Learning a Grammar Inducer from Massive Uncurated Instructional Videos

Songyang Zhang[1]*, **Linfeng Song**[2], **Lifeng Jin**[2], **Haitao Mi**[2], **Kun Xu**[2], **Dong Yu**[2] and **Jiebo Luo**[1]

[1]University of Rochester, Rochester, NY, USA

szhang83@ur.rochester.edu, jluo@cs.rochester.edu

[2]Tencent AI Lab, Bellevue, WA, USA

{lfsong,lifengjin,haitaomi,kxkunxu,dyu}@tencent.com

Abstract

Video-aided grammar induction aims to leverage video information for finding more accurate syntactic grammars for accompanying text. While previous work focuses on building systems for inducing grammars on text that are well-aligned with video content, we investigate the scenario, in which text and video are only in loose correspondence. Such data can be found in abundance online, and the weak correspondence is similar to the indeterminacy problem studied in language acquisition. Furthermore, we build a new model that can better learn video-span correlation without manually designed features adopted by previous work. Experiments show that our model trained only on large-scale YouTube data with no text-video alignment reports strong and robust performances across three unseen datasets, despite domain shift and noisy label issues. Furthermore our model yields higher F1 scores than the previous state-of-the-art systems trained on in-domain data.

1 Introduction

Grammar induction is a fundamental and long-lasting (Lari and Young, 1990; Clark, 2001; Klein and Manning, 2002) problem in computational linguistics, which aims to find hierarchical syntactic structures from plain sentences. Unlike supervised methods (Charniak, 2000; Collins, 2003; Petrov and Klein, 2007; Zhang and Clark, 2011; Cross and Huang, 2016; Kitaev and Klein, 2018) that require human annotated treebanks, *e.g.*, Penn Treebank (Marcus et al., 1993), grammar inducers do not rely on any human annotations for training. Grammar induction is attractive since annotating syntactic trees by human language experts is expensive and time consuming, while the current treebanks are limited to several major languages and domains.

Recently, deep learning models have achieved remarkable success across NLP tasks, and neural models have been designed (Shen et al., 2018b,a; Kim et al., 2019a,b; Jin et al., 2018) for grammar induction, which greatly advanced model performance on induction with raw text. Recent efforts have started to consider other useful information from multiple modalities, such as images (Shi et al., 2019; Jin and Schuler, 2020) and videos (Zhang et al., 2021). Specifically, Zhang et al. (2021) show that multi-modal information (e.g. motion, sound and objects) from videos can significantly improve the induction accuracy on verb and noun phrases. Such work uses curated multi-modal data publicly available on the web, which all assume that the meaning of a sentence needs to be identical (e.g., being a caption) to the corresponding video or image. This assumption limits usable data to several small-scale benchmarks (Lin et al., 2014; Xu et al., 2016; Hendricks et al., 2017) with expensive human annotations on image/video captions.

The noisy correspondence between form and meaning is one of the main research questions in language acquisition (Akhtar and Montague, 1999; Gentner et al., 2001; Dominey and Dodane, 2004), where different proposals attempt to address this indeterminacy faced by children. There has been computational work incorporating such indeterminacy into their models (Yu and Siskind, 2013; Huang et al., 2021). For modeling empirical grammar learning with multi-modal inputs, two important questions still remain open: 1) *how can a grammar inducer benefit from large-scale multi-media data (e.g., YouTube videos) with noisy text-to-video correspondence*? and 2) *how can a grammar inducer show robust performances across multiple domains and datasets*? By using data with only weak cross-modal correspondence, such as YouTube videos and their automatically generated subtitles, we allow the computational models to face a similar indeterminacy problem, and exam-

* This work was done when Songyang Zhang was an intern at Tencent AI Lab.

Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, pages 233–247
December 7-11, 2022

ine how indeterminacy interacts with data size to influence learning behavior and performance of the induction models.

In this paper, we conduct the first investigation on both questions. Specifically, we collect 2.4 million video clips and the corresponding subtitles from instructional YouTube videos (HowTo100M Miech et al. 2019) to train multi-modal grammar inducers, instead of using the training data from a benchmark where text and video are in alignment. We then propose a novel model, named Pre-Trained Compound Probabilistic Context-Free Grammars (PTC-PCFG), that extends previous work (Shi et al., 2019; Zhang et al., 2021) by incorporating a video-span matching loss term into the Compound PCFG (Kim et al., 2019a) model. To better capture the video-span correlation, it leverages CLIP (Miech et al., 2020), a state-of-the-art model pretrained on video subtitle retrieval, as the encoders for both video and text. Compared with previous work (Zhang et al., 2021) that independently extracts features from each modality before merging them using a simple Transformer (Vaswani et al., 2017) encoder, the encoders of our model have been pre-trained to merge such multi-modal information, and no human efforts are needed to select useful modalities from the full set.

Experiments on three benchmarks show that our model, which is trained on noisy YouTube video clips and no data from these benchmarks, produces substantial gains over the previous state-of-the-art system (Zhang et al., 2021) trained on in-domain video clips with human annotated captions. Furthermore, our model demonstrates robust performances across all three datasets. We suggest the limitations of our model and future directions for improvements through analysis and discussions. Code will be released upon paper acceptance.

In summary, the main contributions are:

- We are the first to study training a grammar inducer with massive general-domain noisy video clips instead of benchmark data, introducing the indeterminacy problem to the induction model.
- We propose PTC-PCFG, a novel model for unsupervised grammar induction. It is simpler in design than previous models and can better capture the video-text matching information.
- Trained only on noisy YouTube videos without finetuning on benchmark data, PTC-PCFG reports stronger performances than previous models trained on benchmark data across three benchmarks.

2 Background and Motivation

2.1 Compound PCFGs

A PCFG model in Chomsky Normal Form can be defined as a tuple of 6 terms $(\mathcal{S}, \mathcal{N}, \mathcal{P}, \Sigma, \mathcal{R}, \Pi)$, where they correspond to the start symbol, the sets of non-terminals, pre-terminals, terminals, production rules and their probabilities. Given pre-defined numbers of non-terminals and pre-terminals, a PCFG induction model tries to estimate the probabilities for all production rules.

The compound PCFG (C-PCFG) model (Kim et al., 2019a) adopts a mixture of PCFGs. Instead of a corpus-level prior used in previous work (Kurihara and Sato, 2006; Johnson et al., 2007; Wang and Blunsom, 2013; Jin et al., 2018), C-PCFG imposes a sentence-specific prior on the distribution of possible PCFGs. Specifically in the generative story, the probability π_r for production rule r is estimated by model g that assigns a latent variable $\mathbf{z}$ for each sentence σ, and $\mathbf{z}$ is drawn from a prior distribution:

$$\pi_r = g(r, \mathbf{z}; \theta), \qquad \mathbf{z} \sim p(\mathbf{z}). \tag{1}$$

where θ represents the model parameters. The probabilities for all three types of CFG rules are defined as follows:

$$\begin{aligned}
\pi_{S\to A} &= \frac{\exp(\mathbf{u}_A^\top f_s([\mathbf{w}_S; \mathbf{z}]))}{\sum_{A' \in \mathcal{N}} \exp(\mathbf{u}_{A'}^\top f_s([\mathbf{w}_S; \mathbf{z}]))}, \\
\pi_{A\to BC} &= \frac{\exp(\mathbf{u}_{BC}^\top [\mathbf{w}_A; \mathbf{z}])}{\sum_{B',C' \in \mathcal{N}\cup\mathcal{P}} \exp(\mathbf{u}_{B'C'}^\top [\mathbf{w}_A; \mathbf{z}]))}, \\
\pi_{T\to w} &= \frac{\exp(\mathbf{u}_w^\top f_t([\mathbf{w}_T; \mathbf{z}]))}{\sum_{w' \in \Sigma} \exp(\mathbf{u}_{w'}^\top f_t([\mathbf{w}_T; \mathbf{z}]))},
\end{aligned} \tag{2}$$

where $A \in \mathcal{N}$, B and $C \in \mathcal{N} \cup \mathcal{P}$, $T \in \mathcal{P}$, $w \in \Sigma$. Both $\mathbf{w}$ and $\mathbf{u}$ are dense vectors representing words and all types of non-terminals, and f_s and f_t are neural encoding functions.

Optimizing the C-PCFG model involves maximizing the marginal likelihood $p(\sigma)$ of each training sentence σ for all possible $\mathbf{z}$:

$$\log p_\theta(\sigma) = \log \int_{\mathbf{z}} \sum_{t \in \mathcal{T}_\mathcal{G}(\sigma)} p_\theta(t|\mathbf{z}) p(\mathbf{z}) d\mathbf{z} \tag{3}$$

where $\mathcal{T}_\mathcal{G}(\sigma)$ indicates all possible parsing trees for sentence σ. Since computing the integral over $\mathbf{z}$

is intractable, this objective is optimized by maximizing its evidence lower bound ELBO(σ; ϕ, θ):

$$\begin{aligned}\mathrm{ELBO}(\sigma;\phi,\theta) = \mathbb{E}_{q_\phi(\mathbf{z}|\sigma)}[\log p_\theta(\sigma|\mathbf{z})] \\ -\mathrm{KL}[q_\phi(\mathbf{z}|\sigma)||p(\mathbf{z})],\end{aligned} \tag{4}$$

where $q_\phi(\mathbf{z}|\sigma)$ is the variational posterior calculated by another neural network with parameters ϕ. Given a sampled $\mathbf{z}$, the log-likelihood term $\log p_\theta(\sigma|\mathbf{z})$ is calculated via the inside algorithm. The KL term can be computed analytically when both the prior $p(\mathbf{z})$ and the variational posterior $q_\phi(\mathbf{z}|\sigma)$ are Gaussian (Kingma and Welling, 2014).

2.2 Multi-Modal Compound PCFGs

Multi-Modal Compound PCFGs (MMC-PCFG) (Zhang et al., 2021) extends C-PCFG with a model to match a video v with a span c in a parse tree t of a sentence σ. It extracts M visual and audio features from a video v and encodes them via a multi-modal transformer (Gabeur et al., 2020), denoted as $\mathbf{\Psi} = \{\psi^i\}_{i=1}^M$. The word representation $\mathbf{h}_i$ of the ith word is computed by BiLSTM. Given a particular span $c = w_i, \ldots, w_j$, its representation $\mathbf{c}$ is the weighted sum of all label-specific span representations:

$$\mathbf{c} = \sum_{k=1}^{|\mathcal{N}|} p(k|c,\sigma) f_k \left(\frac{1}{j-i+1} \sum_{l=i}^{j} \mathbf{h}_l \right), \tag{5}$$

where $\{p(k|c,\sigma)|1 \leq k \leq |\mathcal{N}|\}$ are the phrasal label probabilities of span c. The representation of a span $\mathbf{c}$ is then correspondingly projected to M separate embeddings via gated embedding (Miech et al., 2018), denoted as $\mathbf{\Xi} = \{\boldsymbol{\xi}^i\}_{i=1}^M$. Finally the video-text matching loss is defined as a sum over all video-span matching losses weighted by the marginal probability of a span from the parser:

$$s_{mm}(v,\sigma) = \sum_{c\in\sigma} p(c|\sigma) h_{mm}(\mathbf{\Xi},\mathbf{\Psi}), \tag{6}$$

where $h_{mm}(\mathbf{\Xi},\mathbf{\Psi})$ is a hinge loss measuring the distances from video v to the matched and unmatched (*i.e.* span from another sentence) span c and c' and the distances from span c to the matched and unmatched (*i.e.* another video) video v and v':

$$\omega_i(\mathbf{c}) = \frac{\exp(\mathbf{u}_i^\top \mathbf{c})}{\sum_{j=1}^M \exp(\mathbf{u}_j^\top \mathbf{c})}, \tag{7}$$

$$o(\mathbf{\Xi},\mathbf{\Psi}) = \sum_{i=1}^M \omega_i(\mathbf{c}) \cos(\boldsymbol{\xi}^i, \boldsymbol{\psi}^i), \tag{8}$$

$$\begin{aligned}h_{mm}(\mathbf{\Xi},\mathbf{\Psi}) = \mathbb{E}_{c'}[o(\mathbf{\Xi}',\mathbf{\Psi}) - o(\mathbf{\Xi},\mathbf{\Psi})) + \epsilon]_+ \\ + \mathbb{E}_{v'}[o(\mathbf{\Xi},\mathbf{\Psi}') - o(\mathbf{\Xi},\mathbf{\Psi}) + \epsilon]_+,\end{aligned} \tag{9}$$

where $\mathbf{\Xi}'$ is a set of unmatched span expert embeddings of $\mathbf{\Psi}$, $\mathbf{\Psi}'$ is a set of unmatched video representations of $\mathbf{\Xi}$, ϵ is a positive margin, $[\cdot]_+ = max(0,\cdot)$, $\{\mathbf{u}_i\}_{i=1}^M$ are learned weights, and the expectations are approximated with one sample drawn from the training data. During training, both ELBO and the video-text matching loss are jointly optimized.

2.3 Limitation and Motivation

Existing work on multi-modal grammar induction aims at leveraging strict correspondence between image/video and text for information about syntactic categories and structures of the words and spans in the text. However, such datasets are expensive to annotate. Besides, the ambiguous correspondence between language and real-world context, observed in language acquisition, is not really reflected in such training setups.

As a result, we believe that the previous work fails to answer the following important questions: 1) how well a grammar inducer would perform when it is trained only on noisy multi-media data; 2) how the scale of training data would affect the performance and cross-domain robustness?

3 Training a Grammar Inducer with Massive YouTube Videos

We make the first investigation into the above questions by leveraging massive video clips from instructional YouTube videos to train our grammar inducer. Different from the benchmark data used by previous work, the YouTube video clips do not contain paired sentences. This section will first introduce the method for generating noisy training instances (video clip and sentence pairs) from YouTube videos (§3.1), before describing a novel grammar induction model (§3.2) with pre-trained text and video encoders.

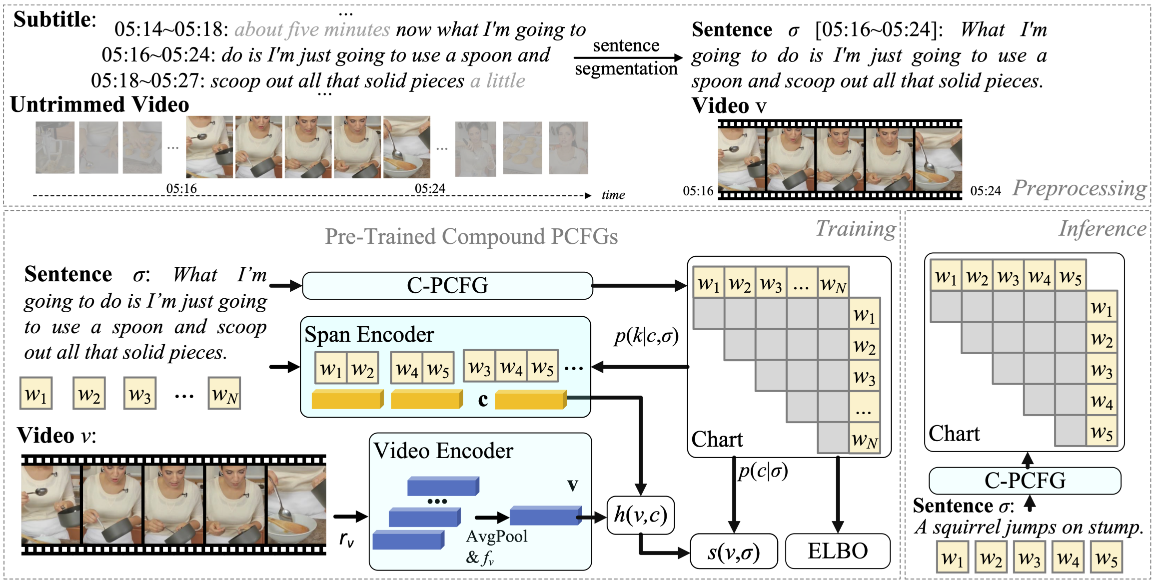

Figure 1: The pipeline of our approach.

3.1 Harvesting Training Instances from YouTube Videos

Given a YouTube video, we would like to generate a set of video clip and subtitle pairs $\Omega = \{(v, \sigma)\}$, where each subtitle σ is a complete sentence and is aligned in time with its paired video clip v. To this end, the YouTube API is chosen to obtain all subtitles of the video. But, our observation finds that most obtained subtitles are not complete sentences, and in some cases, a complete sentence can last for several continuous video fragments. Meanwhile, they do not contain any punctuation, which is a key factor for sentence segmentation. As shown in the top part of Figure 1, we design an algorithm that takes the following steps to find each complete sentence and its corresponding video clip.

Sentence segmentation. In the first step, we try to find complete sentences from the subtitles. We first concatenate all subtitles from the same video are concatenated into a very long sequence of tokens. Next, a punctuation restoration model[1] (Tilk and Alumäe, 2016) is adopted to insert punctuation into the sequence. Lastly, sentences are segmented based on certain punctuation (*e.g.*, ".", "?", "!").

Video clip extraction. In the second step, we trim the corresponding video clips. Each raw subtitle contains its start and an end times. We assume each word within the raw subtitle occupies equal time and record the start and end times for each word. After that, given a complete sentence $\sigma = w_1, w_2, ..., w_N$, we use the start time of its first word w_1 and the end time of its last word w_N as the start and end times of σ. Lastly, we segment a complete sentence σ's corresponding video clip v based on its start and end times.

3.2 Model: Pre-Trained Compound PCFGs

After harvesting large-scale sentence and video pairs, the next step is to build a strong grammar induction model that can benefit from them. In this section, we introduce our Pre-Trained Compound PCFGs (PTC-PCFG) model for unsupervised grammar induction. As shown in the lower part of Figure 1, the PTC-PCFG model composes of a video encoder, a span encoder and a parsing model. Both the video encoder and the span encoder are initialized from the MIL-NCE model (Miech et al., 2020), a pre-trained video-text matching model that takes a simple design and has shown superior zero-shot results on many video understanding tasks, such as video retrieval, video question answering, *etc*. We first introduce the pre-trained video and span encoders, before covering the training and inference details of PTC-PCFG.

Video encoding. The first step is to encode a video v to its representation $\mathbf{v}$. To do this, we first segment v into small video clips, where each video clip v_i consists of T frames. Following Zhang et al. (2021), we sample L video clips with equal interval for efficiency. We use the video encoder from the MIL-NCE model (Miech et al., 2020) as our video encoder and only fine-tune its last fully connected

[1] We manually punctuate subtitles from 10 videos randomly selected from HowTo100M, which contains 461 sentences after annotation. The punctuation restoration model has an overall F1 score of 74.1% with the manual labels.

layer f_v for efficiency. In more detail, for each sampled video clip, we pre-compute the input of f^v as its representation, denoted as $\{\mathbf{h}_i^v\}_{i=1}^L$. Then we feed them into f^v and average the output as its representation $\mathbf{v}$, denoted as,

$$\mathbf{v} = \texttt{AvgPool}(\{f^v(\mathbf{h}_i^v)\}_{i=1}^L), \tag{10}$$

where AvgPool indicates average pooling.

Span encoding. The next step is to compute a span representation $\mathbf{c}$ for each particular span $c = w_i, \ldots, w_j$ $(1 \leq i < j \leq N)$ in sentence $\sigma = w_1, w_2, \ldots, w_N$. The pre-trained text encoder of MIL-NCE consists of a word embedding layer and two stacked fully connected layers, f_0^c and f_1^c. Motivated by Zhao and Titov (2020); Zhang et al. (2021), we expect to learn $|\mathcal{N}|$ different span representations, each is specified for one non-terminal node. However, directly applying the pre-trained text encoder is not feasible, since it has only one output layer f_1^c. Therefore, we duplicate f_1^c for $|\mathcal{N}|$ times, denoted as $\{f_k^c\}_{k=1}^{|\mathcal{N}|}$, and compose $|\mathcal{N}|$ label-specific output layers. In more detail, we first encode each word w_i with the word embedding layer, denoted as $\mathbf{h}_i^c$. Then we feed the word embeddings to f_0^c, ReLU, maximum pooling and each label-specific output layer sequentially. we also compute the probabilities of its phrasal labels $\{p(k|c,\sigma)|1 \leq k \leq |\mathcal{N}|\}$, as illustrated in Section 2.1. Lastly, the span representation $\mathbf{c}$ is the sum of all label-specific span representations weighted by the probabilities we predicted, denoted as:

$$\begin{aligned} \tau &= \texttt{MaxPool}(\texttt{ReLU}(f_0^c(\mathbf{h}_i^c))) \\ \mathbf{c} &= \sum_{k=1}^{|\mathcal{N}|} p(k|c,\sigma) f_k^c(\tau), \end{aligned} \tag{11}$$

where MaxPool is a maximum pooling operation and ReLU is a ReLU activation function.

Training. As shown in lower left of Figure 1, we optimize both the video-text matching loss and evidence lower bound during training. We first compute the similarity between a video clip v and a particular span c via dot product and then compute a triplet hinge loss as following,

$$\begin{aligned} h(v,c) = &\mathbb{E}_{c'}[\mathbf{c}' \cdot \mathbf{v} - \mathbf{c} \cdot \mathbf{v} + \epsilon]_+ \\ &+ \mathbb{E}_{v'}[\mathbf{c} \cdot \mathbf{v}' - \mathbf{c} \cdot \mathbf{v} + \epsilon]_+, \end{aligned} \tag{12}$$

where ϵ is a positive margin, $[\cdot]_+ = max(0, \cdot)$, v' is a clip from a different video and c' is a span from a different sentence. The video-text matching loss is correspondingly defined as,

$$s(v,\sigma) = \Sigma_{c \in \sigma} p(c|\sigma) h(v,c), \tag{13}$$

where $p(c|\sigma)$ is the probability of a particular span c being a syntactic phrase. Finally, the overall loss function is composed by the ELBO and the video-text matching loss:

$$\mathcal{L}(\phi,\theta) = \sum_{(v,\sigma)\in\Omega} -\text{ELBO}(\sigma;\phi,\theta) + \alpha s(v,\sigma), \tag{14}$$

where α is a constant balancing these two terms.

Inference. During inference, given a sentence σ, we predict the most likely tree t^* without accessing videos, as shown in the lower right of Figure 1. Since computing the integral over $\mathbf{z}$ is intractable, we estimate t^* with the following approximation,

$$\begin{aligned} t^* &= \arg\max_t \int_{\mathbf{z}} p_\theta(t|\mathbf{z}) p_\theta(\mathbf{z}|\sigma) d\mathbf{z} \\ &\approx \arg\max_t p_\theta(t|\sigma, \boldsymbol{\mu}_\phi(\sigma)), \end{aligned} \tag{15}$$

where $\boldsymbol{\mu}_\phi(\sigma)$ is the mean vector of the variational posterior $q_\phi(\mathbf{z}|\sigma)$, and t^* is obtained by the CYK algo. (Cocke, 1969; Younger, 1967; Kasami, 1966).

4 Experiments

4.1 Datasets

Following previous work, we evaluate all systems on three benchmarks (i.e., DiDeMo, YouCook2 and MSRVTT). Instead of training on benchmark data, our models are trained on the data harvested from HowTo100M dataset. Below shows more details about these datasets:

DiDeMo (Hendricks et al., 2017) contains 10k unedited personal Flickr videos. Each video is associated with roughly 3-5 video-sentence pairs. There are 32 994, 4 180 and 4021 video pairs in the training, validation and testing sets.

YouCook2 (Zhou et al., 2018) contains 2000 long untrimmed YouTube videos from 89 cooking recipes. The procedure steps for each video are annotated with temporal boundaries and described by imperative English sentences. There are 8 913, 969 and 3 310 video-sentence pairs in the training, validation and testing sets.

MSRVTT (Xu et al., 2016) contains 10k generic YouTube videos accompanied by 200k captions annotated by paid human workers. There are 130 260, 9 940 and 59 794 video-sentence pairs in the training, validation and testing sets.

HowTo100M (Miech et al., 2019) is a large-scale dataset of 136 million video clips sourced from 1.22M narrated instructional web videos depicting humans performing more than 23k different visual tasks. Noted that there are 404 videos in HowTo100M exists in YouCook2, we exclude these videos during training.

4.2 Evaluation

We discard punctuation, lowercase all words, replace numbers with a special token and ignore trivial single-word and sentence-level spans during testing following Kim et al. (2019a). Besides, we follow previous work (Shi et al., 2019; Zhang et al., 2021) by using a state-of-the-art constituency parser (Benepar Kitaev et al. 2019) to obtain the reference trees for evaluation[2]. Following Shi et al. (2020); Zhang et al. (2021), all models are run 5 times for 1 epoch with different random seeds. For each model, we report the averaged sentence-level F1 (S-F1) and corpus-level F1 (C-F1) of its runs on each testing set.

4.3 Implementation Details

We use Spacy [3] for tokenization and keep sentences with fewer than 40 words for training due to the limited computational resources. Each video is decoded at 16 fps and $L = 8$ video clips are sampled in total, where each clip contains $T = 16$ frames. We train baseline models, C-PCFG and MMC-PCFG with the same hyper-parameters suggested by Kim et al. (2019a) and Zhang et al. (2021). The parsing model of PTC-PCFG has the same hyper-parameter setting as C-PCFG and MMC-PCFG (Please refer their papers for details). The constant α is set to 1. We select the top 20 000 most common words in HowTo100M as vocabulary for all datasets. All baseline methods and ours are optimized by Adam (Kingma and Ba, 2015) with a learning rate of 0.001, $\beta_1 = 0.75$ and $\beta_2 = 0.999$. All parameters (except the video-text matching model in PTC-PCFG) are initialized with Xavier uniform initializer (Glorot and Bengio, 2010). All our models in experiments are trained for 1 epoch with batch size of 32, without finetuning on the target dataset.

[2]For each dataset, we randomly select 50 sentences and manually label their constituency parse trees. Benepar has S-F1 scores of 98.1% (DiDeMo), 97.2% (YouCook2) and 98.1% (MSRVTT) with manual labels.

[3]https://spacy.io/

4.4 Main Results

Figure 2-4 compare our proposed PTC-PCFG approach with recently proposed state-of-the-art models: C-PCFG (Kim et al., 2019a) and MMC-PCFG[4] (Zhang et al., 2021). To pinpoint more fine-grained contributions, we also train these models on HowTo100M data.

The effectiveness of HowTo100M. We find that C-PCFG achieve better performance when they are trained with more instances from HowTo100M than the original in-domain training sets, where the largest improvements are +18.1%, +21.7% and +1.4% S-F1 scores on DiDeMo, YouCook2 and MSRVTT, respectively. These results indicate that grammar inducers are generally robust against the instances with noisy text-video correspondence. As the results, learning from noisy YouTube videos can benefit model's overall performance and its generalization ability across multiple domains.

The effectiveness of PTC-PCFG. Comparing C-PCFG, MMC-PCFG and PTC-PCFG trained on different amount of HowTo100M data, we found that PTC-PCFG achieves the best performances in all three datasets. It can further improve S-F1 to +6.3% on DiDeMo, +16.7% on YouCook2 and +2.8% on MSRVTT. This demonstrates the effectiveness of the PTC-PCFG model. In particular, utilizing the video and span encoders pre-trained on a relevant tasks (*e.g.*, video retrieval) can benefit unsupervised grammar induction.

Performance comparison over data scale. On DiDeMo and MSRVTT, we observe that PTC-PCFG achieves the best performance with 592k HowTo100M training samples, and further increasing the number of training instances does not improve the parsing performance on these two datasets. In contrast, the performance gain of PTC-PCFG on YouCook2 further increases with increasing training data. The reason can be that the domain of HowTo100M is closer to YouCook2 (both are instructional videos) than the other two datasets. Future work includes adding data from other sources to the whole training set more domain generic.

[4]Since audios are removed by HowTo100M authors, we implement MMC-PCFG with video features only, including object features(ResNeXt, SENet), action features (I3D, R2P1D, S3DG), scene features, OCR features and face features.

Table 1: Performance comparison across different training set. We use HT to represent HowTo100M dataset for short, where the number in the brackets indicates the number of samples used for training. The values highlighted by **bold** and *italic* fonts indicate the top-2 methods, respectively. All numbers are shown in percentage(%). The remaining tables follow the same notations.

Method	Trainset	DiDeMo		YouCook2		MSRVTT	
		C-F1	S-F1	C-F1	S-F1	C-F1	S-F1
MMC-PCFG	DiDeMo	$55.0_{\pm3.7}$	$58.9_{\pm3.4}$	$49.1_{\pm4.4}$	$53.0_{\pm4.9}$	$49.6_{\pm1.4}$	$53.8_{\pm0.9}$
MMC-PCFG	YouCook2	$40.1_{\pm4.4}$	$44.2_{\pm4.4}$	$44.7_{\pm5.2}$	$48.9_{\pm5.7}$	$34.0_{\pm6.4}$	$37.5_{\pm6.8}$
MMC-PCFG	MSRVTT	$\mathit{59.4}_{\pm2.9}$	$\mathit{62.7}_{\pm3.3}$	$49.6_{\pm3.9}$	$54.2_{\pm4.1}$	$\mathit{56.0}_{\pm1.4}$	$60.0_{\pm1.2}$
MMC-PCFG	HT(592k)	$58.5_{\pm7.3}$	$62.4_{\pm7.9}$	$\mathit{53.9}_{\pm6.6}$	$\mathit{58.0}_{\pm7.1}$	$55.1_{\pm7.0}$	$\mathit{60.2}_{\pm8.0}$
PTC-PCFG	HT(592k)	$\mathbf{61.3}_{\pm3.9}$	$\mathbf{65.2}_{\pm5.3}$	$\mathbf{58.9}_{\pm2.5}$	$\mathbf{63.2}_{\pm2.3}$	$\mathbf{57.4}_{\pm4.6}$	$\mathbf{62.8}_{\pm5.7}$

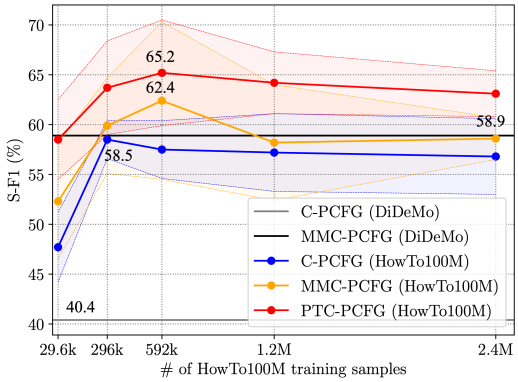

Figure 2: Performance Comparison on DiDeMo. The doted lines and their enclosed area represent the mean and variance of each model trained on HowTo100M at different scales. We mark the highest average S-F1 achieved by each method with numbers. The remaining figures follow the same notations.

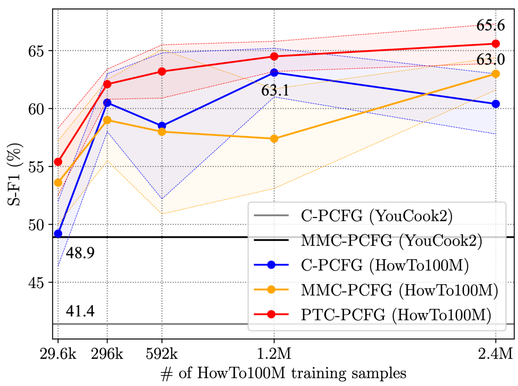

Figure 3: Performance Comparison on YouCook2.

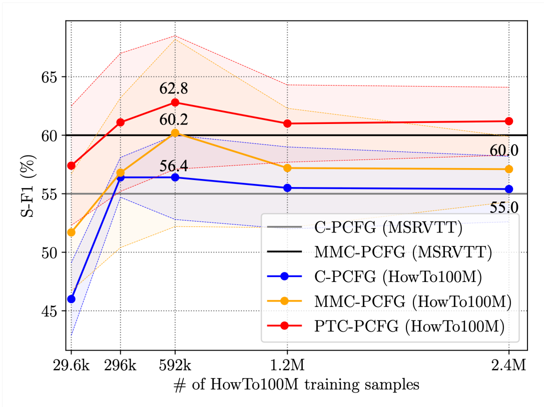

Figure 4: Performance Comparison on MSRVTT.

4.5 Cross-dataset Evaluation

We evaluate the robustness of models across different datasets, as shown in Table 1. Comparing MMC-PCFG trained on in-domain datasets (Row 1-3), we can observe that MMC-PCFG trained on MSRVTT achieves the best overall performance, while MMC-PCFG trained on YouCook2 is the worst. We believe this is due to the different number of training instances[5] and the domain gap between different datasets. Comparing Rows 1-4, we can observe that the MMC-PCFG model trained on HT(592k) (Row 4) is the best or the second place regarding C-F1 and S-F1 compared with its variants trained on in-domain datasets (Rows 1-3). This demonstrates that the our processed video-text training instances are abundant, rich in content and can serve for general purpose. Comparing Rows 4 and 5, PTC-PCFG outperforms MMC-PCFG in both C-F1 and S-F1 in all three datasets and has smaller variance. This demonstrate that our model can leverage pre-trained video-text matching knowledge and learn consistent grammar induction.

4.6 Effectiveness of Pre-Training

In this section, we explore how different pre-trained video and text encoders can affect the parsing performance, and the results are shown in Table 2. In particular, we study different

[5]The number of training instances in YouCook2, DiDeMo and MSRVTT are 8.9K, 32.9K and 130.2K, respectively.

Table 2: Performance comparison across different video and span encoders.

Video-Text Model		Trainset	DiDeMo		YouCook2		MSRVTT	
Video Encoder	Span Encoder		C-F1	S-F1	C-F1	S-F1	C-F1	S-F1
MIL-NCE	LSTM	HT(296k)	$52.4_{\pm 5.5}$	$54.4_{\pm 5.4}$	$51.5_{\pm 5.4}$	$56.5_{\pm 5.2}$	$49.7_{\pm 5.5}$	$53.4_{\pm 5.8}$
MM	LSTM	HT(296k)	$53.6_{\pm 3.2}$	$55.8_{\pm 3.1}$	$53.1_{\pm 5.7}$	$57.9_{\pm 5.6}$	$48.9_{\pm 3.5}$	$52.5_{\pm 3.6}$
MIL-NCE	TinyBERT	HT(296k)	$\mathit{54.8}_{\pm 5.4}$	$\mathit{56.4}_{\pm 6.0}$	$\mathit{55.7}_{\pm 4.0}$	$\mathit{60.2}_{\pm 3.5}$	$\mathit{52.3}_{\pm 4.3}$	$\mathit{56.0}_{\pm 5.0}$
MIL-NCE	MIL-NCE	HT(296k)	$\mathbf{59.5}_{\pm 4.3}$	$\mathbf{63.7}_{\pm 4.7}$	$\mathbf{57.1}_{\pm 1.7}$	$\mathbf{62.1}_{\pm 1.3}$	$\mathbf{55.7}_{\pm 5.0}$	$\mathbf{61.1}_{\pm 5.9}$
CLIP	CLIP	HT(296k)	$52.9_{\pm 2.3}$	$54.9_{\pm 2.6}$	$53.3_{\pm 2.2}$	$58.9_{\pm 2.1}$	$49.1_{\pm 2.6}$	$53.0_{\pm 2.9}$

video encoders[6], including the S3D-based encoder from MIL-NCE (Miech et al., 2020) (*MIL-NCE*), the multi-modal video encoder from MMC-PCFG (Zhang et al., 2021) (*MM*) and the CLIP model for image-text pre-training (Radford et al., 2021) (*CLIP*). We also investigate various text encoders, including an LSTM encoder with random initialization (Zhang et al., 2021; Zhao and Titov, 2020), a pre-trained TinyBERT (Jiao et al., 2020) model, the text encoder from MIL-NCE (Miech et al., 2020), and the text encoder from CLIP (Radford et al., 2021).

Comparing Rows 1 with 2, we can observe that MM is better than the video encoder of MIL-NCE regarding C-F1 and S-F1 on all three datasets, as MM provides more comprehensive video features. By comparing row 1 with 3, we can also observe that TinyBERT, which is distilled from BERT (Devlin et al., 2019), outperforms the randomly initialized LSTM encoder. However, both MM and TinyBERT are independently trained only on vision or language tasks, where the vision-language correspondences are not considered during pre-training. Therefore, we further investigate the encoders jointly pre-trained on large scale multimedia datasets, including the video-text matching model MIL-NCE (Row 4) and the image-text matching model CLIP (Row 5). We can observe that by leveraging both video and text encoders in MIL-NCE can improve the parsing performance by a large margin on all three datasets. On the other hand, CLIP does not perform well, since it is designed for static images and other multi-modal information (e.g., motion) is ignored.

4.7 Qualitative Analysis

In figure 5, we visualize a parser tree predicted by the best run of C-PCFG trained on MSRVTT, MMC-PCFG trained on MSRVTT, MMC-PCFG trained on HT(296k) and PTC-PCFG trained on HT(296k), as well as its reference tree. We can

[6] We list the video processing details in Appendix A.

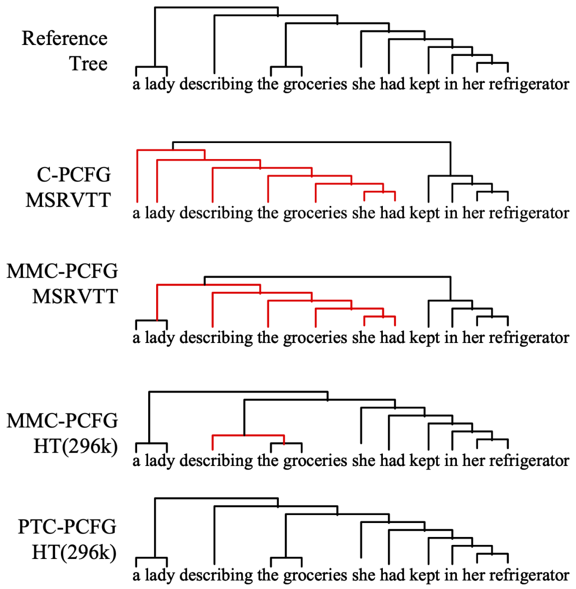

Figure 5: Parse trees predicted by different models for sentence *a lady describing the groceries she had kept in her refrigerator*. The red line shows the difference between the predicted trees and the reference tree.

observe that C-PCFG trained on MSRVTT fails at noun phrase "*a lady*", while MMC-PCFG trained on MSRVTT succeeds. MMC-PCFG can be further improved by training on HT(296k), however, fails at noun phrase "*the groceries she had kept in her refrigerator*". Our PTC-PCFG can leverage the pre-trained matching knowledge and make the correct prediction.

5 Related Work

Grammar Induction has a long and rich history in the computational linguistics. Earlier work (Shen et al., 2018a,b; Drozdov et al., 2019; Kim et al., 2019a; Jin et al., 2019; Yang et al., 2021a,b) on grammar induction with pure unsupervised learning showed promising results. Instead of learning purely from text, recent work improved the parsing performance with paired images (Shi et al., 2019; Zhao and Titov, 2020) or videos (Zhang et al., 2021). However, they are all limited to small benchmarks and specified for a few domains. In contrast, our work leverages massive noisy video-subtitle pairs from YouTube without any manual annotations.

Video Retrieval has been a hot topic in the computer vision field for many years. Earlier approaches focused on model design (Gabeur et al., 2020; Zhang et al., 2019), while more recent approaches (Radford et al., 2021; Miech et al., 2020) focused on the pre-training on a large scale dataset and demonstrated superior zero-shot results on many downstream tasks. These models are simple in design and provide representative features with less human effort in annotations. In this work, we demonstrate that unsupervised grammar induction can also benefit from the pre-trained video-text model.

6 Conclusion

In this paper, we have investigated how massive instructional YouTube video and subtitle pairs can improve grammar induction. We have also proposed a new model that leverages the latest advances in multi-modal pre-training to learn better video-span correlation. Experiments on three benchmarks demonstrate superior and robust performances of our model over previous systems. We leave exploring other pre-trained video-text matching models and more publicly available data (e.g., YouTube videos from other domains and TV shows) in future work.

7 Limitations

Although our model faces a similar indeterminacy problem like children do, and results show that induction works even with noisy correspondence, there are a few factors which prevent this result from being directly applied to language acquisition. Our models only use instructional video and do not have the capability to interact with the world, both of which are unrealistic for human language learners. The complexity of the PCFG induction algorithm we use is cubic to the number of syntactic categories, therefore potentially limits the usefulness of larger amounts of data, where finer subcategories may be learned. Algorithms such as in Yang et al. (2021b) could be used in conjunction with multimodal inputs to examine this issue.

Following previous work, our experiments are only conducted on English video-text datasets. However, our framework is general for grammar induction in many languages. Since our training instances are originally collected from Internet and are uploaded by users, the dataset itself might have misinformation. Meanwhile, training a model on a large-scale dataset could have high cost in energy and carbon emission. We list our computational cost of our experiments in Appendix B.

References

Nameera Akhtar and Lisa Montague. 1999. Early lexical acquisition: the role of cross-situational learning. *First Language*, 19(57):347–358.

Joao Carreira and Andrew Zisserman. 2017. Quo vadis, action recognition? a new model and the kinetics dataset. In *CVPR*.

Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *NAACL*.

Alexander Clark. 2001. Unsupervised induction of stochastic context-free grammars using distributional clustering. In *ConLL*.

John Cocke. 1969. *Programming languages and their compilers: Preliminary notes*. New York University.

Michael Collins. 2003. Head-driven statistical models for natural language parsing. *Computational linguistics*, 29(4):589–637.

James Cross and Liang Huang. 2016. Span-based constituency parsing with a structure-label system and provably optimal dynamic oracles. In *EMNLP*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*.

Peter F Dominey and Christelle Dodane. 2004. Indeterminacy in language acquisition: the role of child directed speech and joint attention. *Journal of Neurolinguistics*, 17(2):121–145.

Andrew Drozdov, Patrick Verga, Mohit Yadav, Mohit Iyyer, and Andrew McCallum. 2019. Unsupervised latent tree induction with deep inside-outside recursive auto-encoders. In *NAACL*.

Valentin Gabeur, Chen Sun, Karteek Alahari, and Cordelia Schmid. 2020. Multi-modal transformer for video retrieval. In *ECCV*.

Dedre Gentner, Lera Boroditsky, Melissa Bowerman, and Stephen Levinson. 2001. Individuation, relativity, and early word. *Language, culture and cognition*, 3:215–256.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*.

Lisa Anne Hendricks, Oliver Wang, Eli Shechtman, Josef Sivic, Trevor Darrell, and Bryan Russell. 2017. Localizing moments in video with natural language. In *ICCV*.

Jie Hu, Li Shen, and Gang Sun. 2018. Squeeze-and-excitation networks. In *CVPR*.

Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. 2017. Densely connected convolutional networks. In *CVPR*.

Zhenyu Huang, Guocheng Niu, Xiao Liu, Wenbiao Ding, Xinyan Xiao, Hua Wu, and Xi Peng. 2021. Learning with noisy correspondence for cross-modal matching. In *NeurIPS*.

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. 2020. Tinybert: Distilling bert for natural language understanding. In *EMNLP: Findings*.

Lifeng Jin, Finale Doshi-Velez, Timothy Miller, William Schuler, and Lane Schwartz. 2018. Unsupervised grammar induction with depth-bounded PCFG. *TACL*.

Lifeng Jin, Finale Doshi-Velez, Timothy Miller, Lane Schwartz, and William Schuler. 2019. Unsupervised learning of PCFGs with normalizing flow. In *ACL*.

Lifeng Jin and William Schuler. 2020. Grounded PCFG induction with images. In *AACL-IJCNLP*.

Mark Johnson, Thomas L Griffiths, and Sharon Goldwater. 2007. Bayesian inference for PCFGs via markov chain monte carlo. In *NAACL*.

Tadao Kasami. 1966. An efficient recognition and syntax-analysis algorithm for context-free languages. *Coordinated Science Laboratory Report no. R-257*.

Yoon Kim, Chris Dyer, and Alexander M Rush. 2019a. Compound probabilistic context-free grammars for grammar induction. In *ACL*.

Yoon Kim, Alexander M Rush, Lei Yu, Adhiguna Kuncoro, Chris Dyer, and Gábor Melis. 2019b. Unsupervised recurrent neural network grammars. In *NAACL*.

Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR*.

Diederik P Kingma and Max Welling. 2014. Auto-encoding variational bayes. In *ICLR*.

Nikita Kitaev, Steven Cao, and Dan Klein. 2019. Multilingual constituency parsing with self-attention and pre-training. In *ACL*.

Nikita Kitaev and Dan Klein. 2018. Constituency parsing with a self-attentive encoder. In *ACL*.

Dan Klein and Christopher D Manning. 2002. A generative constituent-context model for improved grammar induction. In *ACL*.

Kenichi Kurihara and Taisuke Sato. 2006. Variational bayesian grammar induction for natural language. In *ICGI*.

Karim Lari and Steve J Young. 1990. The estimation of stochastic context-free grammars using the inside-outside algorithm. *Computer speech & language*, 4(1):35–56.

Tsung-Yi Lin, Michael Maire, Serge J Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *ECCV*.

Mitch Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of english: The penn treebank. *Computational Linguistics*, 19(2):313–330.

Antoine Miech, Jean-Baptiste Alayrac, Lucas Smaira, Ivan Laptev, Josef Sivic, and Andrew Zisserman. 2020. End-to-end learning of visual representations from uncurated instructional videos. In *CVPR*.

Antoine Miech, Ivan Laptev, and Josef Sivic. 2018. Learning a text-video embedding from incomplete and heterogeneous data. *arXiv preprint arXiv:1804.02516*.

Antoine Miech, Dimitri Zhukov, Jean-Baptiste Alayrac, Makarand Tapaswi, Ivan Laptev, and Josef Sivic. 2019. HowTo100M: Learning a text-video embedding by watching hundred million narrated video clips. In *ICCV*.

Slav Petrov and Dan Klein. 2007. Improved inference for unlexicalized parsing. In *NAACL*.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. 2021. Learning transferable visual models from natural language supervision. *arXiv preprint arXiv:2103.00020*.

Florian Schroff, Dmitry Kalenichenko, and James Philbin. 2015. Facenet: A unified embedding for face recognition and clustering. In *CVPR*.

Yikang Shen, Zhouhan Lin, Chin wei Huang, and Aaron Courville. 2018a. Neural language modeling by jointly learning syntax and lexicon. In *ICLR*.

Yikang Shen, Shawn Tan, Alessandro Sordoni, and Aaron Courville. 2018b. Ordered neurons: Integrating tree structures into recurrent neural networks. In *ICLR*.

Haoyue Shi, Karen Livescu, and Kevin Gimpel. 2020. On the role of supervision in unsupervised constituency parsing. In *EMNLP*.

Haoyue Shi, Jiayuan Mao, Kevin Gimpel, and Karen Livescu. 2019. Visually grounded neural syntax acquisition. In *ACL*.

Ottokar Tilk and Tanel Alumäe. 2016. Bidirectional recurrent neural network with attention mechanism for punctuation restoration. In *Interspeech*.

Du Tran, Heng Wang, Lorenzo Torresani, Jamie Ray, Yann LeCun, and Manohar Paluri. 2018. A closer look at spatiotemporal convolutions for action recognition. In *CVPR*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS*.

Pengyu Wang and Phil Blunsom. 2013. Collapsed variational bayesian inference for PCFGs. In *CoNLL*.

Wenhai Wang, Enze Xie, Xiaoge Song, Yuhang Zang, Wenjia Wang, Tong Lu, Gang Yu, and Chunhua Shen. 2019. Efficient and accurate arbitrary-shaped text detection with pixel aggregation network. In *ICCV*.

Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. 2017. Aggregated residual transformations for deep neural networks. In *CVPR*.

Jun Xu, Tao Mei, Ting Yao, and Yong Rui. 2016. MSR-VTT: A large video description dataset for bridging video and language. In *CVPR*.

Songlin Yang, Yanpeng Zhao, and Kewei Tu. 2021a. Neural bi-lexicalized PCFG induction. In *ACL*.

Songlin Yang, Yanpeng Zhao, and Kewei Tu. 2021b. PCFGs can do better: Inducing probabilistic context-free grammars with many symbols. In *NAACL*.

Daniel H Younger. 1967. Recognition and parsing of context-free languages in time n3. *Information and control*, 10(2):189–208.

Haonan Yu and Jeffrey Mark Siskind. 2013. Grounded language learning from video described with sentences. In *ACL*.

Kaipeng Zhang, Zhanpeng Zhang, Hao Wang, Zhifeng Li, Yu Qiao, and Wei Liu. 2017. Detecting faces using inside cascaded contextual cnn. In *ICCV*.

Songyang Zhang, Linfeng Song, Lifeng Jin, Kun Xu, Dong Yu, and Jiebo Luo. 2021. Video-aided unsupervised grammar induction. In *NAACL*.

Songyang Zhang, Jinsong Su, and Jiebo Luo. 2019. Exploiting temporal relationships in video moment localization with natural language. In *ACMMM*.

Yue Zhang and Stephen Clark. 2011. Syntactic processing using the generalized perceptron and beam search. *Computational linguistics*, 37(1):105–151.

Yanpeng Zhao and Ivan Titov. 2020. Visually grounded compound PCFGs. In *EMNLP*.

Luowei Zhou, Chenliang Xu, and Jason J Corso. 2018. Towards automatic learning of procedures from web instructional videos. In *AAAI*.

A Video Processing Details

MIL-NCE. Following the implementation[7] of MIL-NCE, we extract 1 feature per second from video encoder's last fully connected layer. All videos are decoded at 16 frames per second (fps).
MM. We list the details of object, action, scene, OCR and face feature extraction as below:

- ResNeXt (Xie et al., 2017). We use the *ResNeXt101* version implemented by torchvision[8]. Videos are decoded at 1 fps and we extract 1 feature per second from the last fully connected layer.

- SENet (Hu et al., 2018). We use the *SENet154* version implemented by Cadene[9]. Videos are decoded at 1 fps and we extract 1 feature per second from the last fully connected layer.

- I3D(Carreira and Zisserman, 2017). We use the *I3D_8x8_R50* version implemented by SlowFast[10]. We decode videos at 4 fps and extract 1 feature per 2 seconds from the last fully connected layer.

- S3DG (Miech et al., 2020). We use the implementation from HERO[11]. Videos are decoded at 30 fps and we extract 1 feature per 1.5 seconds from the global averaged pooling layer.

- R2P1D (Tran et al., 2018). We use the *r2plus1d_34* version implemented by torchvision. we decode videos at 16 fps and extract 1 feature per 2 seconds.

- Scene. We use *densenet161* (Huang et al., 2017) implemented by CSAILVision[12]. Videos are decoded at 1 fps and we extract 1 feature per second from the last fully connected layer.

- OCR. We use the text detector PANet (Wang et al., 2019) and the text recognizer *seg_r31* implemented by MMOCR[13]. we decode videos at 0.5 fps and extract 1 feature per 2 seconds.

- Face. We use face detector MTCNN (Zhang et al., 2017) and face recognizer (Schroff et al., 2015) implemented by FaceNet[14]. We decode videos at 1 fps and extract 1 feature per second.

CLIP. Following the implementation[15] of CLIP, we extract 1 video feature per second from *ViT-B/32*'s last fully connected layer. All videos are decoded at 1 fps.

B Computational Cost

All our models are trained on 2 32GB V100 GPUs. The approximate time cost for each run of different model is listed in Table 1. For each model, we run 5 times with different random seeds in parallel. During training, the video encoder, the span encoder and C-PCFG are involved, which contains 76.6M parameters in total. During inference, since only C-PCFG is involved, there are 23.0M parameters in total.

Table 1: The approximate training time (hours) of different model on a single run.

Model	HT(29.6k)	HT(296k)	HT(592k)	HT(1.2M)	HT(2.4M)
C-PCFG	0.07	0.7	1.5	2.9	5.8
MMC-PCFG	0.75	7.5	15	30	60
PTC-PCFG	0.50	5.0	10	20	40

[7] https://github.com/antoine77340/S3D_HowTo100M
[8] https://github.com/pytorch/vision
[9] https://github.com/Cadene/pretrained-models.pytorch
[10] https://github.com/facebookresearch/SlowFast
[11] https://github.com/linjieli222/HERO_Video_Feature_Extractor
[12] https://github.com/CSAILVision/places365
[13] https://github.com/open-mmlab/mmocr
[14] https://github.com/timesler/facenet-pytorch
[15] https://github.com/openai/CLIP

C Performance Comparison - Full Tables

We compare the performances of different models trained on different datasets. The full experiment results are demonstrated in Table 2-4. LBranch, RBranch and Random represent left branching tree, right branching tree and random tree, respectively. In addition to C-F1 and S-F1, we also evaluate the recall of each model on different phrase types, including NP, VP, PP, SBAR, ADJP and ADVP. All numbers are shown in percentage(%).

Table 2: Performance comparison on DiDeMo.

	Method	Trainset	NP	VP	PP	SBAR	ADJP	ADVP	C-F1	S-F1
	LBranch	None	41.7	0.1	0.1	0.7	7.2	0.0	16.2	18.5
	RBranch	None	32.8	**91.5**	66.5	**88.2**	36.9	**63.6**	53.6	57.5
	Random	None	36.5$_{\pm0.6}$	30.5$_{\pm0.5}$	30.1$_{\pm0.5}$	25.7$_{\pm2.8}$	29.5$_{\pm2.3}$	28.5$_{\pm4.8}$	29.4$_{\pm0.3}$	32.7$_{\pm0.5}$
	C-PCFG	DiDeMo	72.9$_{\pm5.5}$	16.5$_{\pm6.2}$	23.4$_{\pm16.9}$	26.6$_{\pm15.9}$	25.0$_{\pm11.6}$	14.7$_{\pm12.8}$	38.2$_{\pm5.0}$	40.4$_{\pm4.1}$
VC-PCFG	ResNeXt	DiDeMo	64.4$_{\pm21.4}$	25.7$_{\pm17.7}$	34.6$_{\pm25.0}$	40.5$_{\pm26.3}$	16.7$_{\pm9.5}$	28.4$_{\pm21.3}$	40.0$_{\pm13.7}$	41.8$_{\pm14.0}$
	SENet	DiDeMo	70.5$_{\pm15.3}$	25.7$_{\pm15.9}$	36.5$_{\pm24.6}$	36.8$_{\pm25.9}$	21.2$_{\pm12.5}$	23.6$_{\pm16.8}$	42.6$_{\pm10.4}$	44.0$_{\pm10.4}$
	I3D	DiDeMo	57.9$_{\pm13.5}$	45.7$_{\pm14.1}$	45.8$_{\pm17.2}$	38.2$_{\pm14.8}$	28.4$_{\pm9.2}$	22.0$_{\pm9.3}$	45.1$_{\pm6.0}$	49.2$_{\pm6.0}$
	R2P1D	DiDeMo	61.2$_{\pm8.5}$	38.1$_{\pm5.4}$	62.1$_{\pm4.1}$	61.5$_{\pm5.1}$	21.4$_{\pm11.4}$	40.8$_{\pm7.3}$	48.1$_{\pm4.4}$	50.7$_{\pm4.2}$
	S3DG	DiDeMo	61.3$_{\pm13.4}$	31.7$_{\pm16.7}$	51.8$_{\pm8.0}$	50.3$_{\pm6.5}$	18.0$_{\pm4.5}$	35.2$_{\pm11.4}$	44.0$_{\pm2.7}$	46.5$_{\pm5.1}$
	Scene	DiDeMo	62.2$_{\pm9.6}$	30.6$_{\pm12.3}$	41.1$_{\pm24.8}$	35.2$_{\pm21.9}$	21.4$_{\pm14.0}$	27.6$_{\pm17.1}$	41.7$_{\pm6.5}$	44.9$_{\pm7.4}$
	Audio	DiDeMo	64.2$_{\pm18.6}$	21.3$_{\pm26.5}$	34.7$_{\pm11.0}$	37.3$_{\pm19.6}$	26.1$_{\pm4.9}$	18.2$_{\pm11.6}$	38.7$_{\pm3.7}$	39.5$_{\pm5.2}$
	OCR	DiDeMo	64.4$_{\pm15.0}$	27.4$_{\pm19.5}$	42.8$_{\pm31.2}$	35.9$_{\pm20.7}$	14.6$_{\pm1.7}$	23.2$_{\pm24.0}$	41.9$_{\pm16.9}$	44.6$_{\pm17.5}$
	Face	DiDeMo	60.8$_{\pm16.0}$	31.5$_{\pm17.0}$	52.8$_{\pm9.8}$	49.3$_{\pm5.6}$	12.6$_{\pm3.3}$	32.9$_{\pm14.6}$	43.9$_{\pm4.5}$	46.3$_{\pm5.5}$
	Speech	DiDeMo	61.8$_{\pm12.8}$	26.6$_{\pm17.6}$	43.8$_{\pm34.5}$	34.2$_{\pm20.6}$	14.4$_{\pm4.8}$	12.9$_{\pm9.6}$	40.9$_{\pm16.0}$	43.1$_{\pm16.1}$
	Concat	DiDeMo	68.6$_{\pm8.6}$	24.9$_{\pm19.9}$	39.7$_{\pm19.5}$	39.3$_{\pm19.8}$	10.8$_{\pm2.8}$	18.3$_{\pm18.1}$	42.2$_{\pm12.3}$	43.2$_{\pm14.2}$
	MMC-PCFG	DiDeMo	67.9$_{\pm9.8}$	52.3$_{\pm9.0}$	63.5$_{\pm8.6}$	60.7$_{\pm10.8}$	34.7$_{\pm17.0}$	50.4$_{\pm8.3}$	55.0$_{\pm3.7}$	58.9$_{\pm3.4}$
	MMC-PCFG	YouCook2	47.9$_{\pm10.4}$	34.6$_{\pm2.7}$	58.2$_{\pm12.9}$	19.9$_{\pm2.6}$	11.0$_{\pm4.0}$	25.5$_{\pm4.8}$	40.1$_{\pm4.4}$	44.2$_{\pm4.4}$
	MMC-PCFG	MSRVTT	56.5$_{\pm6.8}$	*70.8*$_{\pm4.4}$	**82.6**$_{\pm3.3}$	*62.5*$_{\pm6.0}$	42.6$_{\pm2.6}$	52.9$_{\pm7.6}$	59.4$_{\pm2.9}$	62.7$_{\pm3.3}$
	C-PCFG	HT(29.6k)	74.8$_{\pm1.1}$	27.3$_{\pm5.7}$	43.6$_{\pm13.2}$	32.9$_{\pm7.4}$	32.4$_{\pm4.6}$	44.5$_{\pm7.8}$	45.7$_{\pm3.9}$	47.7$_{\pm3.5}$
	MMC-PCFG	HT(29.6k)	75.4$_{\pm2.1}$	33.2$_{\pm12.5}$	54.9$_{\pm7.8}$	33.7$_{\pm9.8}$	39.2$_{\pm4.7}$	43.8$_{\pm7.1}$	49.8$_{\pm4.7}$	52.3$_{\pm5.8}$
	PTC-PCFG	HT(29.6k)	66.0$_{\pm9.4}$	53.7$_{\pm13.9}$	68.2$_{\pm4.5}$	50.4$_{\pm5.0}$	35.2$_{\pm4.2}$	52.7$_{\pm8.9}$	54.9$_{\pm3.4}$	58.5$_{\pm4.0}$
	C-PCFG	HT(296k)	81.4$_{\pm1.6}$	36.4$_{\pm6.9}$	67.0$_{\pm1.9}$	45.9$_{\pm3.5}$	46.5$_{\pm4.8}$	49.9$_{\pm8.3}$	55.6$_{\pm1.4}$	58.5$_{\pm1.9}$
	MMC-PCFG	HT(296k)	81.9$_{\pm2.1}$	42.7$_{\pm15.0}$	65.3$_{\pm6.4}$	39.1$_{\pm8.5}$	*48.0*$_{\pm8.1}$	43.7$_{\pm6.7}$	57.1$_{\pm4.2}$	59.9$_{\pm4.8}$
	PTC-PCFG	HT(296k)	76.0$_{\pm4.9}$	55.3$_{\pm11.9}$	70.7$_{\pm6.4}$	53.7$_{\pm9.5}$	43.4$_{\pm4.8}$	47.2$_{\pm13.3}$	59.5$_{\pm4.3}$	63.7$_{\pm4.7}$
	C-PCFG	HT(592k)	82.4$_{\pm1.9}$	37.6$_{\pm9.2}$	63.1$_{\pm5.8}$	37.4$_{\pm7.9}$	45.8$_{\pm5.5}$	*53.8*$_{\pm9.9}$	55.5$_{\pm2.7}$	57.5$_{\pm2.9}$
	MMC-PCFG	HT(592k)	79.0$_{\pm5.7}$	52.5$_{\pm19.1}$	64.3$_{\pm6.1}$	45.5$_{\pm9.9}$	44.1$_{\pm4.4}$	44.5$_{\pm11.7}$	58.5$_{\pm7.3}$	62.4$_{\pm7.9}$
	PTC-PCFG	HT(592k)	79.1$_{\pm2.9}$	55.3$_{\pm18.4}$	73.7$_{\pm5.1}$	50.6$_{\pm8.2}$	38.3$_{\pm5.6}$	47.2$_{\pm5.5}$	**61.3**$_{\pm3.9}$	**65.2**$_{\pm5.3}$
	C-PCFG	HT(1.2M)	*82.9*$_{\pm1.5}$	32.2$_{\pm12.5}$	67.5$_{\pm2.1}$	41.3$_{\pm8.5}$	45.6$_{\pm4.8}$	48.8$_{\pm8.0}$	55.1$_{\pm3.3}$	57.2$_{\pm3.9}$
	MMC-PCFG	HT(1.2M)	**83.4**$_{\pm2.6}$	43.2$_{\pm15.9}$	51.3$_{\pm19.1}$	36.2$_{\pm7.0}$	**49.4**$_{\pm5.5}$	56.2$_{\pm6.9}$	55.1$_{\pm5.0}$	58.2$_{\pm5.8}$
	PTC-PCFG	HT(1.2M)	77.9$_{\pm1.9}$	49.0$_{\pm7.6}$	*78.3*$_{\pm3.9}$	47.2$_{\pm6.5}$	35.8$_{\pm8.7}$	49.8$_{\pm11.6}$	*60.0*$_{\pm2.3}$	*64.2*$_{\pm3.1}$
	C-PCFG	HT(2.4M)	**83.4**$_{\pm2.2}$	31.0$_{\pm4.7}$	68.1$_{\pm8.4}$	48.5$_{\pm7.7}$	46.9$_{\pm7.3}$	45.9$_{\pm8.2}$	55.2$_{\pm3.0}$	56.8$_{\pm3.8}$
	MMC-PCFG	HT(2.4M)	81.7$_{\pm2.8}$	37.6$_{\pm4.8}$	71.1$_{\pm6.9}$	47.2$_{\pm6.1}$	41.7$_{\pm8.1}$	51.2$_{\pm7.2}$	57.0$_{\pm2.1}$	58.6$_{\pm2.1}$
	PTC-PCFG	HT(2.4M)	78.9$_{\pm1.6}$	49.4$_{\pm6.9}$	75.0$_{\pm5.2}$	48.7$_{\pm6.0}$	36.9$_{\pm5.4}$	51.8$_{\pm13.8}$	*60.0*$_{\pm2.5}$	63.1$_{\pm2.3}$

Table 3: Performance comparison on YouCook2.

	Method	Trainset	NP	VP	PP	SBAR	ADJP	ADVP	C-F1	S-F1
	LBranch	None	1.7	42.8	0.4	8.1	1.5	0.0	6.8	5.9
	RBranch	None	35.6	47.5	67.0	**88.9**	33.9	65.0	35.0	41.6
	Random	None	$27.2_{\pm0.3}$	$27.1_{\pm1.4}$	$29.9_{\pm0.5}$	$31.3_{\pm5.2}$	$26.9_{\pm7.7}$	$26.2_{\pm11.9}$	$21.2_{\pm0.2}$	$24.0_{\pm0.2}$
	C-PCFG	YouCook2	$47.4_{\pm18.4}$	$49.4_{\pm11.9}$	$58.0_{\pm22.6}$	$45.7_{\pm6.0}$	$27.7_{\pm15.1}$	$36.2_{\pm7.4}$	$37.8_{\pm6.7}$	$41.4_{\pm6.6}$
VC-PCFG	ResNeXt	YouCook2	$46.5_{\pm13.7}$	$40.8_{\pm9.8}$	$67.9_{\pm12.7}$	$50.5_{\pm13.3}$	$22.3_{\pm6.7}$	$38.8_{\pm21.3}$	$38.2_{\pm8.3}$	$42.8_{\pm8.4}$
	SENet	YouCook2	$48.3_{\pm14.4}$	$40.7_{\pm9.2}$	$73.6_{\pm11.2}$	$45.5_{\pm17.0}$	$26.9_{\pm13.6}$	$41.2_{\pm17.5}$	$39.9_{\pm8.7}$	$44.9_{\pm8.3}$
	I3D	YouCook2	$48.1_{\pm10.7}$	$39.0_{\pm8.0}$	$79.4_{\pm8.4}$	$50.0_{\pm14.9}$	$18.5_{\pm7.0}$	$41.2_{\pm4.1}$	$40.6_{\pm3.6}$	$45.7_{\pm3.2}$
	R2P1D	YouCook2	$52.4_{\pm10.9}$	$33.7_{\pm16.4}$	$66.7_{\pm10.7}$	$49.5_{\pm13.8}$	$25.8_{\pm10.6}$	$33.8_{\pm12.4}$	$39.4_{\pm8.1}$	$44.4_{\pm8.3}$
	S3DG	YouCook2	$50.4_{\pm13.1}$	$32.6_{\pm16.3}$	$71.7_{\pm7.5}$	$33.3_{\pm5.9}$	$30.8_{\pm17.5}$	$40.0_{\pm7.1}$	$39.3_{\pm6.5}$	$44.1_{\pm6.6}$
	Audio	YouCook2	$51.2_{\pm3.1}$	$42.0_{\pm7.2}$	$61.5_{\pm18.0}$	$51.0_{\pm14.8}$	$23.5_{\pm16.8}$	$48.8_{\pm8.2}$	$39.2_{\pm4.7}$	$43.3_{\pm4.9}$
	OCR	YouCook2	$48.6_{\pm8.1}$	$41.5_{\pm4.1}$	$65.5_{\pm17.4}$	$39.9_{\pm4.4}$	$18.5_{\pm6.6}$	$53.8_{\pm14.7}$	$38.6_{\pm5.5}$	$43.2_{\pm5.6}$
	Concat	YouCook2	$50.3_{\pm10.3}$	$42.3_{\pm2.9}$	$81.6_{\pm8.7}$	$40.1_{\pm3.9}$	$17.7_{\pm8.2}$	$52.5_{\pm5.6}$	$42.3_{\pm5.7}$	$47.0_{\pm5.6}$
	MMC-PCFG	YouCook2	$62.7_{\pm9.8}$	$45.3_{\pm2.8}$	$63.4_{\pm17.7}$	$43.9_{\pm4.8}$	$26.2_{\pm7.5}$	$35.0_{\pm3.5}$	$44.7_{\pm5.2}$	$48.9_{\pm5.7}$
	MMC-PCFG	DiDeMo	$63.8_{\pm4.5}$	$62.1_{\pm7.4}$	$70.7_{\pm9.0}$	$56.8_{\pm9.2}$	$35.4_{\pm7.2}$	$51.2_{\pm4.1}$	$49.1_{\pm4.4}$	$53.0_{\pm4.9}$
	MMC-PCFG	MSRVTT	$63.1_{\pm9.2}$	$51.5_{\pm7.3}$	$82.7_{\pm1.5}$	$64.9_{\pm10.8}$	$30.4_{\pm3.0}$	$40.0_{\pm6.1}$	$49.6_{\pm3.9}$	$54.2_{\pm4.1}$
	C-PCFG	HT(29.6k)	$68.2_{\pm3.3}$	$43.5_{\pm4.9}$	$48.8_{\pm20.2}$	$34.5_{\pm4.2}$	$28.5_{\pm3.2}$	$56.7_{\pm6.2}$	$44.3_{\pm2.4}$	$49.2_{\pm2.8}$
	MMC-PCFG	HT(29.6k)	$68.8_{\pm3.0}$	$51.3_{\pm11.5}$	$65.5_{\pm11.2}$	$33.7_{\pm6.3}$	$32.0_{\pm4.3}$	$55.0_{\pm11.3}$	$48.8_{\pm3.5}$	$53.6_{\pm3.6}$
	PTC-PCFG	HT(29.6k)	$66.3_{\pm4.2}$	$55.5_{\pm8.3}$	$76.5_{\pm5.7}$	$46.2_{\pm9.0}$	$33.4_{\pm7.4}$	$40.0_{\pm9.7}$	$50.2_{\pm3.4}$	$55.4_{\pm2.9}$
	C-PCFG	HT(296k)	$77.5_{\pm3.1}$	$56.6_{\pm4.5}$	$74.4_{\pm5.5}$	$51.1_{\pm9.0}$	$40.5_{\pm6.4}$	$63.3_{\pm8.5}$	$55.0_{\pm2.7}$	$60.5_{\pm2.5}$
	MMC-PCFG	HT(296k)	$76.5_{\pm4.1}$	$61.8_{\pm9.7}$	$67.2_{\pm8.7}$	$34.0_{\pm16.2}$	$41.0_{\pm7.9}$	$68.3_{\pm8.2}$	$53.8_{\pm3.4}$	$59.0_{\pm3.5}$
	PTC-PCFG	HT(296k)	$77.5_{\pm2.5}$	$65.8_{\pm6.1}$	$78.9_{\pm7.0}$	$61.6_{\pm3.9}$	$42.4_{\pm6.3}$	$60.0_{\pm3.3}$	$57.1_{\pm1.7}$	$62.1_{\pm1.3}$
	C-PCFG	HT(592k)	$76.7_{\pm5.6}$	$64.7_{\pm12.3}$	$65.4_{\pm22.5}$	$45.2_{\pm13.9}$	$45.1_{\pm9.4}$	$68.3_{\pm6.2}$	$54.2_{\pm6.0}$	$58.5_{\pm6.3}$
	MMC-PCFG	HT(592k)	$72.5_{\pm12.7}$	$68.9_{\pm7.0}$	$68.3_{\pm18.2}$	$54.3_{\pm5.7}$	$\mathbf{50.2}_{\pm2.1}$	$\mathit{75.0}_{\pm9.1}$	$53.9_{\pm6.6}$	$58.0_{\pm7.1}$
	PTC-PCFG	HT(592k)	$78.7_{\pm5.3}$	$69.9_{\pm3.6}$	$80.5_{\pm2.8}$	$58.9_{\pm12.3}$	$43.2_{\pm4.0}$	$65.0_{\pm6.2}$	$58.9_{\pm2.5}$	$63.2_{\pm2.3}$
	C-PCFG	HT(1.2M)	$\mathit{80.1}_{\pm2.5}$	$63.5_{\pm11.9}$	$78.5_{\pm2.5}$	$52.5_{\pm13.7}$	$42.9_{\pm6.5}$	$66.7_{\pm7.5}$	$58.1_{\pm2.4}$	$63.1_{\pm2.1}$
	MMC-PCFG	HT(1.2M)	$75.7_{\pm2.2}$	$64.8_{\pm9.7}$	$57.7_{\pm21.4}$	$51.2_{\pm9.4}$	$\mathit{49.3}_{\pm4.9}$	$\mathit{75.0}_{\pm7.5}$	$52.5_{\pm4.0}$	$57.4_{\pm4.3}$
	PTC-PCFG	HT(1.2M)	$78.1_{\pm2.6}$	$\mathbf{72.2}_{\pm4.1}$	$\mathbf{85.8}_{\pm5.1}$	$\mathit{69.3}_{\pm6.6}$	$41.5_{\pm8.7}$	$66.7_{\pm10.5}$	$\mathit{60.1}_{\pm1.4}$	$\mathit{64.5}_{\pm1.3}$
	C-PCFG	HT(2.4M)	$75.9_{\pm3.9}$	$61.5_{\pm8.0}$	$78.2_{\pm4.1}$	$59.7_{\pm13.4}$	$45.4_{\pm5.6}$	$75.0_{\pm5.3}$	$55.9_{\pm2.6}$	$60.4_{\pm2.6}$
	MMC-PCFG	HT(2.4M)	$78.0_{\pm2.8}$	$69.8_{\pm4.1}$	$79.2_{\pm5.4}$	$50.6_{\pm14.2}$	$41.5_{\pm5.1}$	$71.7_{\pm10.0}$	$58.3_{\pm1.8}$	$63.0_{\pm1.4}$
	PTC-PCFG	HT(2.4M)	$\mathbf{81.9}_{\pm3.1}$	$\mathit{71.5}_{\pm4.5}$	$\mathit{83.0}_{\pm4.1}$	$59.0_{\pm15.5}$	$40.7_{\pm3.4}$	$\mathbf{81.7}_{\pm6.2}$	$\mathbf{61.1}_{\pm2.0}$	$\mathbf{65.6}_{\pm1.7}$

Table 4: Performance comparison on MSRVTT.

	Method	Trainset	NP	VP	PP	SBAR	ADJP	ADVP	C-F1	S-F1
	LBranch	None	34.6	0.1	0.9	0.2	3.8	0.3	14.4	16.8
	RBranch	None	34.6	$\mathbf{90.9}$	67.5	$\mathbf{94.8}$	25.4	54.8	54.2	58.6
	Random	None	$34.6_{\pm0.1}$	$26.8_{\pm0.1}$	$28.1_{\pm0.2}$	$24.6_{\pm0.3}$	$24.8_{\pm1.0}$	$28.1_{\pm1.4}$	$27.2_{\pm0.1}$	$30.5_{\pm0.1}$
	C-PCFG	MSRVTT	$46.6_{\pm3.2}$	$61.1_{\pm3.3}$	$72.5_{\pm8.3}$	$63.7_{\pm4.0}$	$33.1_{\pm7.1}$	$67.1_{\pm4.7}$	$50.7_{\pm3.2}$	$55.0_{\pm3.2}$
VC-PCFG	ResNeXt	MSRVTT	$48.6_{\pm3.0}$	$59.0_{\pm6.0}$	$72.0_{\pm3.6}$	$62.1_{\pm5.2}$	$32.6_{\pm2.5}$	$70.4_{\pm6.4}$	$50.7_{\pm1.7}$	$54.9_{\pm2.2}$
	SENet	MSRVTT	$49.0_{\pm4.4}$	$63.5_{\pm6.4}$	$71.7_{\pm4.8}$	$60.9_{\pm10.6}$	$34.0_{\pm6.4}$	$\mathit{74.1}_{\pm1.9}$	$52.2_{\pm1.2}$	$56.0_{\pm1.6}$
	I3D	MSRVTT	$53.9_{\pm10.5}$	$63.2_{\pm9.1}$	$73.7_{\pm2.9}$	$65.3_{\pm9.1}$	$35.0_{\pm6.8}$	$73.8_{\pm4.1}$	$54.5_{\pm1.6}$	$59.1_{\pm1.7}$
	R2P1D	MSRVTT	$52.8_{\pm3.6}$	$63.3_{\pm4.6}$	$73.1_{\pm10.1}$	$66.9_{\pm2.0}$	$34.0_{\pm2.2}$	$72.5_{\pm4.2}$	$54.0_{\pm2.5}$	$58.0_{\pm2.3}$
	S3DG	MSRVTT	$48.2_{\pm4.4}$	$60.4_{\pm3.9}$	$71.4_{\pm6.4}$	$58.1_{\pm8.2}$	$25.3_{\pm2.2}$	$61.8_{\pm8.4}$	$50.7_{\pm3.2}$	$54.7_{\pm2.9}$
	Scene	MSRVTT	$50.7_{\pm1.6}$	$65.0_{\pm4.7}$	$78.6_{\pm3.6}$	$\mathit{67.3}_{\pm3.9}$	$34.5_{\pm4.6}$	$71.7_{\pm1.8}$	$54.6_{\pm1.5}$	$58.4_{\pm1.3}$
	Audio	MSRVTT	$50.0_{\pm1.1}$	$63.7_{\pm6.1}$	$72.7_{\pm3.0}$	$61.9_{\pm6.5}$	$34.5_{\pm2.3}$	$68.0_{\pm5.9}$	$52.8_{\pm1.3}$	$56.7_{\pm1.4}$
	OCR	MSRVTT	$48.3_{\pm8.3}$	$57.1_{\pm4.6}$	$\mathit{76.9}_{\pm0.6}$	$60.7_{\pm4.9}$	$33.9_{\pm8.3}$	$72.1_{\pm4.4}$	$51.0_{\pm3.0}$	$55.5_{\pm3.0}$
	Face	MSRVTT	$46.5_{\pm6.8}$	$61.3_{\pm3.6}$	$71.5_{\pm7.1}$	$60.8_{\pm11.0}$	$30.9_{\pm3.4}$	$68.4_{\pm6.0}$	$50.5_{\pm2.6}$	$54.5_{\pm2.6}$
	Speech	MSRVTT	$48.5_{\pm7.6}$	$60.7_{\pm3.5}$	$74.5_{\pm5.7}$	$62.6_{\pm6.2}$	$27.3_{\pm1.8}$	$74.0_{\pm3.1}$	$51.7_{\pm2.6}$	$56.2_{\pm2.5}$
	Concat	MSRVTT	$43.6_{\pm6.0}$	$64.7_{\pm3.0}$	$68.5_{\pm8.0}$	$63.8_{\pm3.8}$	$32.0_{\pm5.5}$	$70.4_{\pm5.9}$	$49.8_{\pm4.1}$	$54.2_{\pm4.0}$
	MMC-PCFG	MSRVTT	$52.3_{\pm5.1}$	$\mathit{68.1}_{\pm2.9}$	$\mathbf{78.2}_{\pm1.9}$	$65.8_{\pm2.4}$	$32.0_{\pm2.0}$	$\mathbf{74.7}_{\pm2.3}$	$56.0_{\pm1.4}$	$60.0_{\pm1.2}$
	MMC-PCFG	DiDeMo	$61.8_{\pm7.7}$	$41.5_{\pm11.8}$	$64.6_{\pm5.2}$	$47.1_{\pm11.1}$	$30.5_{\pm7.1}$	$62.2_{\pm5.1}$	$49.6_{\pm1.4}$	$53.8_{\pm0.9}$
	MMC-PCFG	YouCook2	$40.7_{\pm14.9}$	$23.9_{\pm3.4}$	$59.9_{\pm10.2}$	$16.2_{\pm2.6}$	$14.5_{\pm4.0}$	$23.7_{\pm3.9}$	$34.0_{\pm6.4}$	$37.5_{\pm6.8}$
	C-PCFG	HT(29.6k)	$68.6_{\pm2.1}$	$25.1_{\pm4.8}$	$37.5_{\pm12.1}$	$33.4_{\pm3.6}$	$27.7_{\pm2.9}$	$41.9_{\pm5.0}$	$42.3_{\pm3.3}$	$46.0_{\pm3.1}$
	MMC-PCFG	HT(29.6k)	$70.8_{\pm2.7}$	$32.2_{\pm13.2}$	$48.6_{\pm6.3}$	$36.0_{\pm4.6}$	$32.0_{\pm2.1}$	$43.1_{\pm5.5}$	$47.2_{\pm3.8}$	$51.7_{\pm5.0}$
	PTC-PCFG	HT(29.6k)	$62.2_{\pm7.7}$	$54.0_{\pm13.0}$	$60.0_{\pm4.9}$	$53.0_{\pm3.6}$	$32.4_{\pm4.0}$	$45.9_{\pm5.1}$	$52.2_{\pm3.9}$	$57.4_{\pm5.1}$
	C-PCFG	HT(296k)	$75.5_{\pm1.4}$	$34.8_{\pm4.3}$	$58.6_{\pm1.6}$	$46.9_{\pm2.8}$	$40.0_{\pm3.1}$	$55.4_{\pm7.1}$	$52.0_{\pm1.3}$	$56.4_{\pm1.7}$
	MMC-PCFG	HT(296k)	$75.1_{\pm2.5}$	$39.4_{\pm15.5}$	$55.2_{\pm7.2}$	$40.0_{\pm4.5}$	$40.2_{\pm5.8}$	$51.0_{\pm6.4}$	$52.4_{\pm5.5}$	$56.8_{\pm6.4}$
	PTC-PCFG	HT(296k)	$70.2_{\pm5.8}$	$51.7_{\pm12.1}$	$64.5_{\pm6.2}$	$54.0_{\pm6.5}$	$39.2_{\pm3.2}$	$54.8_{\pm9.5}$	$55.7_{\pm5.0}$	$61.1_{\pm5.9}$
	C-PCFG	HT(592k)	$76.9_{\pm2.6}$	$35.4_{\pm8.3}$	$57.9_{\pm5.4}$	$44.5_{\pm10.2}$	$\mathbf{41.4}_{\pm3.9}$	$57.8_{\pm5.3}$	$52.5_{\pm3.4}$	$56.4_{\pm3.6}$
	MMC-PCFG	HT(592k)	$76.1_{\pm3.4}$	$46.3_{\pm20.0}$	$57.0_{\pm5.8}$	$50.1_{\pm10.1}$	$37.9_{\pm3.1}$	$52.8_{\pm4.6}$	$55.1_{\pm7.0}$	$60.2_{\pm8.0}$
	PTC-PCFG	HT(592k)	$74.0_{\pm3.6}$	$50.2_{\pm18.9}$	$67.0_{\pm4.1}$	$54.5_{\pm9.1}$	$34.7_{\pm2.4}$	$55.4_{\pm7.3}$	$\mathbf{57.4}_{\pm4.6}$	$\mathbf{62.8}_{\pm5.7}$
	C-PCFG	HT(1.2M)	$77.0_{\pm2.0}$	$30.5_{\pm10.4}$	$60.1_{\pm3.4}$	$41.5_{\pm11.1}$	$38.5_{\pm4.4}$	$52.2_{\pm4.8}$	$51.6_{\pm3.1}$	$55.5_{\pm3.5}$
	MMC-PCFG	HT(1.2M)	$\mathbf{77.9}_{\pm2.2}$	$44.2_{\pm13.1}$	$40.6_{\pm22.2}$	$45.6_{\pm5.5}$	$\mathit{40.5}_{\pm4.6}$	$56.3_{\pm6.8}$	$52.4_{\pm4.5}$	$57.2_{\pm5.1}$
	PTC-PCFG	HT(1.2M)	$72.3_{\pm1.6}$	$44.0_{\pm7.7}$	$70.2_{\pm4.6}$	$53.4_{\pm7.9}$	$34.4_{\pm4.6}$	$57.4_{\pm6.9}$	$55.6_{\pm2.5}$	$61.0_{\pm3.3}$
	C-PCFG	HT(2.4M)	$\mathit{77.2}_{\pm2.1}$	$31.1_{\pm4.6}$	$59.8_{\pm8.0}$	$43.3_{\pm9.7}$	$39.1_{\pm3.5}$	$54.5_{\pm6.4}$	$51.9_{\pm2.3}$	$55.4_{\pm2.8}$
	MMC-PCFG	HT(2.4M)	$76.2_{\pm2.2}$	$36.1_{\pm6.3}$	$62.5_{\pm7.5}$	$47.8_{\pm8.1}$	$40.0_{\pm4.5}$	$55.8_{\pm5.3}$	$53.5_{\pm2.4}$	$57.1_{\pm2.8}$
	PTC-PCFG	HT(2.4M)	$74.2_{\pm2.8}$	$46.0_{\pm7.1}$	$67.5_{\pm4.1}$	$52.7_{\pm8.6}$	$40.2_{\pm5.6}$	$58.3_{\pm10.7}$	$\mathit{56.6}_{\pm2.5}$	$\mathit{61.2}_{\pm2.9}$