論文 / 著書情報
Article / Book Information

Title	Action Sequence Recognition in Videos by Combining a CTC Network with a Statistical Language Model
Authors	Mengxi Lin, Nakamasa Inoue, Koichi Shinoda
Citation	Technical Reports of IEICE PRMU, vol. 117, no. 362, pp. 1-6
Pub. date	2017, 12
Copyright	(c) 2017 IEICE

社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
TECHNICAL REPORT OF IEICE.

Action Sequence Recognition in Videos by Combining a CTC Network with a Statistical Language Model

Mengxi LIN†, Nakamasa INOUE†, and Koichi SHINODA†

† Department of Computer Science, Tokyo Institute of Technology,
2-12-1 W8-81, Ookayama, Meguro-ku, Tokyo, 152–8552 Japan
E-mail: †{mengxi,inoue}@ks.cs.titech.ac.jp, ††shinoda@cs.titech.ac.jp

Abstract Action sequence recognition aims to recognize what actions occur in a video and their temporal order. In this paper, we propose to combine an LSTM network trained with Connectionist Temporal Classification (CTC) with a statistical language model for action sequence recognition. The statistical language model captures the relations between action instances, which are hardly learned by the CTC network. Our experiments on the Breakfast dataset show that the statistical language model can significantly boost the recognition accuracy of the CTC network, from 37.0% to 43.4%.

Key words connectionist temporal classification; action sequence recognition; statistical language model; weakly supervised learning

1. Introduction

Researchers in computer vision have developed many effective algorithms for human action understanding in videos since the past few decades [1], [2]. Regardless of their great success, most of the algorithms assume that a video clip contains only one action during training, and require us to take laborious effort to annotate the boundaries between different actions as well as between actions and non-action in videos. It therefore refrains us from leveraging large-scaled video resources to build up more robust recognition systems. Besides, the boundary annotation is error-prone [3], making the situation even harder.

To solve this problem, several approaches [4], [5] have been proposed to recognize an action sequence that consists of more than one action, where no temporal boundaries about actions are given both in the training and test phases. Since annotating temporal order of actions is much easier, it paves a way for utilizing larger amounts of data for model learning.

In previous work [4], [6], [7], Recurrent Neural Network (RNN) trained with Connectionist Temporal Classification (CTC) [8] has shown its effectiveness on modeling video data when only given temporal order supervision. CTC enables the RNN to learn the mapping between a video and its target sequence in an end-to-end manner. The Extended Connectionist Temporal Classification (ECTC) method [4] applies CTC to action segmentation and action sequence recognition. It takes into account the visual similarity between frames to guide the CTC training.

One major problem of the CTC framework for action sequence recognition is that it hardly captures the relationships between actions, in spite of the fact that the relationships are strongly presented in action sequences; e.g. the action *pourWater* is more likely to take place than *pourCoffee* after *addTeabag*. The reason is that the RNN on which CTC is applied still has difficulty in encoding long-range information to cover enough context for capturing the action relations, even with a sophisticated memory cell design like in LSTM. Some evidence given in Singh et al.'s work [9] has showed that a typical Bi-LSTM can only remember 8 seconds' information on average as context. However, the temporal interval between actions is usually longer than that. For instance, the action *pourDough* may happen after the action *stirDough* that takes about 2 *mins* to complete. This imposes difficulty for the RNN to capture the relationship between *pourDough* and *stirDough*, as well as the relation between *pourDough* and *pourMilk* that may precede *stirDough*, since it requires the RNN to remember at least 2 *mins* context.

Some work based on the hierarchical temporal pooling on RNN [11] may alleviate the long-range dependency learning problem to some extent by constructing hierarchical temporal representation. However, the learned temporal representation is difficult to generalize for the long-tailed relationships that have few or even zero examples in the training.

To address these difficulties, we propose to model the relationships between actions explicitly using a statistical n-

gram language model, and combine it with a CTC-trained Bi-LSTM for action sequence recognition [10]. While the statistical n-gram language model can be easily trained from action sequences, it can well capture the distribution of action sequences. When equipped with the backoff smoothing technique [12], it can also help the network to generalize to the action relations that are rarely or never seen in the training.

Compared with our previous published paper [10], this paper investigates more how effective our method is when generalizing to unseen action sequences in the experiments.

This paper is organized as follows. Section 2 reviews some of the related work. After introducing CTC and statistical n-gram language models in Section 3 and Section 4 respectively, we present our approach in Section 5. Finally, we show our experimental results and analysis in Section 6, followed by our conclusion in Section 7.

2. Related Work

2. 1 Action Learning from Pre-segmented Videos.

Many studies have been devoted to the recognition of human actions in pre-segmented videos. Based on the expert understanding about human action patterns, researchers have developed many sophisticated hand-crafted spatio-temporal features [1]. Recently, deep visual representations learned from deep neural networks [2] have drawn much attention and achieved the state-of-the-art performance in human action recognition.

Besides the achievements got by creating advanced feature representations, some studies showed the effectiveness of temporal modeling in action recognition, by applying temporal templates [13], hidden markov models [14], temporal convolution [15] and recurrent neural networks [9]. Some prior work [16] also attempted to model the temporal dependency of action labels.

2. 2 Action Learning from Unsegmented Videos.

The approaches focusing on learning from unsegmented videos are relatively few. They rely on the annotations of action temporal order (i.e. action sequence) in a video as supervision [4], [5], [17], [18]. Bojanowski et al. [17] formulated the problem as a discriminative clustering that clustered video frames under the temporal order constrains, and simultaneously learn a frame-wise classifier. Kuehne et al. [5] were inspired by the success of the Hidden Markov Model (HMM) applied in speech recognition without phoneme boundary annotations, and evaluated it on video data for action segmentation.

Recently, Recurrent Neural Network (RNN) became popular, for its power in modeling temporal dynamics and learning feature representation. Richard et al. [18] proposed to combine HMM and RNN for the temporal order supervision setting, training both of them by an iterative procedure that runs between optimizing the model parameters and realigning video frames with an action sequence. Huang et al. [4] applied the idea of Connectionist Temporal Classification [8], which was able to train an RNN in an end-to-end manner to map a source sequence directly to a target sequence. They also adapted the CTC framework to take into account the visual similarity between frames.

2. 3 Statistical Language Model

Researchers have developed many kinds of statistical language models, which can model the probability distribution of sequences. N-gram statistical language model is the most widely used one. It makes Markov assumption and assigns the probability of a word according to its preceding $(n-1)$ words. In some cases, word subsequences may have zero count in the training set, leading to their probability being under-estimated to zero. To amend it, several smoothing methods have been proposed [12], [19]〜[21]. Recently, Richard et al. [16] applied the n-gram language model to model the relationships between action labels for assisting action detection. However, their framework relies on pre-segmented videos.

Besides the n-gram language model, several other statistical language models were also proposed [22], [23]. Despite their relative success in modeling language sequences, they have been found to give only moderate improvement over the n-gram language model while introducing more complexity for construction [24].

3. Connectionist Temporal Classification

Let $\boldsymbol{X} = (\boldsymbol{x}^1, \boldsymbol{x}^2, ..., \boldsymbol{x}^T)$ denote a sequence of activations of a video output by an RNN, and $\boldsymbol{y} = (y^1, y^2, ..., y^N)$ denote its action sequence, where the number of the activations is not smaller than the number of the actions, i.e. $T \geqq N$. CTC defines the probability of an action sequence $\boldsymbol{y}$ given the activations $\boldsymbol{X}$:

$$p(\boldsymbol{y}|\boldsymbol{X}) = \sum_{\boldsymbol{\pi} \in \{\boldsymbol{\pi}:\mathfrak{R}(\boldsymbol{\pi})=\boldsymbol{y}\}} p(\boldsymbol{\pi}|\boldsymbol{X}), \tag{1}$$

where $\boldsymbol{\pi} = (\pi^1, \pi^2, ..., \pi^T)$ is a path denoting the emission of action labels along $\boldsymbol{X}$, and $\mathfrak{R}$ is an operator to remove consecutively repeated action labels in $\boldsymbol{\pi}$, e.g. both the action paths $[A,A,B,C]$ and $[A,B,C,C]$ are mapped to the same action sequence $[A,B,C]$. CTC makes an assumption that each π^t in $\boldsymbol{\pi}$ is conditionally independent given $\boldsymbol{X}$:

$$p(\boldsymbol{\pi}|\boldsymbol{X}) = \prod_{t=1}^{T} p(\pi^t|\boldsymbol{X}). \tag{2}$$

The value of $p(\pi^t|\boldsymbol{X})$ is given by the action class soft-max output of the activation at time t.

Graves et al. [8] proposed an efficient dynamic programming method to compute Eq. (1). Specifically, let $\boldsymbol{\pi}_1^t$ denote the partial path of $\boldsymbol{\pi}$ from time 1 to t, and $\boldsymbol{y}_1^k$ denote the partial action label sequence composed by the first k labels in order from $\boldsymbol{y}$. They defined a variable $\alpha^t(k)$, which is the sum of the probabilities of $\boldsymbol{\pi}_1^t$ that can be aligned to $\boldsymbol{y}_1^k$:

$$\alpha^t(k) = \sum_{\boldsymbol{\pi} \in \{\boldsymbol{\pi}: \Re(\boldsymbol{\pi}_1^t) = \boldsymbol{y}_1^k\}} p(\boldsymbol{\pi}_1^t | \boldsymbol{X}). \tag{3}$$

By taking the conditional independence assumption in Eq. (2), we can derive $\alpha^t(k)$ recursively:

$$\alpha^t(k) = [\alpha^{t-1}(k-1) + \alpha^{t-1}(k)] s^t(y^k), \tag{4}$$

where $s^t(y^k)$ is the probability of emitting the label y^k at time t. Intuitively, Eq. (4) defines the mapping of $\boldsymbol{X}$ to $\boldsymbol{y}$ up to time t and the k-th label, allowing only the transitions from the $(k-1)$-th and the k-th labels at time $t-1$.

By dynamic programming, CTC is capable to compute the probability $p(\boldsymbol{y}|\boldsymbol{X})$ by deriving $\alpha^T(N)$ in linear time with respect to the length of the video. Using the probability $p(\boldsymbol{y}|\boldsymbol{X})$, CTC defines the loss for an RNN as:

$$L = -\log p(\boldsymbol{y}|\boldsymbol{X}). \tag{5}$$

Since the loss function is differentiable, we can use backpropagation to optimize the network.

Generally, the output of a CTC network can be decoded with the best path [4], [8], i.e. applying $\Re(\cdot)$ to the path that consists of the label with the largest probability at each time step.

4. Statistical N-gram Language Model

Let $\boldsymbol{y} = (y^1, y^2, .., y^N)$ denote a sequence of length N. A statistical n-gram language model can be described as follow:

$$p(\boldsymbol{y}) = \prod_{k=1}^{N} p(y^k | \boldsymbol{y}_1^{k-1}) = \prod_{k=1}^{N} p(y^k | \boldsymbol{y}_{k-n+1}^{k-1}), \tag{6}$$

where y^k denotes the k-th label in $\boldsymbol{y}$ and $\boldsymbol{y}_l^k$ denotes a subsequence composed by the l-th to k-th labels in order from $\boldsymbol{y}$. In the n-gram language model, we call a subsequence of length n an *n-gram*. The probability of emitting a label in a sequence is assumed to be only dependent on its $n-1$ preceding labels. The Maximum Likelihood (ML) estimation of the probability $p(y^k | \boldsymbol{y}_{k-n+1}^{k-1})$ is given by:

$$p(y^k | \boldsymbol{y}_{k-n+1}^{k-1}) = \frac{C(\boldsymbol{y}_{k-n+1}^{k})}{C(\boldsymbol{y}_{k-n+1}^{k-1})}, \tag{7}$$

where $C(\cdot)$ denotes the counts of subsequences appearing in the training set.

Since the training set is usually too small to give robust ML estimation, and under-estimates the probability to zero when an n-gram is unseen in the training set, some of the methods [12], [19] are proposed to smooth the language model. We adopt the Katz backoff [12] to perform smoothing, which has been shown effective in an empirical study conducted by Chen et al. [25]. The basic idea behind the Katz backoff model is to steal some probability mass from the n-gram observed in the training data and re-distribute it to the unseen ones by referring to the lower order n-grams:

$$p^*(y^k | \boldsymbol{y}_{k-n+1}^{k-1}) = \begin{cases} \tilde{p}(y^k | \boldsymbol{y}_{k-n+1}^{k-1}), & \text{if } C(\boldsymbol{y}_{k-n+1}^{k}) > 0, \\ \gamma(\boldsymbol{y}_{k-n+1}^{k-1}) p^*(y^k | \boldsymbol{y}_{k-n+2}^{k-1}), & \text{otherwise.} \end{cases} \tag{8}$$

where $p^*(y^k | \boldsymbol{y}_{k-n+1}^{k-1})$ is the smoothed estimation of an n-gram probability, $\tilde{p}(y^k | \boldsymbol{y}_{k-n+1}^{k-1})$ is the discounted probability of an n-gram that can be observed in the training, $\gamma(\boldsymbol{y}_{k-n+1}^{k-1})$ is the renormalization factor that guarantees the smoothed conditional probabilities sum up to one. For $\tilde{p}(y^k | y_{k-n+1}^{k-1})$, we applied the Witten-Bell discounting method [21], which takes into account the diversity of predicted labels followed a preceding context. It additionally regards the first occurrences of the n-grams as the occurrences of unseen n-grams, leading to:

$$\tilde{p}(y^k | \boldsymbol{y}_{k-n+1}^{k-1}) = \frac{C(\boldsymbol{y}_{k-n+1}^{k})}{C(\boldsymbol{y}_{k-n+1}^{k-1}) + \delta(\boldsymbol{y}_{k-n+1}^{k-1})}, \tag{9}$$

where $\delta(\boldsymbol{y}_{k-n+1}^{k-1})$ denotes the number of label types with the preceding context $\boldsymbol{y}_{k-n+1}^{k-1}$ observed in the training data.

5. Our Approach

5.1 Framework

Fig. 1 shows the framework of our approach when recognizing an action sequence in an unsegmented video. It consists of four components, namely feature extraction, CTC network, beam search decoder and statistical n-gram language model. In recognition stage, feature extraction module extracts appearance and motion features for each frame. The extracted features are fed into the CTC network, which is a Bi-LSTM trained with the CTC loss, to model the temporal structures of the video. Meanwhile, a statistical n-gram language model is learned from the action sequences annotation to capture action relationships. Finally, the beam search decoder combines the knowledge from the network and the statistical n-gram language model to derive the most probable action sequence. Our main contribution is to combine the CTC network and the n-gram statistical language model of actions for joint reasoning.

5.2 Combining CTC Network with N-gram Statistical Language Model

Basically, we want to find the action sequence that maximizes the posterior probability:

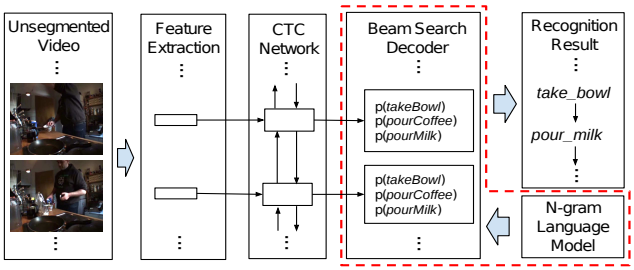

Fig. 1: The framework of our approach: Our framework extracts frame-wise features from an unsegmented video, which are fed into a CTC network to produce action probability at each frame. Then a beam search decoder will take the probability outputs and combine the knowledge from an n-gram statistical language model of actions to generate the recognition result.

$$\begin{aligned} \boldsymbol{y}^* &= \arg\max_{\boldsymbol{y}} p(\boldsymbol{y}|\boldsymbol{X}) \\ &= \arg\max_{\boldsymbol{y}} \frac{p(\boldsymbol{X}|\boldsymbol{y})p(\boldsymbol{y})}{p(\boldsymbol{X})} \\ &= \arg\max_{\boldsymbol{y}} p(\boldsymbol{X}|\boldsymbol{y})p(\boldsymbol{y}), \end{aligned} \tag{10}$$

where $\boldsymbol{y}$ and $\boldsymbol{X}$ denote an action sequence and the network encoding sequence of a video respectively. Here, Bayes Rule is applied to convert the posterior probability $p(\boldsymbol{y}|\boldsymbol{X})$ to the product of the conditional probability $p(\boldsymbol{X}|\boldsymbol{y})$ and the prior probability $p(\boldsymbol{y})$, normalized by an observation probability $p(\boldsymbol{X})$. The conditional probability $p(\boldsymbol{X}|\boldsymbol{y})$ can be interpreted as an *action model* that explains the video data given the action sequence, while the prior probability $p(\boldsymbol{y})$ is interpreted as a *relation model* that explains the relationships between actions. Since the observation probability $p(\boldsymbol{X})$ is independent of action sequences, we ignore it when doing inference.

Given that a CTC network is able to learn the nature of each element action, the output of the CTC network naturally provides a way to model the term $p(\boldsymbol{X}|\boldsymbol{y})$. However, since the CTC network directly estimates a posterior of an action sequence, i.e. $p(\boldsymbol{y}|\boldsymbol{X})$, we are required to convert it to the form of conditional probability, i.e. $p(\boldsymbol{X}|\boldsymbol{y})$, before plugging it into Eq. (10) as the *action model*. Specifically, we scale the CTC network output by the path label priors:

$$p(\boldsymbol{X}|\boldsymbol{y}) \propto \prod_{t=1}^{T} \frac{p(\pi^t|\boldsymbol{X})}{p(\pi^t)}, \tag{11}$$

where we apply the best path approximation for $\boldsymbol{y}$, and π^t is the t-th label in the best path of $\boldsymbol{X}$, whose length is T. Here $p(\pi^t|\boldsymbol{X})$ is the soft-max probability given by the CTC network and $p(\pi^t)$ is the path label prior, which can be estimated by forcedly aligning training videos and their corresponding action sequences using Viterbi algorithm, followed by simply counting, i.e. Maximum Likelihood (ML) estimation. CTC network tends to give very peaky label prediction, with the majority 'background' label dominating the best paths, leading to that the prior estimated by ML is rather skew and not robust. Therefore, we follow the suggestion of [26] to discount the number of 'background' labels and smooth the ML estimation for $p(\pi^t)$.

The *relation model* $p(\boldsymbol{y})$ in Eq. (10) is estimated by the n-gram statistical language model of actions and the prior about the length of an action sequence:

$$p(\boldsymbol{y}) \propto p_{\text{lm}}^{\alpha}(\boldsymbol{y}) N^{\beta}. \tag{12}$$

where $p_{\text{lm}}(\boldsymbol{y})$ is the statistical language model of actions, N is the length of $\boldsymbol{y}$, α and β are two tunable parameters. In addition to the n-gram statistical language model for modeling the action relationships, we propose to add the prior about the global length of an action sequence. This should complement the n-gram statistical language model, since the n-gram statistical language model lacks the knowledge about the global length of an action sequence.

Combining Eq. (11), Eq. (12) into Eq. (10), and taking the logarithm, we can get:

$$\boldsymbol{y}^* = \arg\max_{\boldsymbol{y}} E(\boldsymbol{y}|\boldsymbol{X}), \tag{13}$$

where

$$E(\boldsymbol{y}|\boldsymbol{X}) = \sum_{t=1}^{T} \log \frac{p(\pi^t|\boldsymbol{X})}{p(\pi^t)} + \alpha \log p_{\text{lm}}(\boldsymbol{y}) + \beta \log(N). \tag{14}$$

We refer $\sum_{t=1}^{T} \log \frac{p(\pi^t|\boldsymbol{X})}{p(\pi^t)}$ as *CTC term*, $\alpha \log p_{\text{lm}}(\boldsymbol{y})$ as *LM term* and $\beta \log(N)$ as *length term*. We adopt the beam search decoding mechanism as in [27] to find the optimal $\boldsymbol{y}$ in Eq. (14).

6. Experiments

6.1 Dataset and Evaluation

We evaluate our method on the *Breakfast* [14] dataset, which is a large-scaled video dataset that records realistic human actions of preparing breakfast. It consists of 1,712 video clips with totally 66.7 hours. Each clip records a person performing a goal-directed activity, such as *preparing cereal* and *preparing sandwiches*. The activities are further decomposed into several action units in temporal order. In total, there are 48 kinds of action units in the dataset, such as *takeCup*, *cutOrange*, etc. We use the training/testing splits of the dataset defined in [14].

We follow the evaluation protocol of [14] to evaluate our method under *unit accuracy*. Specifically, we match the recognized action sequence for a video against its groundtruth using Dynamic Time Warping (DTW). The matching using DTW results in three types of error, namely insertion error I, deletion error D and substitution error S. Then the *unit accuracy* is calculated as:

Table 1: Ablation study of our proposed method under *unit accuracy* on *Breakfast* dataset. CTC refers to the CTC baseline. Len refers to including the *length term* in Eq. (14). Bigram, Trigram and Quadgram refer to including the *LM term* in Eq. (14).

Model	**Unit Acc.**
CTC	0.370
CTC+Len	0.381
CTC+Bigram	0.391
CTC+Trigram	0.415
CTC+Quadgram	0.412
CTC+Bigram+Len	0.415
CTC+Trigram+Len	**0.434**
CTC+Quadgram+Len	0.429

Table 2: Performance with non-smoothed and smoothed trigram language model. *Known* consists of the action sequences seen in the training while *Unknown* consists of the ones never seen, with OOV rate of trigrams being 18.1%. *All* denotes the whole test set.

	Known	*UnKnown*	*All*
Non-smoothed	**0.523**	0.328	0.433
Smoothed	0.509	**0.346**	**0.434**

$$\text{Acc.}(\boldsymbol{y}^*, \boldsymbol{y}) = 1 - \frac{I + D + S}{N}, \tag{15}$$

where $\boldsymbol{y}^*$ denotes the recognized action sequence, $\boldsymbol{y}$ denotes the groundtruth and N denotes the length of the groundtruth.

6.2 Experimental Setup

We down-sample the videos from 15fps to 1.5fps for the sake of efficiency, and extract a Fisher Vector of Improved Dense Trajectory [1] for every frame in a way described in [28]. As a result, each frame is featured with a 64-dim descriptor.

We employ a one-layer Bi-LSTM with 256 hidden units for the CTC network. For the n-gram statistical language model, we explore the bigram, trigram as well as quadgram.

6.3 Experimental Results and Discussion

6.3.1 Ablation study

To investigate how effective our statistical language model is for action sequence recognition, we conducted an ablation study with the *unit accuracy* results shown in Table 1.

The first baseline (CTC) is merely a CTC network while the second baseline (CTC+Len) is combining the *CTC term* in Eq. (14) with the *length term*. It gave approximately 1% point improvement by incorporating a prior about the length of action sequences.

We also investigated the effect of the *LM term* in Eq. (14), by excluding the *length term*. Row 3 to Row 5 in Table 1 show the results when the *LM term* was given by a bigram, trigram and quadgram respectively. One can see that by incorporating a statistical language model it helped to boost the performance significantly. Among them, the trigram model gave the best result.

Fig. 2: Qualitative results on *Breakfast* dataset. Our method (CTC+Trigram+Len) is compared with the CTC baseline. The underscores denote deletion error and the red color on the action labels denotes substitution error.

Finally, we show the performance of the complete model in the last three rows in Table 1, where CTC+Trigram+Len achieved the best result (0.434). The improvement by adding both the *LM term* and *length term* demonstrates they are complementary to each other.

In Fig. 2, we show several qualitative results comparing our proposed CTC+Trigram+Len and the CTC baseline. From the results, we can see that our method was able to correct the errors of the CTC network that did not conform the general human action orders. For instance, *takePlate* was corrected to *pourDough2pan* in the first example. Moreover, our method helped to discover some actions that were not well recognized by the CTC network, e.g. *putPancake2plate* in the first example, *addSaltnpepper* in the second example and *takeGlass* in the third example.

6.3.2 Effectiveness of Smoothing

In this section, we investigate how effective the smoothing method is for generalizing to the unseen action relationships. We split the testing set into two mutually exclusive sets, namely *Known* and *Unknown*. The *Known* consists of action sequences which can be observed in the training while the *Unknown* consists of the ones never seen, with the Out-Of-Vocabulary (OOV) rate of trigrams being 18.1%. We performed action sequence recognition in these two sets using the smoothed and non-smoothed trigram language models as *LM term* respectively. The recognition results are shown in

Table 3: The *unit accuracy* of our CTC+Trigram+Len method, compared with the state of the art on *Breakfast* dataset.

Model	**Unit Acc.**
OCDC [17]	0.104
HMM+Grammar [5]	0.207
ECTC [4]	0.367
CTC+Trigram+Len	**0.434**

Table. 2.

The smoothed trigram language model gave better performance on *Unknown* (0.346 Vs. 0.328), which indicates the smoothing is helpful for generalizing our model to unseen relationships. However, we also observe a performance drop on *Known*, compared with the non-smoothed trigram (0.509 Vs. 0.523). The reason is that the smoothing method we adopted over-smooths the distribution of *known*, introducing more insertion and substitution errors. It worths looking further in this issue in the future and develops a better smoothing method. On the total testing set, the smoothed trigram gave minor improvement over the non-smoothed one (0.1% point).

6. 3. 3 Comparison with the state-of-the-art.

In Table 3, we compare our CTC+Trigram+Len method with three state-of-the-art methods, namely Ordered Constrained Discriminative Clustering (OCDC) [17], Hidden Markov Model equipped with a network-based Grammar (HMM+Grammar) [5] and Extended Connectionist Temporal Classification (ECTC), under the *unit accuracy* metric. All of them are capable of learning models from unsegmented videos and perform action sequence recognition.

Our method substantially outperformed all of the three state-of-the-art methods, surpassing the best one among them by 6.7% points (0.434 Vs. 0.367). Note that our CTC baseline has already been slightly better than ECTC (0.370 Vs. 0.367). We ascribe it to our better engineering.

7. Conclusion

In this paper, we introduce to equip a CTC network with a n-gram statistical language model for action sequence recognition. The whole framework does not rely on any segmentation of human actions in videos to learn the model. We demonstrate the effectiveness of our method by experimenting on the realistic *Breakfast* dataset. The experimental results show that our method gives a good performance and outperforms the current state of the art.

Acknowledgment. This work was supported by JST CREST Grant Number JPMJCR1687, Japan.

References

[1] H. Wang and C. Schmid. Action Recognition with Improved Trajectories. In *Proc. ICCV*. 2013.

[2] L. Wang et al. Temporal Segment Networks: Towards Good Practices for Deep Action Recognition. In *Proc. ECCV*. 2016.

[3] D. Moltisanti et al. Trespassing the Boundaries: Labeling Temporal Bounds for Object Interactions in Egocentric Video. In *Proc. ICCV*. 2017.

[4] D.A. Huang et al. Connectionist Temporal Modeling for Weakly Supervised Action Labeling. In *Proc. ECCV*. 2016.

[5] H. Kuehne et al. Weakly Supervised Learning of Actions from Transcripts. In *CVIU*. 2017.

[6] P. Molchanov et al. Online Detection and Classification of Dynamic Hand Gestures with Recurrent 3d Convolutional Neural Network. In *Proc. CVPR*. 2016.

[7] N. C. Camgoz et al. SubUNets: End-to-end Hand Shape and Continuous Sign Language Recognition. In *Proc. ICCV*. 2017.

[8] A. Graves et al. Connectionist Temporal Classification: Labelling Unsegmented Sequence Data with Recurrent Neural Networks. In *Proc. ICML*. 2006.

[9] B. Singh et al. A Multi-stream Bi-directional Recurrent Neural Network for Fine-grained Action Detection. In *Proc. CVPR*. 2016.

[10] M. Lin et al. CTC Network with Statistical Language Modeling for Action Sequence Recognition in Videos. In *Proc. Thematic Workshop on ACM MM*. 2017.

[11] P. Pan et al. Hierarchical Recurrent Neural Encoder for Video Representation with Application to Captioning. In *Proc. CVPR*. 2016.

[12] S. Katz. Estimation of Probabilities from Sparse Data for the Language Model Component of a Speech Recognizer. In *ASSP*. Vol.35, No.3, pp.400-401. 1987.

[13] A. Gaidon et al. Temporal Localization of Actions with Actoms. In *PAMI*. Vol.35, No.11, pp.2782-2795. 2013.

[14] H. Kuehne et al. The language of actions: Recovering the Syntax and Semantics of Goal-directed Human Activities. In *Proc. CVPR*. 2014.

[15] C. Lea et al. Segmental Spatiotemporal CNNs for Fine-grained Action Segmentation. In *Proc. ECCV*. 2016.

[16] A. Richard and J. Gall. Temporal Action Detection Using a Statistical Language Model. In *Proc. CVPR*. 2016.

[17] P. Bojanowski et al. Weakly Supervised Action Labeling in Videos under Ordering Constraints. In *Proc. ECCV*. 2014.

[18] A. Richard et al. Weakly Supervised Action Learning with RNN based Fine-to-coarse Modeling. In *Proc. CVPR*. 2017.

[19] F. Jelinek and R. Mercer. Interpolated Estimation of Markov Source Parameters from Sparse Data. In *Proc. of Workshop on Pattern Recognition in Practice*. 1980.

[20] H. Ney et al. On the Estimation of ' Small ' Probabilities by Leaving-one-out. In *PAMI*. pp.1202-1212. 1995.

[21] T.C. Bell et al. Text Compression. Prentice-Hall, Inc. 1990.

[22] D. Guthrie et al. A Closer Look at Skip-gram Modelling. In *Proc. LREC*. 2006.

[23] T. Mikolov et al. Recurrent Neural Network based Language Model. In *Proc. Interspeech*. 2010.

[24] D. Soutner et al. On a Hybrid NN/HMM Speech Recognition System with a RNN-based Language Model. In *Proc. SPECOM*. 2014.

[25] S.F. Chen and J. Goodman. An Empirical Study of Smoothing Techniques for Language Modeling. In *Proc. ACL*. 1996.

[26] Y. Miao et al. EESEN: End-to-end Speech Recognition Using Deep RNN Models and WFST-based Decoding. In *Proc. ASRU*. 2015.

[27] A.L. Maas et al. Lexicon-Free Conversational Speech Recognition with Neural Networks. In *Proc. NAACL*. 2015.

[28] H. Kuehne et al. An End-to-end Generative Framework for Video Segmentation and Recognition. In *Proc. WACV*. 2016.