathcal{U}. \tag{6}$$

In this equation, the $S \times S$ matrix B is obtained from the input sequences and the cost between different direct object relations by computing $B := Y C_o Y^T$. It represents the pairwise cost at the token level, i.e. the cost of aligning token s in one sequence to token s' in another sequence.

[4] More formally $\mathcal{U}_n := \{U \in \{0,1\}^{S_n\times L}$ s.t. $U\mathbf{1}_L = \mathbf{1}_{S_n}$ and $\forall l, (U_{sl} = 1) \Rightarrow (\forall s' > s, l' \leq l, U_{s'l'} = 0)\}$.

Optimization using Frank-Wolfe. Problem (6) is an integer quadratic program with combinatorial constraints for which the Frank-Wolfe optimization algorithm has been used recently with increasing success [5], [16], [17], [19]. We therefore also apply Frank-Wolfe here. Following [5], we first perform a continuous relaxation of the set of constraints $\mathcal{U}$ by replacing it with its convex hull $\bar{\mathcal{U}}$. Note that the Frank-Wolfe optimization algorithm [16] can solve quadratic program over constraint sets for which we have access to an efficient linear minimization oracle. In the case of $\mathcal{U}$, the linear oracle can be solved exactly with a dynamic program. We note here that even with the continuous relaxation over $\bar{\mathcal{U}}$, the resulting problem is still non-convex because B is not positive semidefinite. However, the standard convergence proof for Frank-Wolfe can easily be extended to show that it converges at a rate of $O(1/\sqrt{k})$ to a stationary point on non-convex objectives [18]. Once the algorithm has converged to a (local) stationary point, we need to round the fractional solution to obtain a valid encoding U. We follow here a similar rounding strategy that was originally proposed by [8] and then re-used in [17]: we pick the last visited corner (which is necessarily integer) which was given as a solution to the linear minimization oracle (this is called Frank-Wolfe rounding).

We have observed empirically (see results in Section 5.2) that the Frank-Wolfe algorithm was giving better solutions (in terms of objective (4)) than the state-of-the-art heuristic procedures for this task [14], [22]. Our Frank-Wolfe based solvers also offer us greater flexibility in defining the alignment cost and scale better with the length of input sequences and the vocabulary of direct object relations.

Extracting the main steps. After a global alignment is obtained, we sort the global template l by the number of direct object relations aligned to each slot. Given K as input, the top K slots give the main instruction steps for the task, unless there are multiple steps with the same support, which go beyond K. In this case, we pick the next smaller number below K which excludes these ties, allowing the choice of an *adaptive* number of main instruction steps when there is not enough saliency for the last steps. This strategy essentially selects $k \leq K$ salient steps, while refusing to make a choice among steps with equal support that would increase the total number of steps beyond K. As we will see in our results in Section 5.3, our algorithm sometimes returns a much smaller number than K for the main instruction steps, giving more robustness to the exact choice of parameter K.

Encoding of the output. We post-process the output of multiple sequence alignment into an assignment matrix $R_n \in \{0,1\}^{S_n \times K}$ for each input video n, where $(R_n)_{sk} = 1$ means that the direct object token d_s^n has been assigned to step k. If a direct object has not been assigned to any step, the corresponding row of the matrix R_n will be zero.

4.2 Discriminative clustering of videos under text constraints

Given the output of the text clustering that identified the important K steps forming a task, we now want to find their temporal location in the video signal. We formalize this problem as looking for an assignment matrix $Z_n \in \{0,1\}^{T_n \times K}$ for each input video n, where $(Z_n)_{tk} = 1$ indicates the visual presence of step k at time interval t in video n, and T_n is the length of video n. Similarly as for the assignment matrix R_n, we allow the possibility that a whole row of Z_n is zero, indicating that no step is visually present for the corresponding time interval.

We propose to tackle this problem using a discriminative clustering approach with global ordering constraints, as was successfully used in the past for the temporal localization of actions in videos [5], but with additional *weak temporal constraints*. In contrast to [5] where the order of actions was manually given for each video, our multiple sequence alignment approach automatically discovers the main steps. More importantly, we also use the *text caption timing* to provide a fine-grained weak temporal supervision for the visual appearance of steps, which is described next.

Temporal weak supervision from text. From the output of the multiple sequence alignment (encoded in the matrix $R_n \in \{0,1\}^{S_n \times K}$), each direct object token d_s^n has been assigned to one of the possible K steps, or to no step at all. We use the tokens that have been assigned to a step as a constraint on the visual appearance of the same step in the video (using the assumption that people do what they say approximately when they say it). We encode the closed caption timing alignment by a binary matrix $A_n \in \{0,1\}^{S_n \times T_n}$ for each video, where $(A_n)_{st}$ is 1 if the s-th direct object is mentioned in a closed caption that overlaps with the time interval t in video. Note that this alignment is only approximate as people usually do not perform the action exactly at the same time that they talk about it, but instead with a varying delay. Second, the alignment is noisy as people typically perform the action only once, but often talk about it multiple times (e.g. in a summary at the beginning of the video). We address these issues by the following two *weak supervision* constraints. First, we consider a larger set of possible time intervals $[t-\Delta_b, t+\Delta_a]$ in the matrix A rather than the exact time interval t given by the timing of the closed caption. Δ_b and Δ_a are global parameters fixed either qualitatively, or by cross-validation if labeled data is provided. Second, we put as a constraint that the action happens at least once in the set of all possible video time intervals where the action is mentioned in the transcript (rather than every time it is mentioned). These constraints can be encoded as the following linear inequality constraint on Z_n: $A_n Z_n \geq R_n$.[5]

Ordering constraint. In addition, we also enforce that the temporal order of the steps appearing visually is consistent with the discovered script from the text, encoding our assumption that there is a common ordered script for the task across videos. We encode these sequence constraints on Z_n in a similar manner to [6], which was shown to work better than the encoding used in [5]. In particular, we only predict the *most salient* time interval in the video that describes a given step. This means that a particular step is assigned to *exactly one* time interval in each video. We denote by $\mathcal{Z}_n$ this sequence ordering constraint set.[6]

Discriminative clustering. The main motivation behind discriminative clustering is to find a *clustering* of the data that can be easily recovered by a *linear classifier* through the minimization of an appropriate *cost function* over the assignment matrix Z_n. The approach introduced in [3] allows to easily add prior information on the expected clustering. Such priors have been recently introduced in the context of aligning video and text [5], [6] in the form of ordering constraints over the latent label variables. Here we use

[5] When $R_{sk} = 0$, then this constraint does not do anything. When $R_{sk} = 1$ (i.e. the text token s was assigned to the main action k), then the constraint enforces that $\sum_{t \in A_s} Z_{tk} \geq 1$, where A_s. represents which video frames are temporally close to the caption time of the text token s. It thus then enforces that at least one temporally close video frame is assigned to the main action k.

[6] More formally $\mathcal{Z}_n := \{Z \in \{0,1\}^{T_n \times K}$ s.t. $\mathbf{1}_{T_n}^T Z \mathbf{1}_K = K$ and $\forall t, (Z_{tk} = 1) \Rightarrow (\forall t' > t, k' \leq k, Z_{t'k'} = 0)$ and $(\forall t' \neq t, Z_{t'k} = 0)\}$.

Changing a tire				Make coffee				Repot a plant			
GT(11)	$K \leq 7$	$K \leq 10$	$K \leq 15$	GT(10)	$K \leq 7$	$K \leq 10$	$K \leq 15$	GT(7)	$K \leq 7$	$K \leq 10$	$K \leq 15$
put brake on				*grind coffee*				*cover hole*			**take piece**
get tools out		**get tire**	**get tire**	*put filter*							**keep soil**
start loose	**loosen nut**	**loosen nut**	**loosen nut**	*add coffee*		**put coffee**	**put coffee**				**stop soil**
			lift car	*even surface*				*take plant*	**take plant**	**take plant**	**take plant**
	put jack	**put jack**	**put jack**			**fill chamber**	**fill chamber**	*put soil*	**use soil**	**use soil**	**use soil**
							make noise	*loosen root*	**loosen soil**	**loosen soil**	**loosen soil**
			raise vehicle	*fill water*	**fill water**	**fill water**	**fill water**	*place plant*	**place plant**	**place plant**	**place plant**
jack car	**jack car**	**jack car**	**jack car**	*screw top*		**put filter**	**put filter**				
							fill basket	*add top*	**add soil**	**add soil**	**add soil**
unscrew wheel	**remove nut**	**remove nut**	**remove nut**			**see steam**	**see steam**				**fill pot**
remove wheel		**take wheel**	**take wheel**	*put stove*	**take minutes**	**take minutes**	**take minutes**				**get soil**
put wheel	**take tire**	**take tire**	**take tire**		**make coffee**	**make coffee**	**make coffee**				**give drink**
screw wheel		**put nut**	**put nut**					*water plant*	**water plant**	**water plant**	**water plant**
lower car	**lower jack**	**lower jack**	**lower jack**	*see coffee*	**see coffee**	**see coffee**	**see coffee**				**give watering**
			remove jack	*withdraw stove*			**turn heat**				
tight wheel	**tighten nut**	**tighten nut**	**tighten nut**	*pour coffee*	**make cup**	**make cup**	**make cup**				
put things back			**take tire**				**pour coffee**				
Precision	0.85	0.90	0.71	Precision	0.80	0.67	0.54	Precision	1.00	1.00	0.54
Recall	0.54	0.90	0.90	Recall	0.40	0.60	0.70	Recall	0.86	0.86	1.00

Performing CPR				Jumping cars			
GT(11)	$K \leq 7$	$K \leq 10$	$K \leq 15$	GT(12)	$K \leq 7$	$K \leq 10$	$K \leq 15$
open airway	**open airway**	**open airway**	**open airway**	*get cars*			
check response				*open hood*			
call 911							**have terminal**
check breathing							**attach cable**
check pulse		**put hand**	**put hand**	*connect red A*	**connect cable**	**connect cable**	**connect cable**
	tilt head	**tilt head**	**tilt head**				
	lift chin	**lift chin**	**lift chin**		**charge battery**	**charge battery**	**charge battery**
	give breath	**give breath**	**give breath**	*connect red B*	**connect end**	**connect end**	**connect end**
give breath	**do compr.**	**do compr.**	**do compr.**	*connect black A*			
give compression	**open airway**	**open airway**	**open airway**	*connect ground*			
		start compr.	**start compr.**	*start car A*	**start car**	**start car**	**start car**
			continue cpr	*start car B*			**start vehicle**
		do compr.	**do compr.**				
			put hand				**start engine**
		give breath	**give breath**	*remove ground*	**remove cable**	**remove cable**	**remove cable**
				remove black A	**disconnect cable**	**disconnect cable**	**disconnect cable**
				remove red B			
				remove red A			
Precision	0.50	0.40	0.33	Precision	0.83	0.83	0.69
Recall	0.43	0.57	0.57	Recall	0.42	0.42	0.67

TABLE 2: Automatically recovered sequences of steps for the five tasks considered in this work. Each recovered step is represented by one of the aligned direct object relations (shown in bold). Note that most of the recovered steps correspond well to the ground truth steps (showed in italic). The results are shown for setting the maximum number of discovered steps, $K = \{7, 10, 15\}$. Note how our method automatically selects less than K steps in some cases. These are the automatically chosen $k \leq K$ steps that are the most salient in the aligned narrations as described in Section 4.1.

a similar approach to cluster the N input video streams (x_t) into a sequence of K steps, as follows. We represent each time interval by a d-dimensional feature vector. The feature vectors for the n-th video are stacked in a $T_n \times d$ design matrix denoted by X_n. We denote by X the $T \times d$ matrix obtained by the concatenation of all X_n matrices (and similarly, by Z, R and A the appropriate concatenation of the Z_n, R_n and A_n matrices over n). In order to obtain the temporal localization into K steps, we learn a linear classifier represented by a $d \times K$ matrix denoted by W. This model is shared among all videos.

The target assignment $\hat{Z}$ is found by minimizing the clustering cost function h under both the consistent script ordering constraints $\mathcal{Z}$ and our weak supervision constraints:

$$\underset{Z}{\text{minimize}} \quad h(Z) \quad \text{s.t.} \quad \underbrace{Z \in \mathcal{Z}}_{\text{ordered script}}, \quad \underbrace{AZ \geq R}_{\substack{\text{weak textual} \\ \text{constraints}}}. \tag{7}$$

The clustering cost $h(Z)$ is given as in DIFFRAC [3] as:

$$h(Z) = \min_{W \in \mathbb{R}^{K \times d}} \underbrace{\frac{1}{2T}\|Z - XW\|_F^2}_{\text{Discriminative loss on data}} + \underbrace{\frac{\lambda}{2}\|W\|_F^2}_{\text{Regularizer}}. \tag{8}$$

The first term in (8) is the discriminative loss on the data that measures how easy the input data X is separable by the linear classifier W when the target classes are given by the assignments Z. For the squared loss considered in eq. (8), the unique optimal weights W^* minimizing (8) can be found in closed form, which significantly simplifies the computation:

$$W^*(Z) = (X^T X + T\lambda I_d)^{-1} X^T Z, \tag{9}$$

where I_d is the d-dimensional identity matrix. We obtain the explicit form for $h(Z)$ by substituting the expression (9) for $W^*(Z)$ in equation (8) and properly simplifying the expression:

$$h(Z) = \frac{1}{2T}\text{Tr}(ZZ^T B), \tag{10}$$

where $B := I_T - X(X^T X + T\lambda I_d)^{-1} X^T$ is a strictly positive definite matrix (and so h is actually strongly convex). The

clustering cost is a quadratic function in Z, encoding how the clustering decisions in one interval t interact with the clustering decisions in another interval t'. Next, we explain how we can optimize the clustering cost $h(Z)$ subject to the constraints of problem (7) using the Frank-Wolfe algorithm.

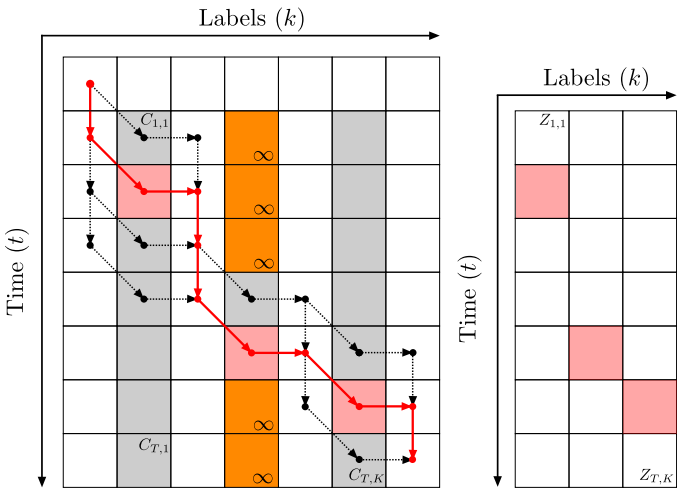
Frank Wolfe algorithm for minimizing $h(Z)$. The Frank Wolfe algorithm is well adapted for our problem as we know how to efficiently solve linear programs over our constraints. Recall that theses constraints encode several concepts. First, it imposes the temporal consistency between the text stream and the video stream. We recall that this constraint was written as $AZ \geq R$, where A encodes the temporal alignment constraints between video and text (type I). Second, it includes the event ordering constraints within each video input (type II). Finally, it encodes the fact that each event is assigned to exactly one time interval within each video (type III). The last two constraints are encoded in the set of constraints $\mathcal{Z}$. To summarize, let $\tilde{\mathcal{Z}}$ denote the resulting discrete feasible space for Z i.e. $\tilde{\mathcal{Z}} := \{Z \in \mathcal{Z} \,|\, AZ \geq R\}$. We are then left with a problem in Z which is still hard to solve because the set $\tilde{\mathcal{Z}}$ is not convex. To approximately optimize h over $\tilde{\mathcal{Z}}$, we again follow the strategy of [5], [6] by replacing $\tilde{\mathcal{Z}}$ by its convex hull $\text{conv}(\tilde{\mathcal{Z}})$. We then use the Frank-Wolfe algorithm to get a fractional solution $Z^* \in \text{conv}(\tilde{\mathcal{Z}})$. Finally, we find a feasible candidate $\hat{Z} \in \tilde{\mathcal{Z}}$ by using a rounding procedure. We now give the details of these steps.

Linear program for our constraints. First, we note that the linear oracle of the Frank-Wolfe algorithm can be solved separately for each video n. Indeed, because we solve a linear program, there is no quadratic term that brings dependence between different videos in the objective, and moreover all the constraints are blockwise in n. Thus, in the following, we will give details for one video only by adding an index n to $\tilde{\mathcal{Z}}$, to Z and to T. Formally, the linear oracle corresponds to the following problem:

$$\min_{Z_n \in \tilde{\mathcal{Z}}_n} \text{Tr}(C_n^\top Z_n), \tag{11}$$

where $C_n \in \mathbb{R}^{T_n \times K}$ is a cost matrix that typically comes from the gradient computation of h with respect to Z_n at the current iterate. We now show that this problem can be solved by an efficient dynamic program.

Dynamic program. Using the formalism of [6], we cast problem (11) as a search for an optimal path inside a cost matrix $\tilde{C}$ that we can solve using dynamic programing. From the constraint of type III (unique prediction per step), we know that each column k of Z_n has exactly one 1 (to be found). From the ordering constraint (type II), we know that if $(Z_n)_{tk} = 1$, then the only possible locations for a 1 in the $(k+1)$-th column is for $t' > t$ (i.e. the pattern of 1's is going downward when traveling from left to right in Z_n). Note that there can be "jumps" in between the time assignment for two subsequent steps k and $k+1$. In order to encode this possibility using a continuous path search in a matrix, we insert dummy columns into the cost matrix C. We first subtract the minimum value from C and then insert columns filled with zeros in between every pair of columns of C. In the end, we pad C with an additional row filled with zeros at the bottom. Finally, the problem that we are interested in is subject to the additional linear constraints given by the clustering of text transcripts (constraints of type I). These constraint can be added by constraining the path in the dynamic programming algorithm. This can be done for instance by setting an infinite alignment cost outside of the constrained region. The resulting cost matrix $\tilde{C}$ is of size $(T_n + 1) \times (2K + 1)$ and is illustrated along with the corresponding update rules in Figure 4.



(a) DP cost matrix $\tilde{C}$ (b) DP solution Z

Fig. 4: Illustration of the dynamic programming solution to the linear program (11). (a) shows an example cost matrix $\tilde{C}$ and the corresponding optimal path in red. The gray entries in the matrix $\tilde{C}$ correspond to the values from the original cost matrix C (see text). The white entries have minimal cost and are thus always preferred over any gray entry. The orange entries have maximal cost, e.g. ∞, and correspond to text constraints (type I). These constraints reduce the number of possible paths. The obtained latent variable solution Z is displayed in (b). Red and white entries respectively correspond to the value 1 and 0.

Extracting discrete temporal locations. At the end of the Frank-Wolfe optimization algorithm, we obtain a continuous solution for assignment matrix Z_n^* that encodes the (fractional) solution for the temporal location of each step in video n. By stacking matrices for all videos together, we obtain a continuous solution Z^*. From the definition of h, we can also look at the corresponding model $W^*(Z^*)$ defined in equation (9) which is shared among all videos. All Z_n^* have to be rounded in order to obtain a feasible point for the initial, non relaxed problem. Several rounding options were suggested in [6]; we found experimentally that the option using W^* gives better results in our case. More precisely, in order to get a good feasible binary matrix $\hat{Z}_n \in \tilde{Z}_n$, we solve the following problem: $\min_{Z_n \in \tilde{\mathcal{Z}}_n} \|Z_n - X_n W^*\|_F^2$. By expanding the norm, we notice that this corresponds to a simple linear program over $\tilde{\mathcal{Z}}_n$ as in equation (11) that can be solved using again the same dynamic program detailed above. Finally, we stack these rounded matrices $\hat{Z}_n$ to obtain our predicted assignment matrix $\hat{Z} \in \tilde{\mathcal{Z}}$.

5 Experimental Evaluation

In this section, we first give the implementation details including the description of extracted text and video features. Then we present results divided into five main experiments: (i) in Section 5.2, we evaluate the performance of our method for solving the multiple sequence alignment problem, (ii) in Section 5.3, we evaluate the quality of steps extracted from the transcribed narrations, (iii) in Section 5.4, we evaluate the temporal localization accuracy of the recovered steps in video using constraints derived from the transcribed narrations, (iv) in Section 5.5, we explore

the effects of the different parameters of our method, and finally (v) in Section 5.6, we investigate the possibility of iterating the two stages of our clustering approach. All the data and code are available on our project webpage [2].

5.1 Implementation details.

Here we describe the extracted video and text features, followed by the details of the WordNet distance used in the multiple sequence alignment algorithm. Finally, we also discuss the running time of the proposed approach and its scalability.

Video and text features. We represent the transcribed narrations as sequences of direct object relations. For this purpose, we run a dependency parser [10] on each transcript. We lemmatize all direct object relations and keep the ones for which the direct object corresponds to nouns. To represent a video, we use motion descriptors in order to capture actions (such as loosening, jacking-up, giving compressions) and frame appearance descriptors to capture the depicted objects (e.g., tire, jack, car). In detail, we split each video into 10-frame time intervals and represent each interval by its motion and appearance descriptors aggregated over a longer block of 30 frames. The motion representation is a histogram of local optical flow (HOF) descriptors aggregated into a single bag-of-visual-word vector of 2,000 dimensions [38]. The visual vocabulary is generated by k-means on a separate large set of training descriptors. To capture the depicted objects in the video, we apply the VGG-verydeep-16 CNN [36] over each frame in a sliding window manner over multiple scales. This can be done efficiently in a fully convolutional manner. The resulting 512-dimensional feature maps of conv5 responses are then aggregated into a single bag-of-visual-word vector of 1,000 dimensions, which aims to capture the presence/absence of different objects within each video block. A similar representation (aggregated into compact VLAD descriptor) was shown to work well recently for a variety of recognition tasks [9]. The bag-of-visual-word vectors representing the motion and the appearance are normalized using the Hellinger normalization and then concatenated into a single 3,000 dimensional vector representing each time interval.

WordNet distance. For the multiple sequence alignment presented in Section 4.1, we set $c(d_1, d_2) = -1$ if d_1 and d_2 have both their verbs and direct objects that match exactly in the WordNet tree (distance equal to 0). Otherwise we set $c(d_1, d_2)$ to be 100. We found this relatively strict requirement to work well ensuring high-precision of the resulting alignment while taking advantage of the semantic information contained in the WordNet tree as it allows to align two expressions that are not identical as long as they are in the same node of the tree. While we have observed the WordNet distance to work well, we also investigate using distance based on the word2vec embedding [26]. In detail, we use the average cosine similarity between the verbs and the objects composing the direct object relations. If the cosine similarity is greater than 0.7[7], we set $c(d_1, d_2) = -1$, otherwise $c(d_1, d_2) = 100$. We use word2vec embedding pre-trained on GoogleNews. We have observed that pre-training the embedding on instruction videos results in a similar performance.

Running time and scalability. For the multiple sequence alignment problem (with $N = 30$, $D = 50$), 400 iterations of our algorithm take less than 10 seconds on a single 2.40 GHz core, which is sufficient for convergence to a stationary point. For video alignment (with $N = 30$, $d = 3000$, $T = 20000$), 600 iterations take less than 20 minutes on the same hardware configuration. Note that the discriminative clustering algorithm could be further scaled-up to 1000s of videos using the Block-Coordinate Frank-Wolfe algorithm [20], [31] as has been recently shown in [25]. This is possible as our constraints, similar to [25], decompose over the different videos.

[7]This threshold has been chosen manually to ensure a good tradeoff between precision and recall.

5.2 Results of multiple sequence alignment

In this section we evaluate the optimization performance of our relaxation of the multiple sequence alignment (MSA) problem. The (MSA) problem (6) is in general NP-hard [39], as is typical for integer quadratic programs. However, significant amount of work has been done in computational biology to develop efficient heuristics to solve this problem, as it arises in several important applications. We briefly describe below some of the existing heuristics, and then compare the optimization performance with our Frank-Wolfe optimization approach, which gave surprisingly good empirical results.[8]

Standard methods. We compare performance to the standard state-of-the-art method for multiple sequence alignment [22]. Similarly to [14], this method first aligns a pair of sequences and merges them in a common template. Then it aligns a new sequence to the template and updates the template. It continues aligning additional sequences until no sequence is left. Differently from [14], this method represents the template by a partial order graph instead of a simple linear representation, which results in a higher accuracy of the final alignment. For the experiments, we use the author's implementation [1].

Results. In Table 3, we give the value of the objective (6) (lower is better) for the rounded solutions obtained by the two different optimization approaches for the MSA problem on our five tasks. Interestingly, we observe that the Frank-Wolfe algorithm consistently outperforms the state-of-the-art MSA method of [22] in our setting.

5.3 Results of step discovery from text narrations

In the previous section we evaluated the optimization performance of our multiple sequence alignment algorithm described in Sec 4.1. In this section, we evaluate how well our multiple sequence alignment approach discovers the main steps of each task from the text narrations. Table 2 shows the automatically recovered sequences of steps for the five tasks considered in this work. The results are shown for setting the maximum number of discovered steps, $K = \{7, 10, 15\}$. Note how our method automatically selects less than K steps in some cases. These are the automatically chosen $k \leq K$ steps that are the most salient in the aligned narrations as described in Section 4.1. This is notably the case for the *Repotting a plant* task. Even for $K \leq 10$, the algorithm recovers only 6 steps that match very well the seven ground truth steps for this task. This saliency based task selection is important because it allows for a better precision at high K without lowering much the recall. Please note also how the steps and their ordering recovered by our method correspond well to the ground truth steps for each task.

[8]We stress here that we do not claim that our formulation of the multiple sequence alignment (MSA) problem as a quadratic program outperforms the state-of-the-art computational biology heuristics for problems *arising in biology*. We report our observations when multiple sequence alignment is applied to our problem set-up, which might have a structure for which these heuristics are not appropriate.

Task	Changing tire	Performing CPR	Repotting plant	Making coffee	Jumping cars
Poa [22]	11.30	-3.82	1.65	-2.99	4.55
Ours using Frank-Wolfe	**-5.18**	**-4.51**	**-3.55**	**-3.86**	**-4.67**

TABLE 3: Comparison of different optimization approaches for solving problem (6). (Objective value, lower is better).

For *CPR*, our method recovers fine-grained steps e.g. *tilt head*, *lift chin*, which are not included in the main ground truth steps, but nevertheless could be helpful in some situations. For *Changing tire*, we also recover more detailed actions such as *remove jack* or *put jack*. In some cases, our method recovers repeated steps. For example, for *CPR* our method learns that one has to alternate between *giving breath* and *performing compressions* even if this alternation was not annotated in the ground truth. For *Jumping Cars* our method learns that cables need to be connected twice (to both cars). These results demonstrate that our method is able to automatically discover meaningful scripts describing very different tasks. The results also show that the constraint of a single script providing an ordering of events is a reasonable prior for a variety of different tasks. In addition to displaying these qualitative results we also report in Table 2 quantitative evaluation using precision and recall. To compute these scores, we first manually define a mapping between the discovered steps and the ground truth steps with the following two constraints: (i) a ground truth step can match to at most one discovered step and (ii) the ordering of the discovered steps must match the ordering of the ground truth steps. Once the discovered steps are mapped to the ground truth steps, we compute precision and recall as follows. Precision is the number of correctly discovered steps (mapped to ground truth) divided by the number of all discovered steps. Recall is the number of correctly discovered steps (mapped to ground truth) divided by the total number of ground truth steps. Note that obtained precision and recall values are relatively high, which demonstrates the strength of the proposed multiple sequence alignment algorithm.

5.4 Results of localizing instruction steps in video

In the previous section, we have evaluated the quality of the sequences of steps recovered from the transcribed narrations. In this section, we evaluate how well we localize the individual instruction steps in the video by running our complete two-stage approach from Section 4.

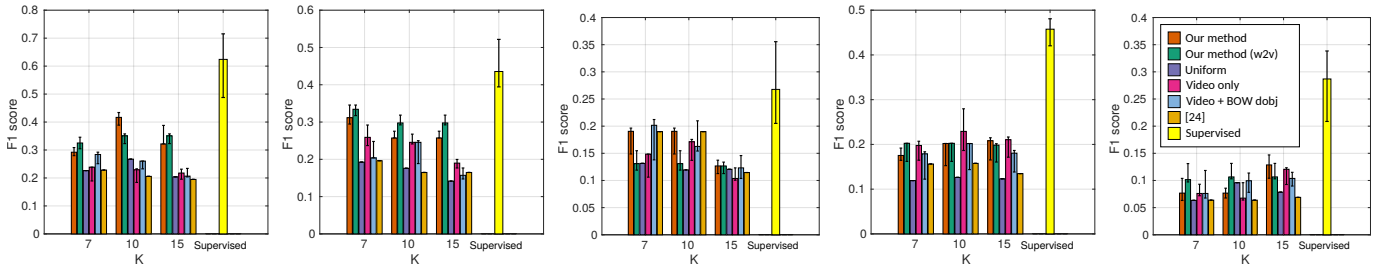
Evaluation metric. To evaluate the temporal localization, we need to have a one-to-one mapping between the discovered steps in the videos and the ground truth steps. Following [23], we look for a one-to-one global matching (shared across all videos of a given task) that maximizes the evaluation score for a given method (using the Hungarian algorithm). Note that this mapping is used only for evaluation, the algorithm does not have access to the ground truth annotations for learning.

The goal is to evaluate whether each ground truth step has been correctly localized in all instruction videos. We thus use the *F1 score* that combines precision and recall into a single score as our evaluation measure. For a given video and a given recovered step, our video clustering method predicts exactly one video time interval t. This detection is considered correct if the time interval falls inside any of the corresponding ground truth intervals, and incorrect otherwise (resulting in a false positive for this video). We compute the recall across all steps and videos, defined as the ratio of the number of correct predictions over the total number of possible ground truth steps across videos. A recall of 1 indicates that every ground truth step has been correctly detected across all videos. The recall decreases towards 0 when we miss some ground truth steps (missed detections). This happens either because this step was not recovered globally, or because it was detected in the video at an incorrect location. This is because the algorithm predicts exactly one occurrence of each step in each video. Similarly, precision measures the proportion of correct predictions among all $N \cdot K_{\text{pred}}$ possible predictions, where N is the number of videos and K_{pred} is the number of main steps used by the method. The F1 score is the harmonic mean of precision and recall, giving a score that ranges between 0 and 1, with the perfect score of 1 when all the steps are predicted at their correct locations in all videos.

Hyperparameters. We set the values of text constraint time interval parameters, Δ_b and Δ_a, to 0 and 10 seconds. The setting is the same for all five tasks. This models the fact that typically each step is first described verbally and then performed on the camera. We set $\lambda = 1/(NK_{\text{pred}})$ for all methods that use (8). See Section 5.5 for an analysis of the effects of these hyperparameters.

Baselines. We compare results to four baselines. To demonstrate the difficulty of our dataset, we first evaluate a "Uniform" baseline, which simply distributes instructions steps uniformly over the entire instruction video. The second baseline "Video only" [5] does not use the narration and performs only discriminative clustering on visual features with a global order constraint. In particular, we compare to the improved model from [6], which does not require a "background class" and yields a stronger baseline equivalent to our model (7) *without* the weak textual constraints. The third baseline "Video + BOW dobj" adds text-based features to the "Video only" baseline (by concatenating the text and video features in the discriminative clustering approach). Here the goal is to evaluate the benefits of our two-stage clustering approach, in contrast to this single-stage clustering baseline. The text features are bag-of-words histograms over a fixed vocabulary of direct object relations. An alternative set-up creating separate bag-of-words histograms for nouns and verbs gave similar results and is not considered here for brevity. The fourth baseline is our own implementation of the alignment method of [24] (without the supervised vision refinement procedure that requires a set of pre-trained visual classifiers that are not available a-priori in our case). We use [24] to re-align the speech transcripts to the sequence of steps discovered by our method from Section 4.1 (as a proxy for the recipe assumed to be known in [24]).[9] To assess the difficulty of the task and dataset, we also compare results with a "Supervised" approach. For that, we divide the N input videos in 5 different folds. One fold is kept for the test set while the the other 4 are used as the trainining/validation dataset. With the 4 remaining folds, we perform a 4-fold cross validation in order to choose the hyperparameter λ. Once the hyper parameter is fixed, we retrain a model on the 4 folds and evaluate it on the test set. By iterating over the five possible test folds, we report variation in performance with error bars. The supervised method learns classifiers W for

[9]Note that our method finds at the same time the sequence of steps (a recipe in [24]) and the alignment of the transcripts.



(a) Change tire (**11**) (b) Perform CPR (**7**) (c) Repot plant (**7**) (d) Make coffee (**10**) (e) Jump car (**12**)

Fig. 5: Results for temporally localizing recovered steps in the input videos. We give in **bold** the number of ground truth steps.

all the visual steps. This is achieved by minimizing the cost defined in (7) under the ground truth annotation constraints on the training set. At test time, we simply apply classifiers W on the test set performing least-square prediction of Z_{test} with the ordering constraints.

Error bars for unsupervised methods. For the unsupervised methods we compute the error bars in the following manner. Recall that the Frank-Wolfe algorithm solves a continuous relaxation of the integer problem (7). To obtain an integer solution, we round the continuous solution using the rounding method described in Section 4.2. This rounding procedure is performed at each iteration of the optimization. When the stopping criterion of the Frank-Wolfe algorithm is reached (fixed number of iterations or target sub-optimality in practice), we have as many rounded solutions as the number of iterations. Our output integer solution is the rounded solution that achieves the lowest objective. Note that we are only guaranteed to lower the objective in the continuous domain whereas the objective of the rounded solution can increase (and often does), and hence there is no guarantee that the last rounded solution will have the lowest objective. In order to illustrate the variation of the performance during optimization, we define error bars as the interval determined by the minimal and maximal performance (measured by the F1 score) obtained *after* visiting the best rounded point (the integer solution with the lowest objective). This also explains why the error bars in Figure 5 are not necessarily symmetric. Overall, the observed variation is not very high, highlighting the stability of the optimization procedure.

Results. Results for localizing the discovered instruction steps are shown in Figure 5. In order to perform a fair comparison to the baseline methods that require a known number of steps K, we report results for a range of K values. Note that in our case the actual number of automatically recovered steps can be (and often is) smaller than K. First, we note that for each task there is a value of K for which our method outperforms the baselines. We discuss the results for the different tasks next. For *Change tire* and *Perform CPR*, our method consistently outperforms all baselines for all values of K demonstrating the benefits of our approach. For *Repot*, our method is comparable to text-based baselines, underlying the importance of the text signal for this problem. For *Jump car*, our method delivers the best result (for $K = 15$) but struggles for lower values of K, which we found was due to visually similar repeating steps (e.g. start car A and start car B) which are mixed-up for lower values of K. For the *Make coffee* task, the video only baseline is comparable to our method, which by inspecting the output could be attributed to large variability of narrations for this task. We also report results using the word2vec text similarity metric (see 'Our method (w2v)' in Figure 5) instead of the WordNet metric used in the first stage of our algorithm. We observe that both metrics result in a similar performance. Qualitative results of the recovered steps are illustrated in Figure 6.

5.5 Parameter analysis

In this section, we present a parameter sensitivity analysis divided into two main experiments: (i) in Section 5.5.1 we evaluate the sensitivity of our method to the choice of hyperparameter λ, which controls the strength of the regularization, and hyperparameters (Δ_b, Δ_a), which control the temporal caption-video alignment; (ii) in Section 5.5.2 we analyze the two main factors that influence the performance of the method, i.e. the number of constraints that are extracted from the textual narration and their temporal localization.

5.5.1 Hyperparameter analysis

Our method has two main hyperparameters. The first one, denoted λ, controls the amount of regularization of the estimated classifier parameters W in the discriminative loss (8). We use a single fixed setting of λ as our method is unsupervised and has no access to ground truth annotations for cross-validation at training time. Hence in all our experiments we set $\lambda = 1/(NK_{\text{pred}})$ which is equal to the inverse of the number of predicted latent variables. This choice is motivated by the form of the ridge regression generalization bound discussed in [15]. The second set of hyperparameters (Δ_b, Δ_a) adjusts the temporal alignment of transcribed narrations to the video. In particular, it allows us to model the fact that people usually perform the action after having talked about it. In our work we set the value of Δ_b (the 'before' delay) to 0 seconds and of Δ_a (the 'after' delay) to 12 seconds. In the following we quantify the effect of these hyperparameters on the final performance.

In Figure 7a, we report the F1 score averaged over the five tasks as a function of λ. As explained in Section 5.4, the error bars (shown by the shaded area) represent the F1 score variation during optimization. For this experiment, (Δ_b, Δ_a) is set to $(0\text{ s}, 12\text{ s})$ as in Section 5.4. We observe that, (i) a good choice of λ can result in up to 5% improvement on average across all five tasks, and (ii) the right range of values is located around $\lambda = 1/(NK_{\text{pred}})$ (see red line in Figure 7a). We report here the average performance but we have observed that the best setting of λ varies across different tasks. For example for the *performing cardiopulmonary resuscitation (CPR)* task, the good range of values for λ was a bit higher ($\approx 10^{-1}$) than for the other four tasks ($\approx 10^{-4} - 10^{-3}$). If a small annotated set is available, λ can be tuned for each task using cross-validation. If not, then $\lambda = 1/(NK_{\text{pred}})$ presents a reasonable choice.

In Figure 7b, we report the average F1 score over the five tasks as a function of (Δ_b, Δ_a). For this experiment, we set

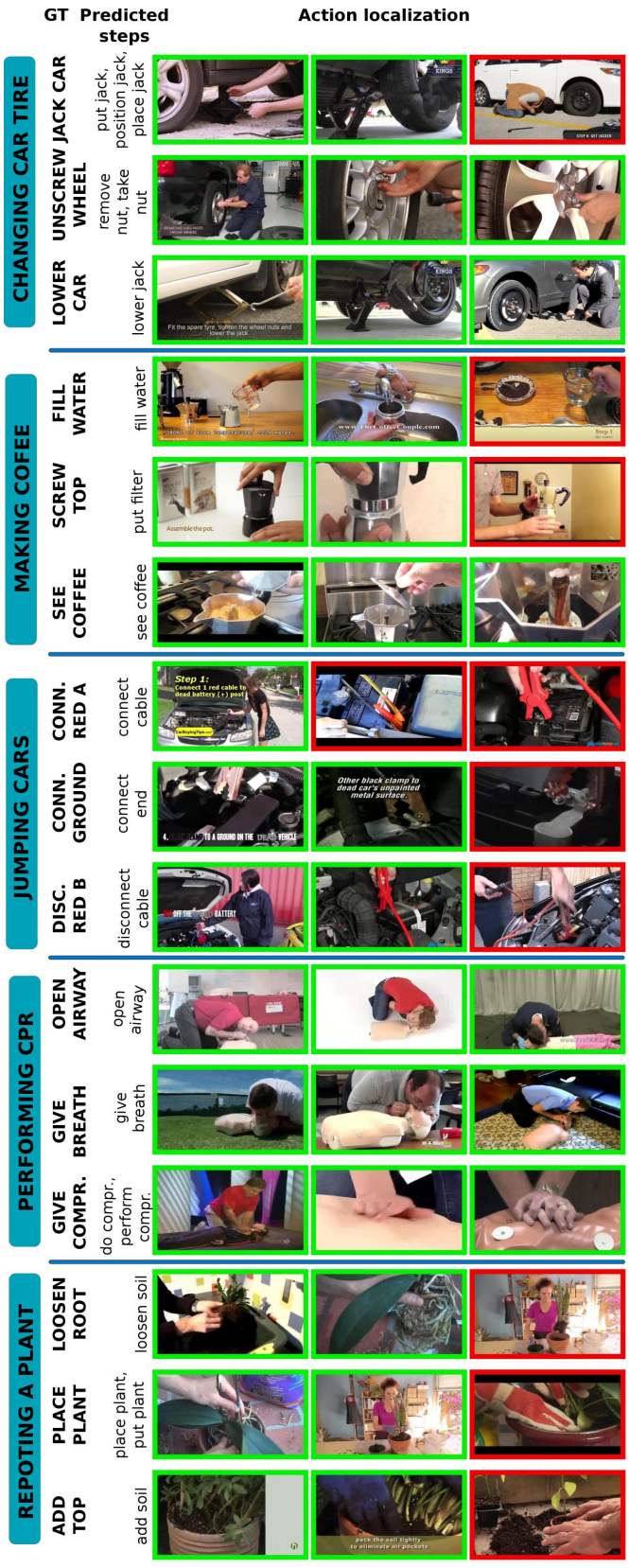
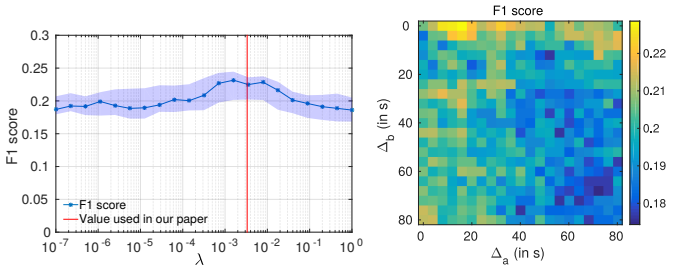


Fig. 6: **Examples of three recovered instruction steps for each of the five tasks in our dataset.** For each step, we first show clustered direct object relations, followed by representative example frames localizing the step in the videos. Correct localizations are shown in green. Some steps are incorrectly localized in some videos (red), but often look visually very similar.



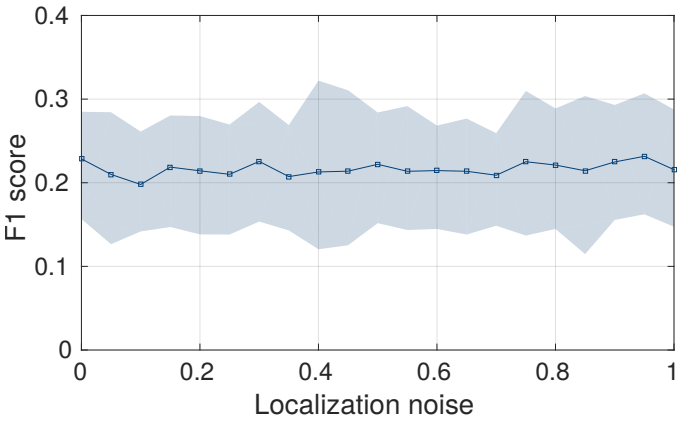
(a) Regularization strength λ (b) Subtitle delays (Δ_b, Δ_a)

Fig. 7: Sensitivity of the performance (measured by the F1 score) to the different hyperparameters.

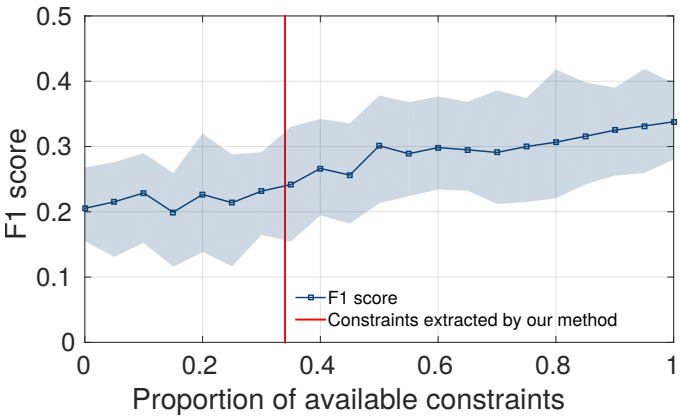
$\lambda = 1/(NK_{\text{pred}})$. For very large $(\Delta_b, \Delta_a) > (60, 60)$, the method is almost equivalent to the video only baseline (with a drop in performance of more than 5%) as the transcribed narration provides only very weak constraints on the temporal localization of the different steps. The good range for these hyperparameters is asymmetric, centered around $\Delta_a \approx 15$ s and $\Delta_b \approx 0$ s. This confirms the hypothesis that usually people first orally introduce a step and only then perform the action.

5.5.2 *Analysis of constraints from transcribed narrations*

The overall performance of our method relies on the amount and quality of constraints extracted from transcribed narrations. In this section we quantify this dependence experimentally. We perform two experiments.



(a) Influence of precise time localization of each constraint.



(b) Influence of proportion of extracted constraints.

Fig. 8: Analysis of constraints extracted from transcribed narrations.

First, we assess the importance of a precise time localization of each constraint. This type of mistake often occurs as the narration is usually not precisely aligned with the video, i.e. the speaker often first talks about the action before performing it. Figure 8a shows how the overall performance (measured by average F1 score on all tasks) varies with respect to the localization noise of extracted constraints. The set of constraints is fixed to those discovered automatically by our method. When the localization noise is 0, the temporal extent of each constraint obtained from the narration is corrected to the manually obtained temporal extent of the action performed in the video. When the localization noise is 1 constraints correspond to those extracted automatically by our method. In between values are obtained by interpolating the temporal extent of each constraint between these two extreme values. The average performance is obtained by 5-fold cross validation. The error bars correspond to minimum and maximum performance across the five folds. Interestingly, the results do not show a drop in performance with increased localization noise in the constraints. This suggests that our method is tolerant to some amount of localization noise and the constraints extracted by our method are localized reasonably well.

Next, in Figure 8b we study the influence of the amount of extracted constraints. In particular, we vary the proportion of extracted constraints from 0 (equivalent to the video only setting) to 1 (extracted constraint for each step performed in video). Note that we only use constraints that respect the ground truth global ordering of steps. To remove the effect of temporal localization, we use the manually corrected temporal extent for each constraint corresponding to the zero localization noise in the first experiment above. The graph shows average performance and error bars (shaded region) obtained by 5-fold cross validation as described in the first experiment above. Our automatic method using multiple sequence alignment extracts from the transcribed narration about 35% of constraints (red vertical bar). The results suggest that increasing the number of extracted constraints can significantly improve the overall performance. Hence, designing better language models that could discover more constraints is an important direction for future research.

5.6 Joint clustering of video and text

Recall that the proposed approach is formulated as two clustering tasks, one in text and one in video, applied in sequence and linked by joint constraints on the two modalities. To further investigate the complementary nature of the two signals, we experiment here with performing another iteration of our algorithm with the aim of transferring information from the visual to the textual domain. We proceed as follows. Once we obtain our temporal predictions in videos (latent variable Z), we use the inferred Z to adjust the values in the cost alignment matrix C_o in multiple sequence alignment of text sequences. More precisely, we match expressions if their similarity is high (but with a slightly lower threshold than in the first iteration) *and* if they were mentioned at least two times within the predicted time interval of the same step in different videos. We run the multiple sequence alignment with the new cost matrix C_o and use this solution to perform another iteration of the video alignment procedure. This additional iteration results in an average improvement of about 1% in the F1-score across all tasks.

6 Conclusion and Future Work

We have described a method to automatically discover the main steps of a task from a set of narrated instruction videos in an unsupervised manner. The proposed approach has been tested on a new annotated dataset of challenging real-world instruction videos containing complex person-object interactions in a variety of indoor and outdoor scenes. Our work opens up the possibility for large scale learning from instruction videos on the Internet. Our model currently assumes the existence of a common script with a fixed ordering of the main steps. While this assumption is often true, e.g. one cannot remove the wheel before jacking up the car, or make coffee before filling the water, some tasks can be performed while swapping (or even leaving out) some of the steps. Recovering more complex temporal structures is an interesting direction for future work.

Acknowledgments This work has been supported in part by ERC grants ACTIVIA (no. 307574) and LEAP (no. 336845), CIFAR Learning in Machines & Brains program, ESIF, OP Research, development and education Project IMPACT No. CZ.02.1.01/0.0/0.0/15_003/0000468 and a Google Research Award and the Google Research Award.

References

[1] Partial order alignment code for multiple sequence alignment. http://sourceforge.net/projects/poamsa/. 9
[2] Project webpage (code/dataset). http://www.di.ens.fr/willow/research/instructionvideos/. 3, 9
[3] F. Bach and Z. Harchaoui. DIFFRAC: A discriminative and flexible framework for clustering. In *NIPS*, 2007. 6, 7
[4] P. Bojanowski, F. Bach, I. Laptev, J. Ponce, C. Schmid, and J. Sivic. Finding actors and actions in movies. In *ICCV*, 2013. 2
[5] P. Bojanowski, R. Lajugie, F. Bach, I. Laptev, J. Ponce, C. Schmid, and J. Sivic. Weakly supervised action labeling in videos under ordering constraints. In *ECCV*, 2014. 2, 6, 8, 10
[6] P. Bojanowski, R. Lajugie, E. Grave, F. Bach, I. Laptev, J. Ponce, and C. Schmid. Weakly-supervised alignment of video with text. In *ICCV*, 2015. 2, 6, 8, 10
[7] N. Chambers and D. Jurafsky. Unsupervised learning of narrative event chains. In *ACL*, 2008. 1, 2
[8] V. Chari, S. Lacoste-Julien, I. Laptev, and J. Sivic. On pairwise costs for network flow multi-object tracking. In *CVPR*, 2015. 6
[9] M. Cimpoi, S. Maji, and A. Vedaldi. Deep filter banks for texture recognition and segmentation. In *CVPR*, 2015. 9
[10] M.-C. de Marneffe, B. MacCartney, and C. D. Manning. Generating typed dependency parses from phrase structure parses. In *LREC*, 2006. 4, 9
[11] O. Duchenne, I. Laptev, J. Sivic, F. Bach, and J. Ponce. Automatic annotation of human actions in video. In *ICCV*, 2009. 2
[12] C. Fellbaum. Wordnet: An electronic lexical database. *Cambridge, MA: MIT Press.*, 1998. 5
[13] L. Frermann, I. Titov, and M. Pinkal. A hierarchical Bayesian model for unsupervised induction of script knowledge. In *EACL*, 2014. 1, 2
[14] D. G. Higgins and P. M. Sharp. Clustal: A package for performing multiple sequence alignment on a microcomputer. *Gene*, 1988. 6, 9
[15] D. Hsu, S. M. Kakade, and T. Zhang. Random design analysis of ridge regression. *Foundations of Computational Mathematics*, 14(3), 2014. 11
[16] M. Jaggi. Revisiting Frank-Wolfe: Projection-free sparse convex optimization. In *ICML*, 2013. 5, 6
[17] A. Joulin, K. Tang, and L. Fei-Fei. Efficient image and video co-localization with Frank-Wolfe algorithm. In *ECCV*, 2014. 6
[18] S. Lacoste-Julien. Convergence rate of Frank-Wolfe for non-convex objectives. *arXiv preprint*, 2016. 6
[19] S. Lacoste-Julien and M. Jaggi. On the global linear convergence of Frank-Wolfe optimization variants. In *NIPS*, 2015. 6
[20] S. Lacoste-Julien, M. Jaggi, M. Schmidt, and P. Pletscher. Block-coordinate Frank-Wolfe optimization for structural SVMs. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2013. 9
[21] I. Laptev, M. Marszalek, C. Schmid, and B. Rozenfeld. Learning realistic human actions from movies. In *CVPR*, 2008. 2
[22] C. Lee, C. Grasso, and M. Sharlow. Multiple sequence alignment using partial order graphs. *Bioinformatics*, 2002. 5, 6, 9, 10
[23] T. Liao. Clustering of time series data, a survey. *Pattern recognition*, 2014. 10

[24] J. Malmaud, J. Huang, V. Rathod, N. Johnston, A. Rabinovich, and K. Murphy. What's cookin'? Interpreting cooking videos using text, speech and vision. In *NAACL*, 2015. 1, 2, 10

[25] A. Miech, J.-B. Alayrac, P. Bojanowski, I. Laptev, and J. Sivic. Learning from video and text via large-scale discriminative clustering. In *ICCV*, 2017. 9

[26] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *NIPS*, 2013. 9

[27] G. A. Miller. Wordnet: A lexical database for english. *Communications of the ACM*, 1995. 5

[28] I. Naim, Y. Chol Song, Q. Liu, L. Huang, H. Kautz, J. Luo, and D. Gildea. Discriminative unsupervised alignment of natural language instructions with corresponding video segments. In *NAACL*, 2015. 2

[29] J. C. Niebles, C.-W. Chen, and L. Fei-Fei. Modeling temporal structure of decomposable motion segments for activity classification. In *ECCV*, 2010. 2

[30] J. C. Niebles, H. Wang, and L. Fei-Fei. Unsupervised learning of human action categories using spatial-temporal words. *IJCV*, 2008. 2

[31] A. Osokin, J.-B. Alayrac, I. Lukasewitz, P. K. Dokania, and S. Lacoste-Julien. Minding the gaps for block Frank-Wolfe optimization of structured SVMs. In *Proceedings of The 33rd International Conference of Machine Learning (ICML)*, 2016. 9

[32] D. Potapov, M. Douze, Z. Harchaoui, and C. Schmid. Category-specific video summarization. In *ECCV*, 2014. 2

[33] M. Raptis and L. Sigal. Poselet key-framing: A model for human activity recognition. In *CVPR*, 2013. 2

[34] M. Regneri, A. Koller, and M. Pinkal. Learning script knowledge with Web experiments. In *ACL*, 2010. 1, 2, 4

[35] O. Sener, A. Zamir, S. Savarese, and A. Saxena. Unsupervised semantic parsing of video collections. In *ICCV*, 2015. 2

[36] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015. 9

[37] M. Sun, A. Farhadi, and S. Seitz. Ranking domain-specific highlights by analyzing edited videos. In *ECCV*, 2014. 2

[38] H. Wang and C. Schmid. Action recognition with improved trajectories. In *ICCV*, 2013. 9

[39] L. Wang and T. Jiang. On the complexity of multiple sequence alignment. *Journal of computational biology*, 1(4):337–348, 1994. 4, 5, 9

Jean-Baptiste Alayrac received the MS degree in computer science in Ecole Normale Supérieure (ENS), in Paris in 2014. He is currently working toward the PhD degree in the research teams WILLOW and SIERRA at INRIA Paris under the supervision of Josef Sivic, Ivan Laptev and Simon Lacoste-Julien. His research focuses on structured prediction from vision and natural language.

Piotr Bojanowski Piotr Bojanowski is a Post Doctoral Researcher at Facebook AI Research. He graduated from a Ph.D. at Willow team at INRIA Paris in 2016 where he was supervised by Jean Ponce, Cordelia Schmid, Ivan Laptev and Josef Sivic. His work focuses on automatic video and image understanding.

Nishant Agrawal Nishant Agrawal received his Bachelors of Technology in Computer Science with an Honors in Computer Vision from IIIT Hyderabad in 2016. He is currently a graduate student at Carnegie Mellon University pursuing his MS in Computer Vision and is expected to graduate in December of 2017.

Ivan Laptev Ivan Laptev is a research director at INRIA Paris, France. He received a Habilitation degree from Ecole Normale Supérieure in 2013 and a PhD degree in Computer Science from the Royal Institute of Technology in 2004. Ivan's main research interests include visual recognition of human actions, objects and interactions. He has published over 50 papers at international conferences and journals of computer vision and machine learning. He serves as an associate editor of IJCV and TPAMI journals, he will serve as a program chair for CVPR18, he was/is an area chair for major computer vision conferences, he has co-organized several tutorials, workshops and challenges at major computer vision conferences. He has also co-organized a series of INRIA summer schools on computer vision and machine learning (2010-2013). He received ERC Starting Grant in 2012.

Josef Sivic received the MS degree from the Czech Technical University in Prague and PhD from the University of Oxford. He currently holds a permanent position as an INRIA senior researcher in the Department of Computer Science at the Ecole Normale Supérieure (ENS) in Paris. His interests lie in developing learnable image representations for automatic visual search and recognition applied to large image and video collections. He has published more than 60 scientific publications, has served as an area chair for major computer vision conferences and as a program chair for ICCV'15. He currently serves as an associate editor for the International Journal of Computer Vision and IEEE Transactions on Pattern Analysis and Machine Intelligence. He was awarded an ERC grant in 2013.

Simon Lacoste-Julien is currently an assistant professor at the Université de Montréal. Before that, he was an INRIA researcher in the Department of Computer Science at the Ecole Normale Supérieure (ENS) in Paris. He received his PhD from the University of California, Berkeley in 2009. His research interests are in machine learning, optimization and statistics with applications in natural language processing, computer vision and information retrieval. He has published more than 20 scientific publications in machine learning and has served as an area chair for UAI, ICML and NIPS. He received a Google Focused Research Award in 2016.