

# Procedural Text Generation from Instructional Videos



**Taichi Nishimura**

Graduate School of Informatics  
Kyoto University

This thesis is submitted for the degree of  
*Doctor of Philosophy in Informatics*

August 2023





## Acknowledgements

It seems that my Ph.D. journey is finally over. When I entered the graduate school of Informatics at Kyoto University, I did not know anything about research. I learned its essence here, such as surveying existing work, designing experimental settings, writing efficient codes, rebutting reviewers' comments and publishing academic papers. Although I improved my research skills, there is still plenty of room for improvement in them—But, anyway, my Ph.D. life is coming to an end. Here, I would like to express my gratitude to many people.

I am first deeply grateful to my supervisor, Professor Shinsuke Mori. He has hospitably guided me since I was an undergraduate student at Kyushu University. His insightful and constructive comments lead me to have an opportunity to think deeply about my research.

I also wish to express sincere appreciation to my thesis committee, Professor Ko Nishino and Professor Yuichi Nakamura, for taking the time to review my thesis and giving many helpful comments.

I especially thank the main two co-authors in my papers: Dr. Atsushi Hashimoto and Dr. Yoshitaka Ushiku, who are working at OMRON SINIC X Corporation. They are familiar with vision-and-language research, and their comments gratefully lead my research to be successful. They have always discussed with me not only the main concept of my research but also its technical details. During my research, we shared joy and sorrow with each other. Many rejected papers we wrote together made me grow significantly; I sometimes remember that we stayed up all night to submit the ECCV'20 paper. Besides research topics, I enjoyed small talk with them. I will have never forgotten their cheerful support in my Ph.D. life.

Special thanks to other co-authors who I worked with. I thank Associate Professor Yoko Yamakata at The University of Tokyo. She supported me when I was an undergraduate student at Kyushu University. I am also deeply grateful to Dr. Jun Harashima, who supported me to build a new dataset. I also thank Dr. Katsuhiko Ishiguro, Dr. Keita Higuchi and Dr. Masaaki Kotera, who are mentors at the summer internship at Preferred Networks (PFN). I really enjoyed working with them. Based on the work, I could publish my paper at the international conference and surprisingly win the best paper award. I also thank Dr. Fumihito Ono and Ms. Natsuko Okuda at Osaka Medical and Pharmaceutical University. Based on their support, I could construct the BioVL2 dataset and publish it. Without their support, I

could not accomplish this work. Gladly, I could achieve the best paper award at the annual meeting of the Association for Natural Language Processing.

I am also thankful for the members at Mori Laboratory. Assistant Professor Hirotaka Kameko and Mr. Keisuke Shirai have discussed with me the detailed experimental settings and technical supports, such as how to implement the proposed method. I thank Professor Hiroaki Nanjo at Shiga University. His comments are always useful for improving my research. I also enjoyed the small talk with him. Other lab members helped me to move my research forward in many respects, and their insightful comments through the discussion were an enormous help to me. Especially, I thank the students I mentored during my Ph.D. course: Mr. Kojiro Sakoda, Kento Tanaka, Komei Hoshijima, Sho Kinoshita, Kosuke Morita, Jieyong Zhu, Keyaki Ohno, Tomoya Yoshida, Koki Yamamoto, Keiya Kajimura and Yuto Haneji. In addition to the research discussion, we enjoyed the small talk, which significantly relieved my stress. I am also thankful for the members of the PFN Smashbros group. I enjoyed playing the game with them all.

This thesis could not be completed without financial support. I am deeply grateful to JSPS doctoral fellowship (DC1), which totally supports my research life. This is my first grant and I remember that I had a hard time writing the proposal. Dr. Hiroshi Anzai, a friend I met when I was an undergraduate student at Kyushu University, supported me to write proposals by reading and making a lot of insightful comments on my proposal. I really appreciate your help, which definitely leads me to obtain the grant. In addition, I enjoyed talking about various topics with him, such as our research and future plans.

I wish to express my gratitude to my family, Yuichi, Hisami and Minoru. Without their financial and mental support, I could not complete my Ph.D. course. Finally, I wish to express my deepest gratitude to my wife, Kokoro. Without her, I absolutely could not complete my Ph.D. I hope we stay together forever and ever.

## Abstract

The overarching goal of this thesis is to develop an integrated intelligence that processes procedural text and instructional videos to assist people in their work. To achieve this goal, researchers have constructed multimedia archives for high-level video processing and proposed methodologies to utilize them to support human activities.

With the rapid growth of deep learning, computers have become capable of linking with different media using neural networks, leading to the emergence of novel multimedia applications. Among them, video captioning is a remarkable approach, which aims at generating descriptive sentences from the input videos. If we can extend this technology to generate procedural text from instructional videos, a broad impact is expected on the multimedia community. For example, it contributes to expanding the multimedia archives by converting video content into textual symbols directly without preparing the triplets of videos, transcription and procedural text that previous video-text alignment research utilized.

Motivated by this, in this thesis, we tackle procedural text generation from instructional videos. Given the videos, our task requires computers to (1) extract key events that are essential to achieve a task and (2) generate sentences that reflect on their contents. The key challenges of the task are three-fold: (1) developing models capable of generating accurate procedural text based on video content, (2) learning both the event extractor and sentence generator in a story-aware manner and (3) extending our research focus from everyday to important domains that are in high demand for video verbalization.

Challenge (1) is important to develop reliable systems that can generate accurate procedural text (Chapter 3). It consists of two sub-tasks: (a) recognizing actions and materials in the events and (b) generating sentences by considering the dependency of the actions and materials throughout the entire video. While practical performance on (a) was achieved in multimedia food computing research, (b) is still challenging because models should be robust to recognize material state changes, which are often accompanied by drastic appearance transformations. For instance, while preparing scrambled eggs, we recognize the eggs through their changes in form from raw to cracked and finally stirred. We observe that the general video captioning models are not capable of tracking them, failing to generate accurate procedural text. To address this issue, we propose a novel approach to represent the material state

changes as the transitions of material vectors in the latent feature space. Our experimental results show that it can generate more accurate procedural text than state-of-the-art video captioning models. In addition, the qualitative evaluation demonstrates that they can model the material state changes like humans and generate procedural text based on them.

Challenge (2) is essential to achieve efficient comprehension of videos for humans (Chapter 4). This task is similar to dense video captioning (DVC), which aims at detecting events thoroughly and generating sentences describing them. However, the state-of-the-art DVC models adopted parallel prediction on events and sentences independently, leading to redundant outputs, which are not suitable for humans to grasp the video contents. In contrast, our task requires the models to extract the appropriate numbers of events in the correct order and generate sentences based on them. In this thesis, we refer to this characteristic as the story of instructional videos. To achieve this, we propose a multimodal recurrent learning approach of the event extractor and sentence generator. It predicts events and generates sentences recurrently by memorizing and mixing the previous prediction on both events and sentences, which encourages the model to achieve story awareness. Our experimental results show that the proposed method achieves better performance than DVC models and confirm that it can extract the appropriate number of events in the correct order in a story-aware manner.

Challenge (3) is crucial in order to spread the developed system widely (Chapter 5). Previously, many researchers focused on the cooking domain as the benchmark because of its easiness of constructing large-scale video and text pairs collected from the web. In addition, the cooking domain also has the merit to cover a large variety of actions and objects. More recently, several researchers started to collect instructional videos on a large variety of everyday topics, such as DIY and house gardening. However, the covered domains are typically limited to everyday topics, lacking the motivation to verbalize the video content. Towards real-world applications, it is crucial to explore new domains that are highly demanded from a perspective of reproducibility. To achieve this, we select the biochemical domain and construct the first, egocentric biochemical video-and-language dataset (BioVL2), which contains four basic biochemical experiments with eight videos per experiment, summing 32 videos in total. Based on it, we tackled the task of generating protocols from experiment videos. We apply the existing procedural text generation method and analyze the model's behavior (i.e., strength and weakness) for details. In addition, from our lessons, we provide guidelines for dataset construction.

Finally, we conclude our work and discuss future research directions (Chapter 6).

# Table of contents

<b>List of figures</b>	<b>xi</b>
<b>List of tables</b>	<b>xv</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Background and motivations . . . . .	1
1.2 Challenges . . . . .	4
1.2.1 Developing models capable of generating accurate procedural text based on video content . . . . .	4
1.2.2 Learning both the event extractor and sentence generator in a story- aware manner . . . . .	5
1.2.3 Extending our research focus from everyday to important domains that are in high demand for video verbalization . . . . .	6
1.3 Contributions . . . . .	7
1.3.1 Imitating human-like understanding of the material state changes for accurate procedural text generation . . . . .	7
1.3.2 Multimodal recurrent learning of the event selector and sentence generator . . . . .	8
1.3.3 Exploring important domains in terms of practical applications: a case study on the biochemical domain . . . . .	8
1.4 Thesis outline . . . . .	9
1.4.1 Introduction (Chapter 1) . . . . .	9
1.4.2 Related work (Chapter 2) . . . . .	9
1.4.3 State-aware procedural text generation from segmented key events (Chapter 3) . . . . .	9
1.4.4 Multimodal recurrent learning of the event selector and sentence generator (Chapter 4) . . . . .	10

1.4.5	BioVL2: Egocentric biochemical video-and-language dataset (Chapter 5)	10
1.4.6	Conclusion (Chapter 6)	10
<b>2</b>	<b>Related Work</b>	<b>11</b>
2.1	Multimedia research that targets procedural text and instructional videos	11
2.2	Vision and language	14
2.2.1	Tasks, methods and datasets	15
2.3	Vision and language in instructional domains	18
2.3.1	Multi-modal understanding for procedural text with vision	19
2.3.2	Procedural text generation from visual observations	20
2.3.3	Multi-modal instructional video datasets	21
<b>3</b>	<b>State-aware Procedural Text Generation from Segmented Key Events</b>	<b>25</b>
3.1	Introduction	25
3.2	Proposed method	26
3.2.1	Overview	27
3.2.2	Encoder	28
3.2.3	Visual simulator	29
3.2.4	Decoder	31
3.2.5	Textual re-simulator	33
3.2.6	Loss functions	33
3.3	Experiments	34
3.3.1	Experimental settings	35
3.3.2	Word-overlap evaluation	38
3.3.3	Ingredient prediction	40
3.3.4	Retrieval evaluation	43
3.3.5	Qualitative analysis	43
3.3.6	Discussion of the learned embedding	44
3.3.7	Experiments on the full prediction setting	47
3.4	Conclusion	49
<b>4</b>	<b>Multimodal Recurrent Learning of the Event Selector and Sentence Generator</b>	<b>51</b>
4.1	Introduction	51
4.2	Oracle-based analysis of the existing DVC model	54
4.2.1	Quantitative evaluation	56
4.2.2	Qualitative evaluation	57

---

4.3	Proposed method . . . . .	57
4.3.1	Event selector . . . . .	60
4.3.2	Sentence generator . . . . .	61
4.3.3	Multimodal memory mixing . . . . .	61
4.3.4	Loss functions . . . . .	62
4.4	Extended model . . . . .	62
4.4.1	Dot-product visual simulator . . . . .	64
4.4.2	Textual attention . . . . .	66
4.4.3	Loss functions . . . . .	67
4.5	Experiments . . . . .	69
4.5.1	Word-overlap evaluation . . . . .	71
4.5.2	Discussion on the number of predicted events . . . . .	71
4.5.3	Qualitative analysis . . . . .	72
4.5.4	Discussion on the detailed model settings . . . . .	73
4.6	Conclusion . . . . .	76
<b>5</b>	<b>BioVL2: Egocentric Biochemical Video-and-Language Dataset</b>	<b>79</b>
5.1	Introduction . . . . .	79
5.2	BioVL2 dataset . . . . .	82
5.2.1	Dataset construction . . . . .	82
5.2.2	Statistics . . . . .	85
5.2.3	Annotation agreement . . . . .	87
5.3	Protocol generation from experiment videos . . . . .	89
5.3.1	b-NE sequence construction . . . . .	91
5.3.2	Generating protocol candidate sentence from b-NE sequence . . . . .	91
5.3.3	Protocol generation . . . . .	94
5.4	Experiments . . . . .	94
5.4.1	Experimental settings . . . . .	94
5.4.2	Quantitative evaluation . . . . .	96
5.4.3	Qualitative evaluation . . . . .	97
5.4.4	Discussion . . . . .	98
5.4.5	PMI-based analysis on the language model between objects and actions	98
5.4.6	Relationship between the number of steps and word-overlap evaluation	99
5.4.7	Future development of the BioVL project . . . . .	100
5.5	Conclusion . . . . .	100

<b>6 Conclusion</b>	<b>103</b>
6.1 Summary . . . . .	103
6.2 Limitation . . . . .	104
6.3 Future work . . . . .	105
6.3.1 Build accurate models for untrimmed egocentric videos . . . . .	105
6.3.2 Build domain-agnostic models to generate procedural text from instructional videos . . . . .	105
6.3.3 Transfer the learned representations into other tasks . . . . .	106
<b>References</b>	<b>107</b>



# List of figures

1.1	An overview of our task in this thesis. . . . .	2
1.2	An overview of challenge (1) and its solution. . . . .	4
1.3	An overview of challenge (2) and its solution. . . . .	5
1.4	An overview of challenge (3) and its solution. . . . .	6
2.1	A prototype screen of the Infomedia project quoted from Figure 2 in [22]. . .	12
2.2	An overview of the vision-and-language tasks. Example images and videos are used from the MSCOCO [68] and YouCook2 [147] datasets. . . . .	16
3.1	Concept of the proposed method. Given key events and material list, the proposed method generates a procedural text by reasoning the state transition of materials at each step. . . . .	26
3.2	An overview of the proposed method. To track material states in key events, we incorporate the visual simulator $R_v$ into the transformer-based encoder-decoder architecture ( $E$ and $D$ ). In addition, based on our intuition that the state transition of materials is traceable from the generated procedural texts, we attach the textual re-simulator $R_t$ to the model. . . . .	27
3.3	An overview of the (visual) simulator. The simulator recurrently reasons the state transition of the materials at each step. Specifically, it predicts executed actions and involved materials in (1) the action and (2) material selector and then updates the state of materials in (3) the updater. The updated materials are forwarded to the next step. The textual re-simulator has the same modules.	27
3.4	An overview of the material and key event encoders. . . . .	28
3.5	An overview of the decoder. . . . .	32
3.6	A screen of our browser-based web annotation tool. Annotators write ingredients that appear in the recipes. To ease annotation, we estimated ingredients using the NER method [1] pre-trained on the English recipe flow graph corpus [138], and we set the default values of inputs. . . . .	36

3.7	An overview of our baseline +ingredients (-I) implementation. These models also incorporate the material encoder described in Chapter 3.2.2 and copy mechanism into the baselines. . . . .	37
3.8	An example of videos taken from different viewpoints: exocentric and egocentric views. . . . .	40
3.9	Examples of generated recipes. Here, we compare four models, MART-I (baseline), VI, VIV and VIVT with the ground truth. Green bold and red words represent semantically correct and incorrect ingredients, respectively. Words in parentheses indicate missing ingredients, which should be included in the sentence. Note that parallel words in a sentence are not separated from the commas in the YouCook2 dataset (see step 1 in (a) in the ground truth).	41
3.10	An annotation example of the visual simulator. . . . .	44
3.11	Learned embedding of ingredients obtained by the VIVT model. Note that only raw and updated (the attention weight in the material selector is higher than 0.5) ingredients are transformed by t-SNE [120]. Red and blue colors represent the raw and updated ingredients, respectively. . . . .	46
3.12	Arithmetics using the learned embedding of ingredients. Examples (a) to (d) and (e) to (g) represent the first-order (raw-to-updated) and second-order (updated-to-updated) transformations, respectively. (d) and (g) show the failure cases for each transformation. . . . .	47
3.13	An overview of how to integrate ingredient decoder into the model. . . . .	48
4.1	A conceptual comparison of our approach and existing DVC studies. While the existing DVC models adopted parallel prediction, our approach employ multimodal recurrent prediction, which estimates events and sentences by memorizing and fusing the previously prediction results. . . . .	52
4.2	tIoU distribution of oracle events on the training and validation sets of YouCook2. . . . .	56
4.3	Comparison of the procedural texts generated by the oracle selection and ground truth. $N = 100$ , which is a default hyper-parameter of PDVC, is used in this example. . . . .	57
4.4	An introductory overview of our approach. Unlike the previous DVC approaches, we propose a multimodal recurrent learning approach to train the event selector and sentence generator. Both modules represent the previously predicted events and sentences as memory vectors and predict the next step. These memory vectors are updated and mixed to effectively share the previous prediction belonging to different modalities. . . . .	58

---

4.5	Multimodal recurrent learning approach of the event selector and sentence generator for recipe generation from unsegmented cooking videos. The event selector chooses oracle events from event candidates repeatedly (Chapter 4.3.1) and the sentence generator outputs sentences for the selected events (Chapter 4.3.1). The memories are updated and mixed to effectively remember the history of the events/sentences for predicting the next step (Chapter 4.3.3). . . . .	59
4.6	An overview of the extended model for procedural text generation from unsegmented instructional videos. To generate more accurate procedural text, it has additional two modules: (1) dot-product visual simulator and (2) textual attention. The dot-product visual simulator is introduced to learn the state transition of materials. The textual attention module encourages the sentence generator to verbalize actions and materials more accurately. . . .	63
4.7	An overview of the extended dot-product visual simulator that reasons about the state transition of materials. It has three components: (1) action selector, (2) material selector and (3) updater. The dot-product attention is employed to treat the event candidates. . . . .	64
4.8	An overview of loss computation of visual selector loss and textual attention loss. . . . .	68
4.9	Histogram of the number of predicted events and ground truth. . . . .	72
4.10	Examples of the generated procedural texts. We compare three models: PDVC, B and BIVT with the ground truth. . . . .	73
5.1	An overview of the BioVL2 dataset, which consists of two types of video-and-language annotations: (a) alignment between events and sentences and (b) bounding boxes for objects that appear in the protocols. . . . .	80
5.2	“まほろ,” a robot conducting biomedical experiments developed by National Institute of Advanced Industrial Science and Technology. The proto is quoted from <a href="https://www.aist.go.jp/sst/ja/exhibition/innovation_zone/zone11/index.html">https://www.aist.go.jp/sst/ja/exhibition/innovation_zone/zone11/index.html</a> (accessed 5/21, 2023). . . . .	81
5.3	(a) and (b) shows the recording studio of experiments and the view from the equipped first-person camera, respectively. . . . .	82
5.4	A screen of the bounding box web annotation tool. (A), (B) and (C) shows the objects in the protocols, annotation bounding box pane and operation pane, respectively. . . . .	84
5.5	Distribution of event length $d$ ( $d$ represents seconds). . . . .	87

---

5.6	(a) and (b) shows AP and mean of APs (mAP) and the number of true positives and false positives, respectively. . . . .	89
5.7	A comparison of frames between manipulating Primer1 and Primer2. . . . .	90
5.8	An overview of the protocol generation from experiment videos. . . . .	91
5.9	Transformer-based sentence generation model. Given objects, the model is trained to generate corresponding sentences. . . . .	92
5.10	Examples of the generated protocols and the ground-truth on PCR. The model in this figure is “Reagent, Location and Device (w/ #word filtering).” We also show the selected frames with b-NE annotation information. Note that the registered trademark symbol ®is shown in the figure, but the generated protocols do not contain them. . . . .	97
5.11	The experiment environments, where QR codes ®are attached to objects used in the protocol. . . . .	101

# List of tables

2.1	Comparative overview of multi-modal instructional video datasets. . . . .	21
3.1	YouCook2-ingredient+ dataset statistics. . . . .	35
3.2	Paragraph- and sentence-level word-overlap evaluation for the baseline and the proposed models with ablation studies. The scores in bold are the best among the comparative models. “I” indicates whether the model uses ingredient information or not. B=BLEU, M=METEOR, C=CIDEr-D, RL=ROUGE-L. . . . .	38
3.3	Change in paragraph-level word-overlap evaluation with controlled $\lambda$ . . . . .	39
3.4	Change in paragraph-level word-overlap evaluation on different viewpoints: egocentric, exocentric and mixed views. . . . .	39
3.5	Results of ingredient prediction. . . . .	42
3.6	Results of retrieval evaluation. $\downarrow$ indicates that lower is better. . . . .	42
3.7	Quantitative evaluation of the visual simulators. . . . .	44
3.8	Paragraph- and sentence-level word-overlap evaluation on the full prediction setting. . . . .	48
3.9	Results of ingredient prediction on the full prediction setting. . . . .	49
3.10	Results of retrieval evaluation on the full prediction setting. . . . .	49
3.11	Performance of an ingredient decoder on the full prediction setting. Note that we compute the micro-recall, precision, and F1 on the multi-label classification setting. . . . .	50
4.1	Word-overlap metrics of the oracle selection on the YouCook2 dataset. $N$ represents the number of candidate events, a hyper-parameter of PDVC. The bold scores are the best among the comparative settings. . . . .	55
4.2	Word-overlap metrics for the baseline and proposed method. The bold scores are the best among the comparative methods. . . . .	71

4.3	Percentage of procedural texts that satisfy $ p - q  \leq \eta$ , where $p, q, \eta$ represents the number of predicted events, ground-truth events, and a threshold, respectively. In this experiment, we change $\eta$ from 0 to 3. . . . .	72
4.4	Loss ablation studies. MS, AS, MA, and AA represent material selection, action selection, material attention, and action attention losses, respectively.	74
4.5	Comparison of memory update strategies: separate vs joint. . . . .	74
4.6	Comparison of input modalities: video only and multimodal versions. . . .	75
4.7	Comparison of the model’s performance when varying the event encoders: TSN and MIL-NCE. Note that unlike TSN, which is pre-trained on only vision resources, the MIL-NCE is pre-trained on instructional vision-and-language resource, Howto100M. . . . .	75
4.8	Comparison of the model’s performance when varying the number of the event candidates $N$ . . . . .	76
5.1	An example of the annotation for PCR. The values in the table represent seconds. Note that @is written in the table, but not included in the real dataset.	83
5.2	Statistics of text annotations. The average and standard deviation (std) are written in the table. Note that in miniprep and agarose gel creation, several steps are skipped depending on the situation. Thus, note that their standard deviations are not equal to 0. . . . .	85
5.3	The number of unique objects and verbs, which do not appear in the other kinds of protocols. . . . .	85
5.4	Statistics of video length. The table values show mean and standard deviations.	86
5.5	Statistics of frames that has bounding box annotations. . . . .	86
5.6	Statistics on unique objects associated with bounding box annotations after classifying them into b-NE tag category types. Although among b-NE tags, object-based NEs are Reagent, Location, Device, and Seal, Seal does not appear in the BioVL2 dataset. Thus, we count Reagent, Location, Device in the table. . . . .	88
5.7	Statistics on bounding box annotations. . . . .	88
5.8	Agreement of event-and-sentence alignments. The mean of tIoU is shown in the table. . . . .	89

---

5.9	Statistics on the WLP dataset, including three filtering settings in the table. The first one is using all of the b-NE tags, the second is using only the object-based NEs, and the third is using only the Reagent and Location tags in the WLP dataset. For each case, we also show the statistics, where more than 20 words are filtered. We use the original dataset split to the original WLP dataset because BioVL2 is used for the test set. . . . .	95
5.10	Word-overlap evaluation. The bold scores represent the best among the methods. . . . .	96
5.11	The most frequent top-10 verbs and objects which has the highest PMIs for the verbs on the training set of the WLP dataset. The PMI scores overestimate the low frequency words, thus we report the objects which appear more than 10 times in the dataset. . . . .	98
5.12	The most frequent top-10 objects and verbs have the highest PMIs for the verbs on the training set of the WLP dataset. The PMI scores overestimate the low frequent words, thus we report the verbs which appear more than 10 times in the dataset. . . . .	99
5.13	Relationship between the number of steps in the protocols and the word-overlap evaluation results. The model we use in this table is “Reagent, Location, and Device (w/ #word filtering).” . . . . .	100





# Chapter 1

## Introduction

### 1.1 Background and motivations

What Descartes did was a good step. You have added much several ways, and especially in taking the colours of thin plates into philosophical consideration. If I have seen a little further it is by standing on the shoulders of giants.

Isaac Newton, *Letter to Robert Hooke*.

Human beings have accumulated knowledge by standing on the shoulder of giants. We have invented based on the existing ideas and written down the procedures to reproduce the work. Procedural text (e.g., recipes and scientific protocols) is sequential instructions written in natural language and contains actions, objects and supplementary information (e.g., time and quantity) to achieve a specific task. It plays an important role in telling our procedural knowledge to others efficiently. We acquire new skills by reading and executing it, come up with new ideas and yield inventions to develop our civilization.

Another method of documenting tasks is recording the work as videos. While text formats are suitable for describing quantitative information (e.g., time and quantity), videos help convey qualitative information (e.g., color and shape). For instance, instead of explaining the term “golden brown” to a cooking beginner, visual information can be used to intuitively demonstrate the concept. These advantages lead to the abundance of instructional videos uploaded on video-sharing platforms (e.g., YouTube), which greatly assist us in our procedural tasks.

The overarching goal of this thesis is to develop an integrated intelligence that processes video and language from different modalities to assist people in their work. This research topic has been discussed in the multimedia community for over 30 years. The motivation is to construct intelligent multimedia archives, which understand the users’ demands from their

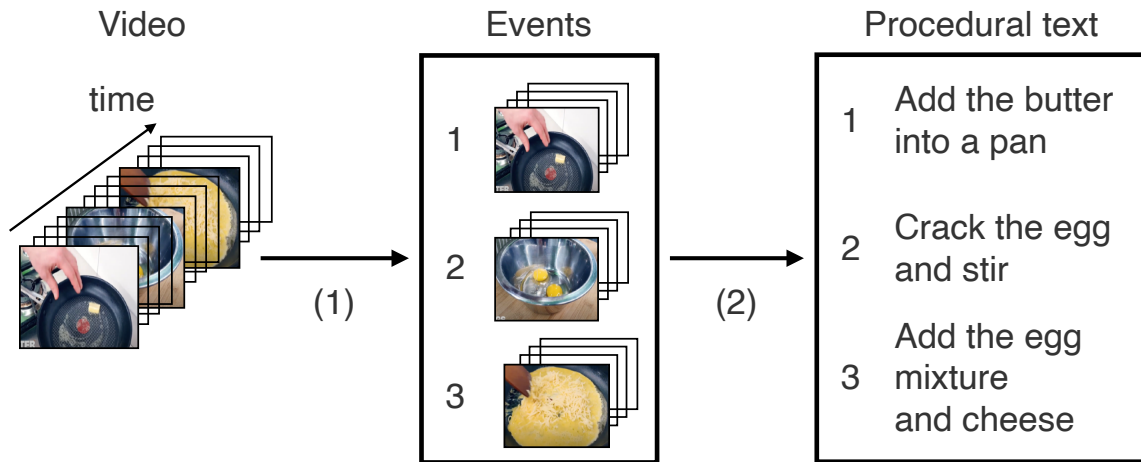


Fig. 1.1 An overview of our task in this thesis.

textual queries and provide the corresponding video information, such as retrieving a video from a video database or searching specific scenes, objects and actions from the videos.

Towards this objective, in the 1990s, Infomedia [126] was introduced at the forefront of that era. It pioneered the association of videos and audio with text and laid the foundation for constructing multimedia archives for such high-level video processing. In the early 2000s, a research group led by Hamada *et al.* [41, 43, 42, 78] focused on the cooking domain and developed multimedia archives by proposing methods to align video segments with recipe steps. In addition, based on the constructed archives, they also proposed a smart kitchen that assists people to cook by providing both visual and textual information on the next step. The core concept of the smart kitchen has been followed by successor researchers [46, 86, 45].

From the mid-2000s to early 2010s, following the success of statistical machine learning (ML), much research on unsupervised video-text alignment research emerged. Shibata *et al.* [109] utilized structural representations of recipes and proposed a method to align video and text using hidden Markov models (HMMs). Several researchers subsequently embraced this idea to achieve video narration alignment [10, 72] as well as video-(verb, noun) alignment [81, 82]. However, the integration of video and language processing was not yet fully established due to a lack of methods for embedding the feature space that captures common high-level semantics from low-level features of vision and language. Therefore, they utilized transcription (e.g., narrations and closed captions) as a cue to align video and text, leveraging two merits: (1) timestamps that correspond to the videos and (2) ease of computing similarity to procedural text.

With the rapid advancement of deep learning [64], computers have become capable of constructing the joint representation space to connect different modalities, leading to the development of novel multimedia applications. Among them, video captioning [123], a task of generating descriptive sentences from input videos, made remarkable progress recently. Before the deep-learning era, traditional methods adopted the pipeline approach of object detection and template-based language models but failed to achieve satisfactory performance due to the accumulated errors [39, 5]. With the benefit of deep neural networks, the current video captioning research can jointly optimize these modules in an end-to-end manner, achieving practical performance on short video datasets. However, it is still challenging how to develop a new video captioning model on long, instructional videos that contain a large variety of actions and objects.

If we can extend this technology to generate procedural text from instructional videos, its potential impact is substantial for both academic and industrial communities. In academia, video captioning greatly contributes to expanding multimedia archives because it can link video and text by directly converting video content into textual symbols. This point represents a clear advancement from previous video-text alignment [109, 108, 2, 72], which requires the triplets of a video, transcription, and procedural text. The accumulation of generated video-text pairs would provide a foundation for future interdisciplinary research between computer vision (CV), natural language processing (NLP) and robotics, extending beyond the multimedia community. In industry, the generated procedural text enables us to comprehend an overview of instructional videos by providing multimedia summarization through pairs of videos and sentences. This ability is also useful for ensuring the reproducibility of their work by assisting people to write procedural text.

Motivated by this, in this thesis, we focus on procedural text generation from instructional videos. We formulate this as a two-stage prediction task (Fig. 1.1), which requires computers to (1) extract key events that are essential to complete the goal and (2) generate sentences that reflect their contents. Although the definition of events varies in CV literature [57, 147] according to task settings, in this thesis, we define *events* as specific segments in a video that humans want to focus on to accomplish a particular task. Although our work is an extension of video captioning research, it contains unique challenges specifically tailored to procedural text and instructional videos. In the subsequent section, we will describe these distinctive challenges.

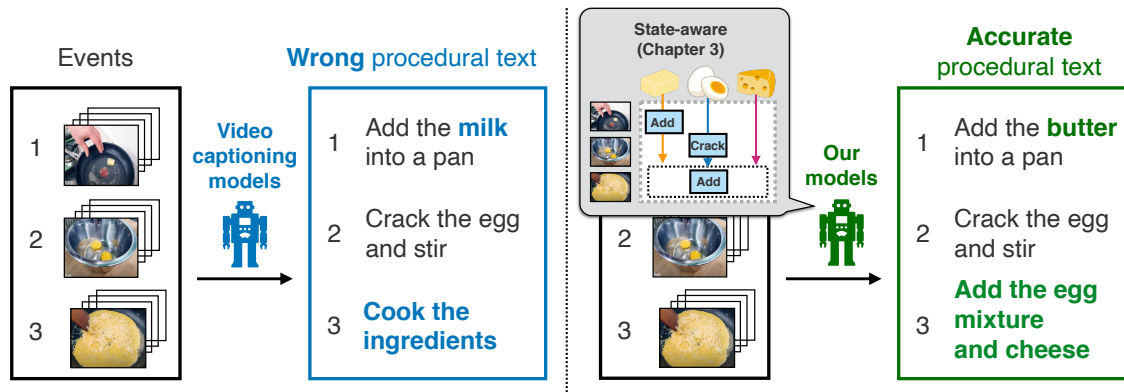


Fig. 1.2 An overview of challenge (1) and its solution.

## 1.2 Challenges

The key challenges of the task are three-fold: (1) developing models capable of generating accurate procedural text based on video content, (2) learning both the event extractor and sentence generator in a story-aware manner and (3) extending our research focus from everyday to important domains that are in high demand for video verbalization. Figs 1.2, 1.3 and 1.4 show an overview of these challenges and solutions.

### 1.2.1 Developing models capable of generating accurate procedural text based on video content

Challenge (1) is important to develop reliable systems that can generate accurate procedural text. It consists of two sub-tasks: (a) recognizing actions and materials in the events and (b) generating sentences by considering the dependency of the actions and materials throughout the entire video. (a) has been treated in multimedia food computing research. Several researchers have constructed video datasets to recognize actions/materials and proposed methods to predict them accurately [27, 46, 59]. Owing to their effort, practical performance on event-level recognition has been achieved.

It is still challenging to tackle (b) because models should be robust to recognize material state changes, which are often accompanied by drastic appearance transformations [139, 3]. For example, when cooking scrambled eggs, the eggs are transformed from raw to cracked, added, and finally stirred form. While humans can easily identify the materials after manipulations, computers cannot. We observe that the general video captioning models lack this ability, failing to generate accurate sentences for the events. They miss and hallucinate materials (e.g., in step 1 in Fig. 1.2, say “milk” instead of the accurate objects “butter”),

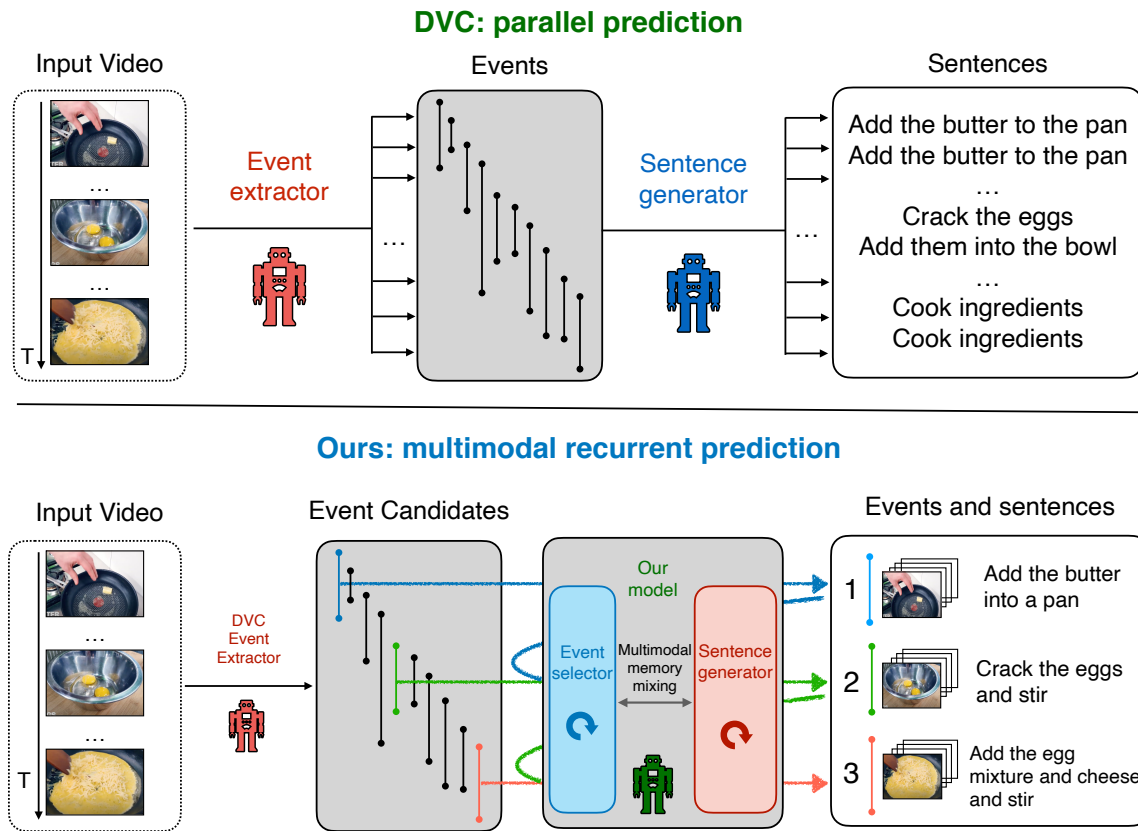


Fig. 1.3 An overview of challenge (2) and its solution.

or output abstract sentences (e.g., say “cook the ingredients”). This impels us to develop a model that acquires a human-like understanding of material state changes to generate accurate procedural text from instructional videos. We address this problem in the context of generating procedural text based on key events in videos and the associated materials.

### 1.2.2 Learning both the event extractor and sentence generator in a story-aware manner

Challenge (2) is essential to achieve efficient comprehension of videos for humans. The task is to train event extraction and sentence generation jointly, and each task was handled in the previous research in CV and NLP literature. They are called as temporal event localization [110, 21] and video captioning [123, 148, 65], respectively. The straightforward method is to pipeline them, but good performance cannot be achieved because of the accumulated errors of the individual components.

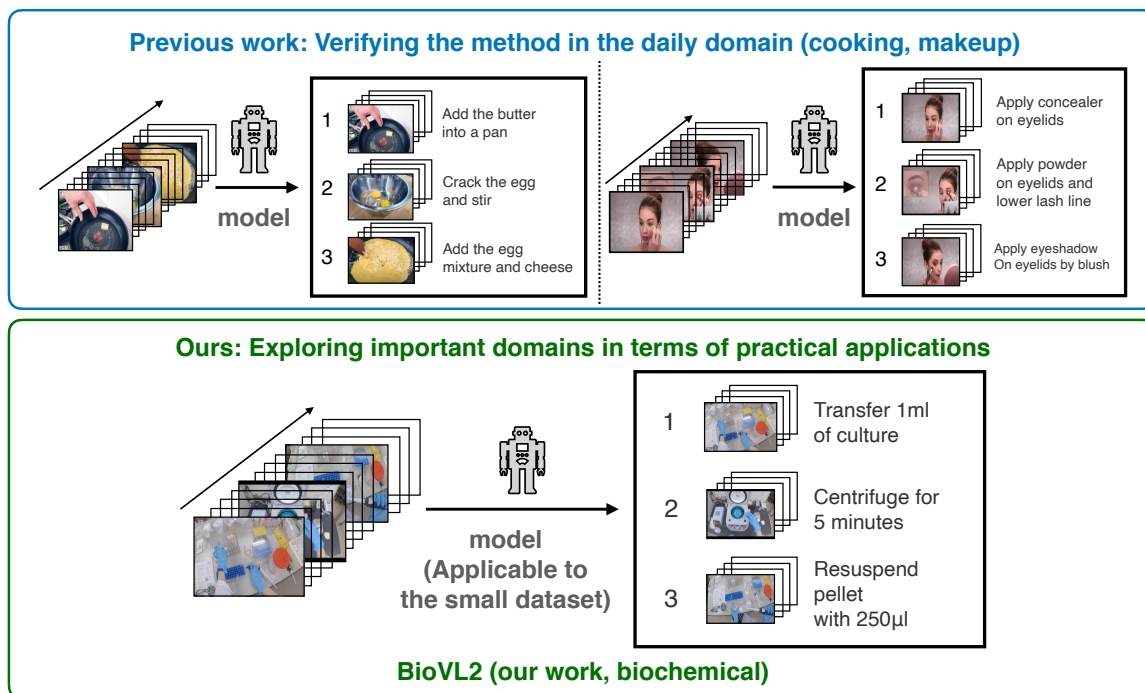


Fig. 1.4 An overview of challenge (3) and its solution.

To address this issue, Krishna *et al.* [57] proposed dense video captioning (DVC), which optimizes both modules to detect events thoroughly and generate sentences for them. To achieve dense prediction on events and sentences, the common DVC approaches adopted parallel prediction, where events and sentences are predicted independently (Fig. 1.3 top). This leads to redundant outputs but is reasonable for the DVC objective of thoroughly detecting events and generating sentences for multimedia retrieval. However, it is not suitable for our procedural text generation task because we place high importance on providing an overview of videos that can be easily grasped by humans. Our task requires the models to extract the appropriate numbers of events in the correct order and generate sentences based on them. In this thesis, we refer to this characteristic as the story awareness of procedural text, and it remains an unresolved problem to train both modules jointly while considering this.

### 1.2.3 Extending our research focus from everyday to important domains that are in high demand for video verbalization

Challenge (3) is crucial in order to spread the developed system widely. Previous work mostly focused on the cooking domain as the benchmark because of its easiness of constructing

large-scale video and text pairs collected from the web. In addition, the cooking domain also has the merit to cover a large variety of actions and objects [99, 27].

The research community found that it remains uncertain whether the methods developed for a single domain are effective in other domains. Exploring other domains beyond cooking is necessary for the research community to verify the universality of their developed approaches. To address this, recently, several researchers started to collect instructional videos in a large variety of domains, such as DIY, makeup and house gardening [75, 9, 131] (Fig. 1.4 top). COIN [116] is the largest annotated dataset, which consists of 11,827 web videos covering 180 different tasks and manual annotations of events and sentences.

Nevertheless, the covered domains are typically limited to everyday topics, lacking the motivation to verbalize the video content. To address this issue, it is crucial to explore new domains that are highly demanded from a perspective of reproducibility, such as biochemical experiments and industrial manufacturing. Due to the lack of web videos in such domains, it is difficult to construct a large-scale video dataset automatically. Therefore we have to cope with two issues: (a) how to prepare the dataset by designing a scalable video collection framework and (b) how to achieve procedural text generation models on the limited dataset.

## 1.3 Contributions

To achieve the above challenges, our contributions are the following three-fold.

### 1.3.1 Imitating human-like understanding of the material state changes for accurate procedural text generation

To imitate the human-like understanding of the material state changes, we get inspiration from natural language understanding (NLU), which aims at acquiring human-like reading comprehension ability from documents [96, 137]. The procedural text has long been a popular target for NLU, and two broad approaches have been proposed: representing state changes as graph structures [53, 54, 80] and latent feature transitions [12, 40, 4].

In this thesis, we focus on the latter approach by extending it to the multi-modal version and incorporating it into procedural text generation models. We call this *state-aware* approach, which can represent the material state changes in the latent space to generate procedural text in an end-to-end manner. Compared with the graph-based approach [85, 89], this method has two merits: (1) it is trainable without any manual annotations and (2) it is less likely to propagate the failure of this module to the downstream process because of its soft prediction. In our experiments, we confirm that they perform better than the current state-of-the-art

video captioning models. The qualitative evaluation demonstrates that these models generate accurate procedural text by modeling the material state changes.

### **1.3.2 Multimodal recurrent learning of the event selector and sentence generator**

Although the events predicted by DVC models are redundant, we observed that (1) several events are adoptable as a story of procedural text, but (2) the generated sentences for such events are not grounded well to the visual contents (i.e., materials and actions in the sentences are incorrect). We confirm this by analyzing the outputs of the state-of-the-art DVC model and set our goal to obtain correct procedural text by selecting oracle events from the event candidates and re-generating sentences for them.

To achieve this, the crucial idea is multimodal recurrent prediction, which estimates the next step by understanding previously predicted events and generated sentences (Fig. 1.3 bottom). We realize this idea by proposing a multimodal recurrent learning approach of the event extractor and sentence generator. The advantage of this approach is that the modules can memorize and mix the history of the previously predicted events and sentences, which facilitates the model to acquire story awareness. Our experimental results show that the proposed method achieves better performance than the DVC models. In addition, we confirm that it can grasp the story of procedural text; that is, it extracts the appropriate number of events in the correct order and generates accurate procedural text that reflects the event contents.

### **1.3.3 Exploring important domains in terms of practical applications: a case study on the biochemical domain**

To address the challenge (3), we focus on the biochemical domain because this domain seriously suffers from the reproducibility problem [7] and it is highly needed to generate procedural text to resolve it (Fig. 1.4 bottom). The main issue is no public biochemical video-and-language datasets. In addition, it is difficult to prepare the datasets automatically due to the poor resources on the web.

To address this, we construct the BioVL2, egocentric biochemical video-and-language dataset, which consists of eight egocentric videos for four types of experiments with text protocols and two types of video-and-language annotations. Here, we selected the first-person cameras because they are suitable for a scalable video collection framework. Unlike third-



person cameras that require wet-lab researchers to set up multiple cameras and synchronize them, first-person cameras are easy to use without any special configurations.

Based on the BioVL2 dataset, we develop a system of generating protocols from experiment videos. Because the dataset size is limited, we opt to utilize an existing procedural text generator that is specifically designed to be applicable in such scenarios. Our experimental results and behavior analysis of the model highlight the insights and limitations of this approach. Additionally, we provide guidelines based on our experiences in dataset construction for the future development of the BioVL project.

## **1.4 Thesis outline**

This thesis consists of six chapters as follows.

### **1.4.1 Introduction (Chapter 1)**

In this chapter, we describe the background and motivations to tackle the task of generating procedural text from instructional videos. Then, we enumerate three key challenges and explain the solutions to them.

### **1.4.2 Related work (Chapter 2)**

In Chapter 2, we review the related work to the thesis from two perspectives: (1) multimedia and (2) CV and NLP research. From the viewpoint of multimedia research, we emphasize the novelty and usefulness of our work by describing the detailed evolution steps in the past 30 years. From the viewpoint of CV and NLP, we especially focus on “vision and language,” an interdisciplinary research field between them. After reviewing the general vision-and-language research by enumerating tasks, common approaches and benchmark datasets, we focus on describing the existing work that specifically targets procedural text and instructional videos.

### **1.4.3 State-aware procedural text generation from segmented key events (Chapter 3)**

In Chapter 3, we propose a state-aware procedural text generation model from segmented key events and materials. This model focuses on representing their state transition in the latent feature space and generating procedural text based on the simulated results. Our experimental results show that the proposed method achieves better performance than the video captioning

models. In addition, the qualitative analysis of material vectors reveals that they acquire the state transition of materials effectively.

#### **1.4.4 Multimodal recurrent learning of the event selector and sentence generator (Chapter 4)**

In Chapter 4, we propose a multimodal recurrent learning approach for the event selector and sentence generator. The event selector chooses the events from the event candidates of DVC outputs and the sentence generator generates procedural text based on them. Their prediction results are shared between both modules by memory vectors, which hold the history of previously predicted events and sentences. Our experimental results show that our method achieves better performance than the DVC models. In addition, we reveal that it captures the overall story of instructional videos; that is, it can select the appropriate number of events in the correct order. Our detailed ablation studies show the essential components to achieve optimal performance.

#### **1.4.5 BioVL2: Egocentric biochemical video-and-language dataset (Chapter 5)**

In Chapter 5, we build the BioVL2 dataset, which consists of wet-lab experiment videos, protocols and the two types of video-and-language annotations: (1) event-sentence alignment and (2) bounding boxes for objects in the protocols. This is the first attempt to release the video-and-language dataset in the biochemical domain. We analyze the characteristics of the video and language sides and ensure their diversity. Then, we apply the existing procedural text generation model to the dataset. Our experimental results show that it achieves better performance than the weak baseline. Our further investigation reveals the insights and limitations of the method. Finally, we discuss the model's behavior and limitations for details and provide guidelines from experiences of the dataset construction for future development of the BioVL project.

#### **1.4.6 Conclusion (Chapter 6)**

In Chapter 6, we state the summary, limitation and future work.

# Chapter 2

## Related Work

In this chapter, we first review related multimedia research and clarify the novelty and usefulness of our work by describing the detailed evolution steps in the past 30 years. Then, we review the vision-and-language research, interdisciplinary research between CV and NLP. After introducing general research topics on vision and language, we describe existing work that specifically targets procedural text and instructional videos.

### 2.1 Multimedia research that targets procedural text and instructional videos

Here, we overview the previous multimedia research that focuses on procedural text and instructional videos. We define multimedia as a collection of technologies that deal with multiple media (e.g., images, videos, audio and text) to assist human activities. In this community, the long-standing question is how to develop an integrated intelligence that processes video and language from different modalities to assist people in their work. Here, we describe the detailed evolving research steps in chronological order.

In the 1990s, the researchers at Carnegie Mellon University launched the Infomedia project (Fig. 2.1), where its aim is mentioned as<sup>1</sup>:

The overarching goal of the Informedia initiatives is to achieve a machine understanding of video and film media, including all aspects of search, retrieval, visualization and summarization in both contemporaneous and archival content collections.

Howard Wactlar, *Informedia Digital Video Library*.

---

<sup>1</sup><https://www.ri.cmu.edu/project/informedia-digital-video-library/>

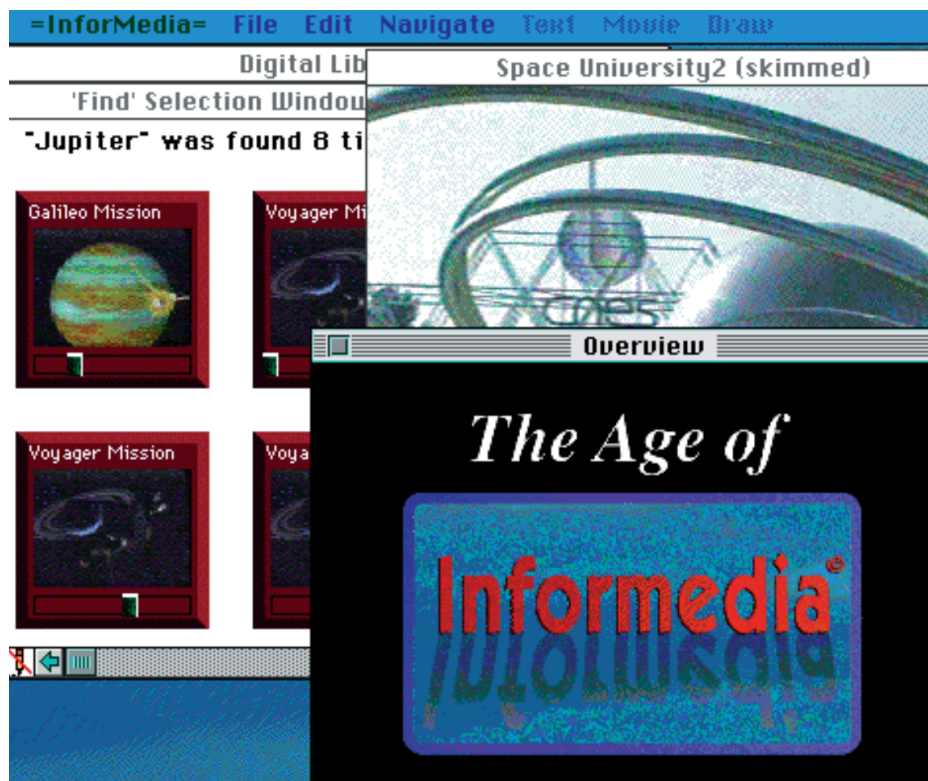


Fig. 2.1 A prototype screen of the Infomedia project quoted from Figure 2 in [22].

The technology developed under the Infomedia project is to associate speech, images, videos with natural language to segment and index for high-level video processing (e.g., text-based moment retrieval). For example, Smith *et al.* [111] proposed a method to integrate the video and audio information and create a skimmed video that represents a very short video abstraction from the original videos. Based on this, Christel *et al.* [23] studied how to create video summarization to satisfy user's demands for video watching. In this way, the Infomedia project presented many novel technologies and could be located as a pioneer study in multimedia archiving.

In the early 2000s, a group led by Hamada and Ide *et al.* started to focus on instructional videos as a research target towards novel multimedia applications, such as the virtual assistant to help us achieve procedural tasks. Their goal is to construct multimedia archives and develop an assistant system for users to cook well. To achieve this, [41] aims at associating video segments and text recipes. Their proposed method first extracts keywords and key events from the audio and video, respectively, then associates text recipes with video events using keywords and corresponding timestamps as a pivot. In line with this work, they also tackled the task of video summarization. Miura *et al.* [78] focused on the motion repetition of

cooking actions and proposed a method to extract important frames that can represent the best overview of the video.

Based on the findings knowledge on the above studies, they developed a cooking navigation system [43], which assists people to cook by providing multimodal information about the next step. Note that this system requires the chief to press the foot switch to obtain the information. Hashimoto *et al.* [45] extended this idea and proposed a user-centric assistant system that provides useful information to cook without any user actions. Another line of this idea is to introduce language-based communication between humans and systems. Laroche *et al.* [63] proposed a dialogue-based cooking coach, which guides people to cook by answering user's questions using voice. Such voice-based assistant systems drew attention from the market. In 2006, Nintendo released a navigation game [150], and it became a huge hit, selling one million copies.

In the mid-2000s, following the success of statistical ML, ML-based approaches were introduced. For example, Shibata *et al.* [109] proposed a structural learning method that converts procedural text into graphical representations and links them into video clips. To connect vision and language information efficiently, they utilized closed captions, and their corresponding timestamps are fully leveraged to connect different two modalities. Based on it, they developed a video summarization and retrieval system, which is helpful for users to grasp the video contents without watching them [108].

An Inria group [2, 3, 10] also focused on the narration in the video and proposed a method to learn video-text alignment in an unsupervised manner. The goal of this study is to discover  $K$  important steps from the video and narration pairs, and their approach is to construct discriminative clustering by preserving the textual constraint. Naim *et al.* [81, 82] also proposed an unsupervised approach to align video segments and key steps in the procedural text. They incorporate HMMs into the IBM model [14], which was originally proposed in the machine translation field to compute word alignment in different languages.

Despite the significant progress in unsupervised approaches, the integration of video and language processing was not yet fully established due to a lack of methods for embedding the feature space that captures common high-level semantics from low-level features of vision and language. Therefore, their approaches utilized transcription as a cue to align video and text, leveraging two merits: (1) timestamps that correspond to the videos and (2) ease of computing similarity to procedural text.

From the 2010s to the present, the rapid advancement of deep learning has enabled computers to connect different modalities using neural networks. Among various research topics, video captioning has paid attention to multimedia researchers because of a high expectation for real-world applications. One of the innovative aspects of video captioning

is its ability to directly convert video content into textual symbols, without relying on transcription, accelerating the construction of multimedia archives. This is a clear advancement from the previous video-text alignment that requires input as triplets of videos, transcription, and procedural text. Moreover, the generated procedural text is also useful to assist human understanding of the videos by providing multimedia summarization through pairs of videos and sentences.

Despite such broad benefits, few researchers tackled this problem in the literature. Ushiku *et al.* [119] were unique researchers to tackle this topic and proposed a procedural text generation method from instructional videos. Their approach is designed for a task setting that is applicable to the small dataset. This is reasonable as a first attempt because they can create a prototype, avoiding the annotation costs between video-and-text pairs. In general, it is difficult to train end-to-end video captioning models on a small dataset, thus they combine two off-the-shelf modules of an object detector and language model. The former detects objects from the videos and the latter generates candidate sentences based on them. The procedural text is outputted by exploring the most plausible combinations using the Viterbi algorithm [125]. In Chapter 5, we applied this method to the BioVL2 dataset because it contains only 32 videos, which is not enough to train end-to-end models.

Their approach has a clear limitation on generating correct actions because it inputs only the object names and discards other visual information. In this thesis, we realize an end-to-end procedural text generation model in Chapters 3 and 4. Although our approaches require annotated pairs between videos and sentences, they have a powerful ability to verbalize both objects and actions based on the input videos.

## 2.2 Vision and language

CV and NLP are essential research topics to achieve human-like intelligence in computers. While CV aims at acquiring eyes for computers to see the world like us, NLP aims at understanding language to communicate with us. For a long time, researchers recognized the importance of dealing with multiple modalities as well as developing single-modal intelligence. For example, template-based approaches have been investigated for visual captioning [36, 61, 88]. However, the lack of flexibility causes a large gap between manual and machine descriptions.

The breakthrough of neural networks, deep learning [64], is a game changer in this field. It enables computers to connect vision and language sequentially to transform images into texts and vice versa. The *show and tell* paper [124] firstly proposed a neural-network-based

image captioning model, which consists of convolutional neural networks (CNNs) [58] and recurrent neural networks (RNNs) [114] to encode and decode images and texts, respectively.

Vision and language are hot topics in CV and NLP now. The goal is to accomplish multi-modal intelligence, which can convert multi-modal information mutually. The researchers believe that this is an essential component of embodied agents that complete tasks in the real world [13, 35]. In this chapter, we show representative tasks with commonly-used approaches and datasets in vision and language.

### 2.2.1 Tasks, methods and datasets

Various kinds of tasks have been proposed with a rapid advance in vision and language. Among them, we mainly focus on four representative tasks: (1) vision-and-language alignment, (2) vision-and-language retrieval, (3) visual captioning and (4) dense video captioning. Fig. 2.2 shows an outline of these four tasks.

#### Vision-and-language alignment

Vision-and-language alignment (1 in Fig. 2.2) is a task to estimate the correspondence between visual information (e.g., objects, relationships and video events) with textual expressions (e.g., phrases and sentences). This task derives into two sub-topics, depending on the types of input visual/textual resources. In the case where the inputs are images and sentences, this task is called image grounding. When the input pairs are videos and paragraphs, it is called video-text alignment. Here, we describe both of these subtopics.

**Image grounding** is a task to estimate the alignment between objects in images and phrases (namely spans, word sequences) in sentences. The common approach is to cascade two-stage predictions: (1) detecting objects using the object detection models (e.g., Faster RCNN [98]) and (2) computing the correspondence of objects and phrases using image and text encoders. ResNet [47] and BERT [32] are frequently used for powerful encoders of images and texts, respectively. The common datasets are RefCOCO, RefCOCO+ [141], RefCOCOG [73] and Flickr30k entities [93], which provide annotations between objects and phrases in the sentences for MSCOCO [68] and Flickr30k [140] datasets.

**Video-text alignment** is a task to localize events (start and end timestamps) given sequential sentences. Note that it assumes that one sentence corresponds to one event in the video. This task was well-investigated [10, 2, 72] before the deep-learning era by employing the hidden Markov model (HMMs) and conditional random fields (CRFs) [62]. However, it is necessary to pre-compute the correspondence scores with hand-crafted features (e.g., how many objects are matched) for learning HMMs and CRFs. The current common

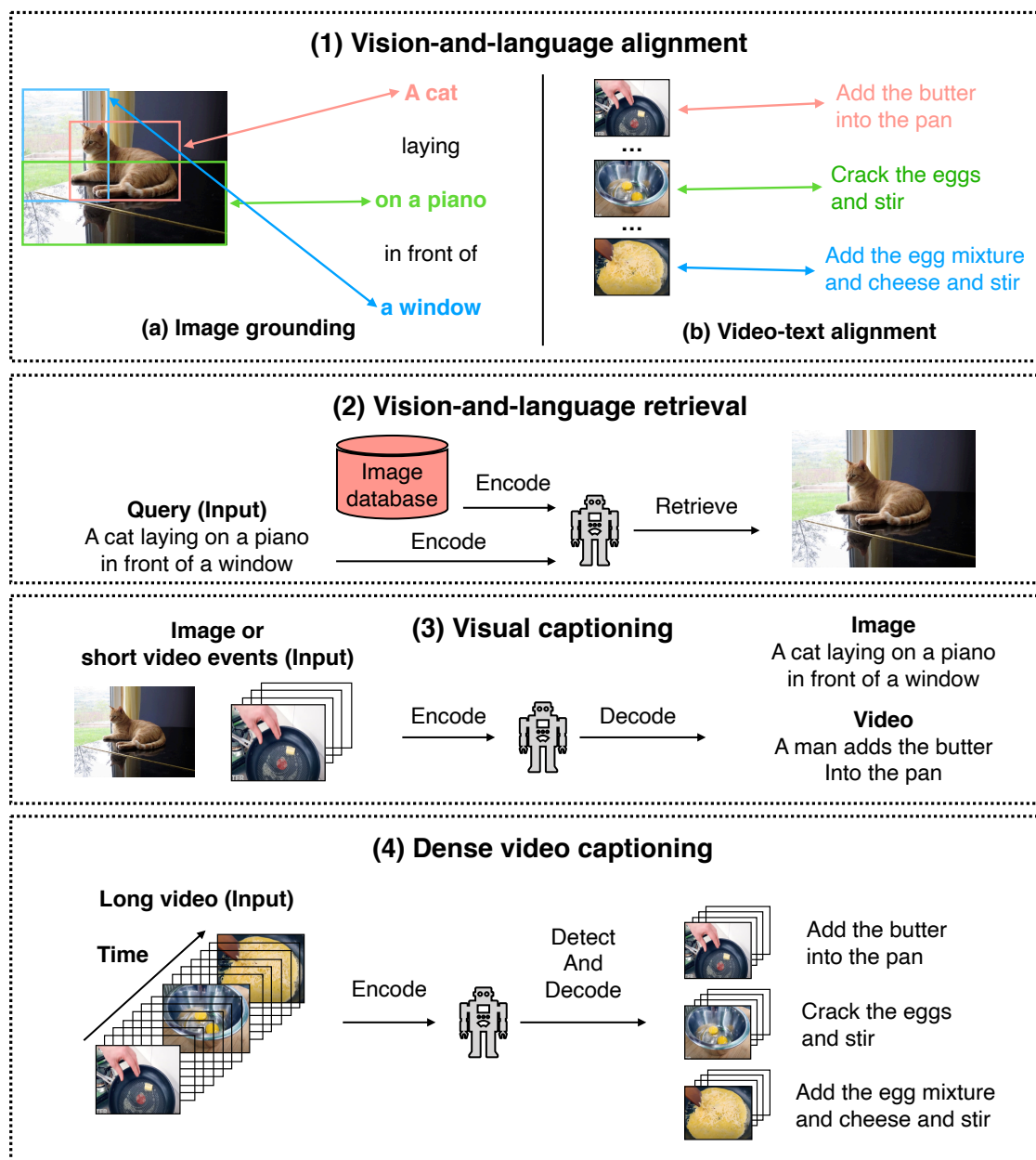


Fig. 2.2 An overview of the vision-and-language tasks. Example images and videos are used from the MSCOCO [68] and YouCook2 [147] datasets.



approach is to employ neural networks to directly compute correspondence between events and sentences with differentiable alignment [17, 105], enabling to train the model in an end-to-end manner. The commonly-used benchmark datasets are TACoS [97], which provides annotations between events and sentences.

### **Vision-and-language retrieval**

Given visual or textual queries, vision-and-language retrieval aims at retrieving other modality items that are semantically similar to the query (2 in Fig. 2.2). The common approach is to encode modality-independent representations and project them into common feature space to acquire their joint representations. To achieve this, many kinds of neural architectures and loss functions have been explored [101, 70, 113]. Recent research demonstrated that contrastive learning [20] is quite effective to construct the joint representations of vision and language. CLIP [94] is the most popular one, which learns 400 million image-text pairs and achieves the best performance on the many vision-and-language downstream tasks. Several researchers tried to extend CLIP to video-and-text versions, such as VideoCLIP [135] and Frozen [6]. The commonly-used benchmark datasets are MSCOCO [68], Flickr30k [140], ActivityNet Captions [57], YouCook2 [147] and MSR-VTT [136], which consists of visual resources with manually-created textual descriptions.

### **Visual captioning**

Visual captioning is a task to describe the content of an input image or video in natural language (3 in Fig. 2.2). This task requires computers to recognize not only fine-grained objects but also their relationships (e.g., positions and context) to generate accurate and concrete descriptions. The common approach of visual captioning is to employ seq2seq [114], an encoder-decoder architecture of neural networks, originally proposed for machine translation. The encoder converts images/videos into visual representations and the decoder translates the encoded representations into textual descriptions. As mentioned in Chapter 2.2, Vinyals *et al.* [124] was a pioneer to apply the encoder-decoder architecture to the image captioning task. They demonstrated the effectiveness of the neural architecture for the image captioning task by outperforming the previous template-based image captioning models. They employed the CNN encoder and RNN decoder as ResNet [47] and LSTM [48], respectively. Several researchers reported that similar architectures are effective for video captioning [123, 34]. The commonly-used datasets are the same as the vision-and-language retrieval.

### Dense video captioning

DVC is a sub-topic of video captioning, aiming at detecting events thoroughly and generating sentences for them (4 in Fig. 2.2). This task is similar to our task because the input/output pairs are in the same format: an input is videos and the outputs are events and sentences. What is the difference between DVC and our task? To clarify this, we first describe traditional DVC approaches and datasets, then explain the key difference.

DVC aims at thoroughly detecting events from the video and generating sentences for them. The naive baseline pipelines the event extractor and the sentence generator [57], but it does not perform well because the sentence generation loss is not backpropagated to the event extraction module. To address this issue, several researchers tried to train them in an end-to-end manner. Zhou *et al.* [145] proposed a Transformer [121]-based DVC model, Masked Transformer, which detects events and generates sentences via a differentiable mask. Their approach, however, outputs more than 200 redundant events per video on average [37]. To handle this issue, Wang *et al.* [129] proposed parallel DVC (PDVC), which detects events in parallel, re-ranks the top  $K$  ( $K$  is also the prediction target) and generates sentences for the re-ranked events. The commonly-used datasets are ActivityNet Captions [57] and YouCook2 [147], which have the annotations of event timestamps and corresponding sentences.

The key difference between our task and DVC lies in the story awareness of instructional videos. Although DVC allows computers to exclusively detect false-positive events, our task requires computers to extract an appropriate number of events in the correct order. The redundancy of the outputs of DVC makes it difficult for users to grasp an overview of the video contents. In Chapter 4, our task resolves this problem by introducing the story-awareness into the model.

## 2.3 Vision and language in instructional domains

In this section, we describe the vision and language research that specifically targets procedural text and instructional videos. Here, we emphasize the novelty of the work by comparing it with existing works from three perspectives: (1) multi-modal understanding of the procedural text with vision, (2) procedural text generation from visual observations and (3) multi-modal instructional video datasets.

### 2.3.1 Multi-modal understanding for procedural text with vision

This thesis discusses how to model the material state changes in Chapter 3. The idea for the proposed methods is inspired by this research topic, so we review the previous work of multi-modal understanding for procedural text with vision.

The origin of this topic is NLU and multi-modal versions are extended from previous NLU work. Thus we first describe NLU research for procedural text, then review the multi-modal extensions.

**Procedural text understanding.** In NLU, procedural text is a popular target for understanding, and recipes have been featured in this field for a long time. Many researchers have proposed methods to understand recipes, which are roughly divided into two categories: (1) a graph-based parser and (2) a reasoning-based simulator.

The initial idea of a graph-based parser dates back to the 1980s. Momouchi [79] proposed a PT (Procedural Text)-chart, which represents an entire workflow of actions and materials as the Backus-Naur form. This work is pioneer research to convert procedural text into graphical representations. After several rule-based or supervised methods were proposed [42, 71], Kiddon *et al.* [54] firstly proposed an unsupervised method to estimate a graph structure, dubbed action graph from a recipe. An action graph is also used in the challenging visual reference resolution task described in [49, 50]. There are several varieties of graphical representations. Mori *et al.* [80] provided a recipe flow graph dataset, which contains fine-grained expressions as nodes and expresses their relationships as edges. Although the action graph targets actions, ingredients and tools, the recipe flow graph additionally covers other textual expressions, such as times, quantity and the state of foods. Jermurawong and Nizar [53] have proposed the Simplified Ingredient Merging Map in Recipes (SIMMR) dataset, which provides a merging tree, a tree structure tracking the ingredient merging operations.

A reasoning-based simulator was recently proposed to model the state transition of ingredients in recipes by updating the ingredient states at each step. Compared with the graph-based parser, the reasoning-based simulator enables the computers to softly predict the state transition of ingredients. Gupta and Durrett [40] proposed a structured simulator that imposes constraints on the state transition of ingredients (e.g., stir-fried potatoes are not cut in later steps). Bosselut *et al.* [12] proposed the neural process network (NPN), which learns the state transition of ingredients by predicting executed actions and involved ingredients from the procedural text. Owing to the ability to capture the state transition of ingredients, these simulators reported competitive results in NLP downstream tasks, such as question answering (QA) and natural language generation (NLG).

**Multi-modal understanding for procedural text.** When reading and executing procedural text, we identify objects through the material state changes (e.g., cut potatoes) and

interpret the procedures by connecting their visual information to the descriptions. To imitate this, several studies proposed NLU-inspired multi-modal extensions.

The multi-modal version of a graph-based parser was recently investigated research topic. The main issue of the multi-modal graph-based parser is the limited available datasets. To address this issue, Pan *et al.* [89] recently created an MM-ReS dataset, which consists of 9,850 recipes, image sequences and annotated tree structures, allowing researchers to analyze the cause-and-effect relations between step sentences and images in the recipe and image sequence. In line with this work, Nishimura *et al.* [85] proposed visual SIMMR (vSIMMR), which can be seen as a simple extension of the MM-ReS dataset by connecting images not only with step texts but also with ingredients on the Cookpad Image dataset [44]. The connection between images and ingredients allows us to further investigate the state changes from raw to manipulated processes. They verify the effectiveness of vSIMMR for the task of generating procedural text from image sequences [83, 16].

The multi-modal version of a reasoning-based simulator was also recently investigated. Amac *et al.* [4] proposed a method to answer multi-modal QA tasks by reasoning the state transition of ingredients using a relation reasoning network [102]. They reported that a reasoning-based simulator is quite effective for multi-modal QA.

**Our extension from previous work.** This thesis focuses on the reasoning-based simulator to learn the material state changes. Compared with the graph-based approach [85, 89], it has two clear merits: (1) the model is trainable without any manual annotations and (2) it is less likely to propagate the failure of this module to the downstream process because of its soft prediction. In Chapter 3, we extend the NPN to the multi-modal version, called a visual simulator, which can represent the state transition of ingredients as the trajectory of ingredient vectors in the latent feature space. We incorporate this into the transformer-based encoder-decoder architecture and verify the effectiveness of generating procedural text from key events.

### 2.3.2 Procedural text generation from visual observations

Procedural text is a popular target in NLG to verify the computer’s ability to generate coherent sentences [55, 11]. Recipes are commonly used as the testbed for this task, and the inputs of previous work are the title and ingredients of recipes. Recently, many multi-modal cooking datasets have appeared [101, 44] and recipe generation from images has also been proposed [100, 127]. Salvador *et al.* [100] tackled this problem using a transformer-based model to generate high-quality recipes from a single image of a completed dish. Other researchers focused on generating a recipe from an image sequence depicting the intermediate food states [16, 83, 85].

Table 2.1 Comparative overview of multi-modal instructional video datasets.

	Domain	Procedural text?	Materials?	Visual data	#Videos
CMU MMAC [29]	Cooking			Third-person	185
TUM Kitchen [117]	Cooking			Third-person	20
50 salads [112]	Cooking			Third-person	50
KUSK [46]	Cooking	✓		Third-person	14
Breakfast [59]	Cooking			Third-person	433
TACoS [97]	Cooking	✓		Third-person	212
YouCook [28]	Cooking	✓		Third-person	88
YouCook2 [147]	Cooking	✓		Third-person	2,000
YouMakeup [131]	Makeup	✓		Third-person	2,800
Howto100M [75]	General			Third-person	1.22M
COIN [116]	General	✓		Third-person	11,827
EPIC-Kitchens [27]	Cooking			Egocentric	700
Assembly101 [104]	Assembly			Egocentric	4,321
EgoProceL [9]	General			Egocentric	329
<b>BioVL2 (ours)</b>	Biochemical	✓	✓	Egocentric	32

The essential limitation of generating recipes from images or image sequences is the lack of detailed, continuous scene information of human manipulations, which images do not contain. Hence, this task is essentially an ill-posed problem. It depends on the obtained language model to generate correct actions. Meanwhile, videos contain this information and have been attracting the attention of many researchers in recent years [84, 133, 106, 107]. The task of this thesis requires computers to (1) extract key events from videos and (2) generate accurate sentences for events. Most researchers have focused on (2) and formulated a task of generating accurate recipes from pre-segmented key events [84, 107, 133]. However, only a few researchers attempt to learn both (1) and (2) to generate recipes from unsegmented videos. Shi *et al.* [106] are pioneer researchers that have tackled this problem by utilizing the narration of videos effectively. Although this approach is effective for narrated videos, transcription is not always available for all cooking videos. Speaking during cooking or adding narration to videos requires significant effort. In addition, narrations may lead the model to attend to textual features, ignoring the visual features of videos. In Chapter 4, we first tried to generate a procedural text only from the videos. This enables the model to treat even non-narrated videos and be more practical than the previous settings.

### 2.3.3 Multi-modal instructional video datasets

Multi-modal instructional video datasets, which consist of procedural text with visual observations, are rising in popularity for learning procedural tasks [149, 75, 2]. Table 2.1 shows a

comparison of the constructed BioVL2 with other video datasets. The datasets are roughly divided into two categories: the third-person and egocentric datasets.

### Third-person datasets

In the last decades, third-person instructional videos are widely used for training and evaluating the model on CV downstream tasks [147, 2, 116, 75] because of its easiness to collect videos from the Internet. Based on it, video and language annotations have been provided by researchers. The domains of such video datasets can be divided into two categories: cooking and other domains.

**Cooking domain.** For a decade and more, the cooking domain was targeted by many researchers because of its large variety of actions and objects [33, 41, 43, 42, 46, 29, 117, 112, 28]. From the 2000s to early 2010s, the main topic among them was fine-grained action and object recognition [29, 117, 112, 59, 28]. The researchers proposed generative and discriminative approaches to recognize actions and objects using HMMs, CRFs and support vector machines (SVMs). These datasets were proposed before the deep-learning era and the size of the dataset was not emphasized. As shown in Table 2.1, YouCook, TaCoS and Breakfast have 88, 212 and 433 videos, respectively, and more than a thousand scale datasets did not appear in this era.

In the late 2010s, researchers recognized that neural networks are data-hungry thus constructing large-scale datasets is inevitable to improve the model’s generalization ability. In addition, the rapid growth of video-sharing platforms enables researchers to collect videos from the web. Zhou *et al.* [147] are a pioneer and constructed the YouCook2 dataset, which consists of 2,000 YouTube videos, recipes and annotations of alignment between events in the video and sentences in the recipe. The dataset size is much larger than the previous work. In Chapters 3 and 4, we use the YouCook2 dataset for the experiments.

**Other domains.** As with YouCook2, several domain-specific datasets have been proposed in anticipation of real-world applications. YouMakeup [131], which consists of 2,800 makeup videos with the same annotation format as YouCook2, was released for multi-modal retrieval and video captioning of makeup videos. More recently, several researchers started to try to collect instructional videos in the general domain not limited to specific domains. Howto100M [75] is the largest dataset, which consists of about 1.22M videos and narrations automatically collected from YouTube. Based on the constructed dataset, they proposed multiple instance learning and noise contrastive estimation (MIL-NCE) models [74] pre-trained on it and reported that MIL-NCE outperforms the existing models on the CV downstream tasks, such as action recognition, temporal action localization and video-text retrieval.

Because the narrations in Howto100M are collected automatically by using automatic speech recognition, the video-text annotations are weakly-paired and noisy. The manual annotations are necessary at least for evaluating the model’s performance. COIN [116] is another largest dataset, which consists of 11,827 videos with manual annotations of video events and textual descriptions. It is now used for tasks of learning from instructional videos, such as video procedure captioning [133] and procedure planning [18, 144].

### Egocentric datasets

The egocentric dataset [27, 59, 104], which includes videos captured from a first-person view using wearable cameras, has attracted significant interest from researchers in recent years. The egocentric view can capture detailed human-object interactions during task completion, providing valuable insight for applications such as augmented reality and robotics.

Scalability is the main problem to construct egocentric datasets because of the cost to collect large-scale videos. This leads to a limited dataset size of fewer than 100 videos. Damen *et al.* [27] first reached this problem and constructed the EPIC-KITCHEN dataset, which consists of 700 videos that record all of the cooking manipulation processes completed by 32 participants. Each video contains linguistic annotations of fine-grained human actions (e.g., opening the refrigerator and cutting the potatoes). Bansal *et al.* [9] followed this research and constructed the EgoProceL dataset, which consists of 329 egocentric videos with key step annotations in the general domain. Now, the largest egocentric instructional dataset is Assembly101 [104], which has 4,321 videos of assembling toy cars. It also has a rich variety of annotations, including coarse and fine actions, key events, 3D hand poses, skill levels and mistake actions of the workers. This dataset was used for the shared task benchmark at the ECCV workshop in 2022 <sup>2</sup>.

Covering various domains is still problematic for dataset construction. In this thesis, we focus on the biochemical domain because of its potential applications to enhance the reproducibility of experiments. Although a few researchers tackle video-and-language research in wet-lab domains [81, 82], there are no publicly available datasets on the web. To address this issue, in Chapter 5, we build the BioVL2 dataset, which is the first attempt to release the egocentric biochemical video-and-language dataset on the web only for the research purpose <sup>3</sup>. It consists of eight videos for four kinds of experiments, summing 32 videos with 2.5 hours. As shown in Table 2.1, the dataset size is much smaller than that of the EPIC-KITCHEN, Assembly101 and EgoProceL. However, we believe that it will be the first

---

<sup>2</sup><https://sites.google.com/view/egocentric-hand-body-activity/home>

<sup>3</sup><https://github.com/awkrail/BioVL2>

step for the future development of new video-and-language technologies in the biochemical domain.



# Chapter 3

## State-aware Procedural Text Generation from Segmented Key Events

### 3.1 Introduction

In this chapter, we focus on generating a procedural text from key events and materials pre-segmented in instructional videos (Fig. 3.1). As mentioned in Chapter 1, the essential difficulty is to track material state changes to describe detailed material manipulations from visual observations. For example, when generating step 3 in Fig. 3.1, the models should be aware that the yellow liquid in the bowl and white liquid in the pan correspond to the eggs and butter manipulated in steps 1 and 2, respectively. Existing video captioning models [34, 115, 65, 134, 145] cannot track material states, leading to the failure to generate accurate procedural text.

To address this challenge, this chapter aims to generate procedural texts by modeling the state transition of materials from visual observations. Previously, NLU researchers share the same motivation of tracking material state changes and proposed the reasoning-based simulator [12, 4]. These simulators represent materials in the latent feature space and recurrently update them by predicting executed actions and involved materials at each step in procedural text.

Inspired by these ideas, we propose a novel method, which modifies an existing NLU simulator as a *visual simulator* and incorporates it into an encoder-decoder architecture. We call this approach a state-aware method. Fig. 3.1 illustrates the idea of the proposed method. Given key events and material sets, the proposed method reasons the state transition of materials and generates procedural text accurately. We emphasize that this work is the first attempt to integrate the NLU simulator into a video captioning model. In addition, based on

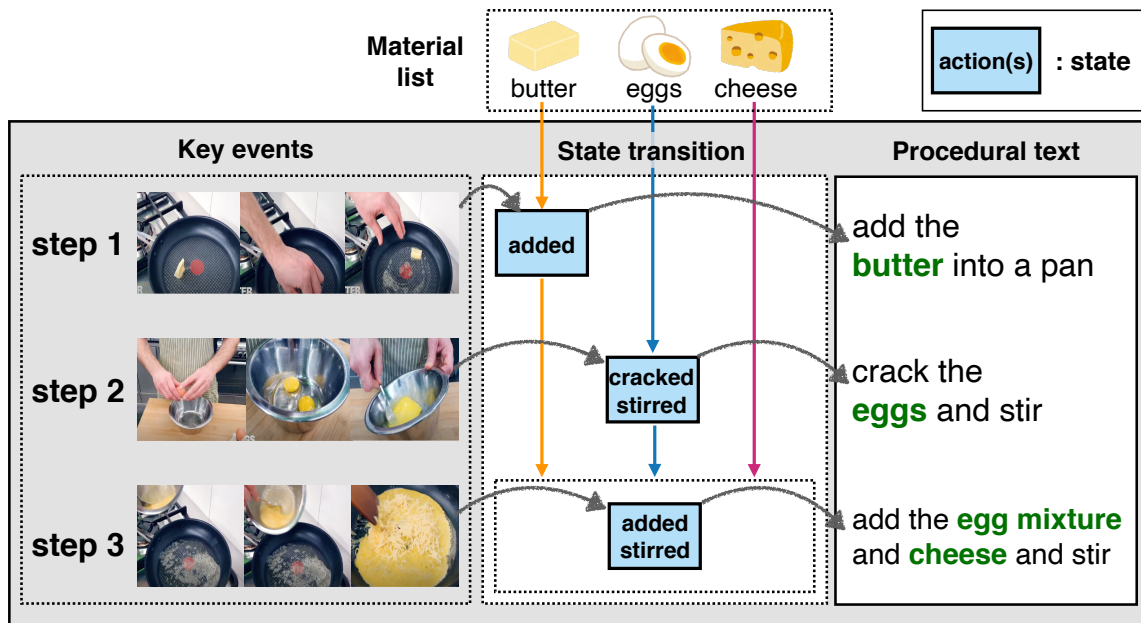


Fig. 3.1 Concept of the proposed method. Given key events and material list, the proposed method generates a procedural text by reasoning the state transition of materials at each step.

the intuition that the state transition of materials is consistently traceable from the generated procedural text, we attach a novel *textual re-simulator*, facilitating the model to generate procedural text even more accurately.

In our experiments, we test the proposed method in the cooking domain and compare it with two state-of-the-art video captioning models on four evaluations: word-overlap evaluation, ingredient prediction, retrieval evaluation and qualitative analysis. Our experimental results show that the proposed method outperforms the state-of-the-art video captioning models. In addition, ablation studies show the effectiveness of integrating visual and textual simulators into the model. Our analysis of the learned embedding demonstrates that the simulators effectively capture the state transition of materials. Finally, we conduct the full prediction settings, where the materials are not given, but are predicted from the video events in advance. Then we discuss whether the proposed method performs well with the input of the predicted ingredients or not.

## 3.2 Proposed method

In this section, we present the proposed method. After describing an overview in Chapter 3.2.1, we explain the components of the proposed method from Chapter 3.2.2 to Chapter 3.2.5. Finally, in Chapter 3.2.6 we explain the loss functions to train the model.

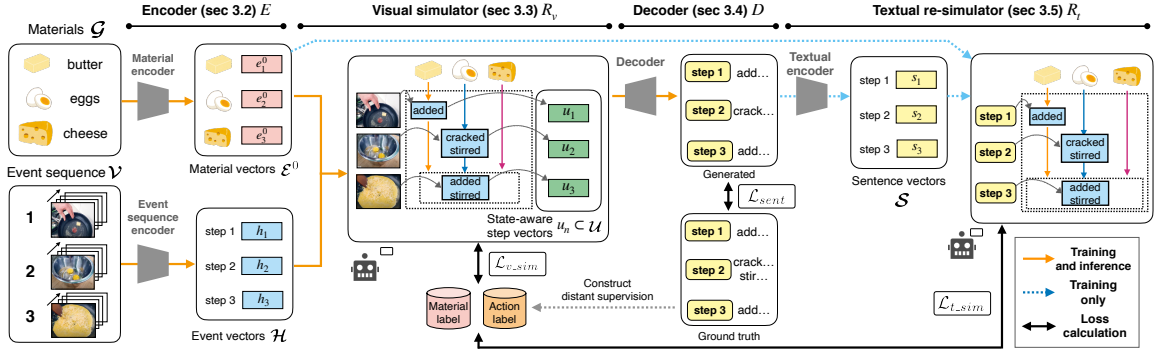


Fig. 3.2 An overview of the proposed method. To track material states in key events, we incorporate the visual simulator  $R_v$  into the transformer-based encoder-decoder architecture ( $E$  and  $D$ ). In addition, based on our intuition that the state transition of materials is traceable from the generated procedural texts, we attach the textual re-simulator  $R_t$  to the model.

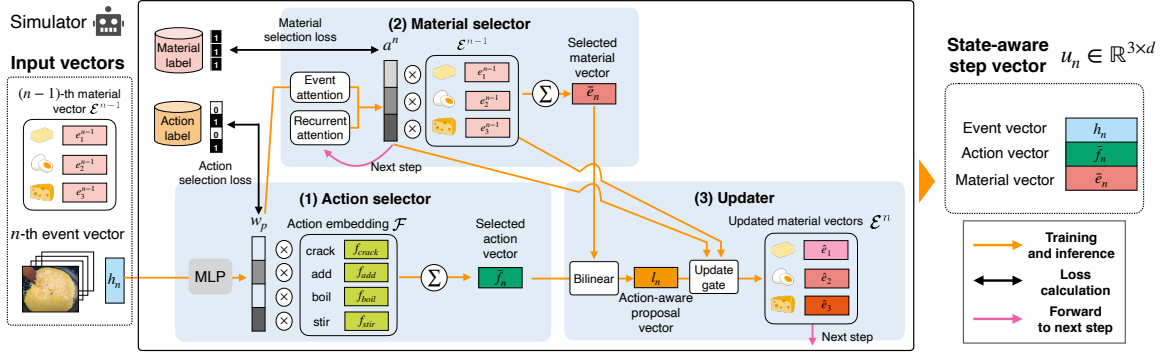


Fig. 3.3 An overview of the (visual) simulator. The simulator recurrently reasons the state transition of the materials at each step. Specifically, it predicts executed actions and involved materials in (1) the action and (2) material selector and then updates the state of materials in (3) the updater. The updated materials are forwarded to the next step. The textual re-simulator has the same modules.

### 3.2.1 Overview

Fig. 3.2 shows an overview of the proposed method. Given key events  $\mathcal{V} = (v_1, \dots, v_n, \dots, v_N)$  and a material list  $\mathcal{G} = (g_1, \dots, g_m, \dots, g_M)$ , our goal is to output procedural text  $\mathcal{Y} = (y_1, \dots, y_n, \dots, y_N)$  by recurrently generating sentences from corresponding events. To generate accurate procedural text, it is essential for models to track material states in the key events (e.g., eggs are transformed into cracked, stirred, then fried forms).

Inspired by recent advances in NLU, we achieve this by modifying the existing reasoning-based NLU simulator NPN as the visual simulator  $R_v$ , and incorporate it into the transformer-based encoder-decoder architecture ( $E$  and  $D$ ). Specifically, given event  $\mathcal{H}$  and material encoded vectors  $\mathcal{E}^0$ , the visual simulator reasons the state transition of materials and outputs

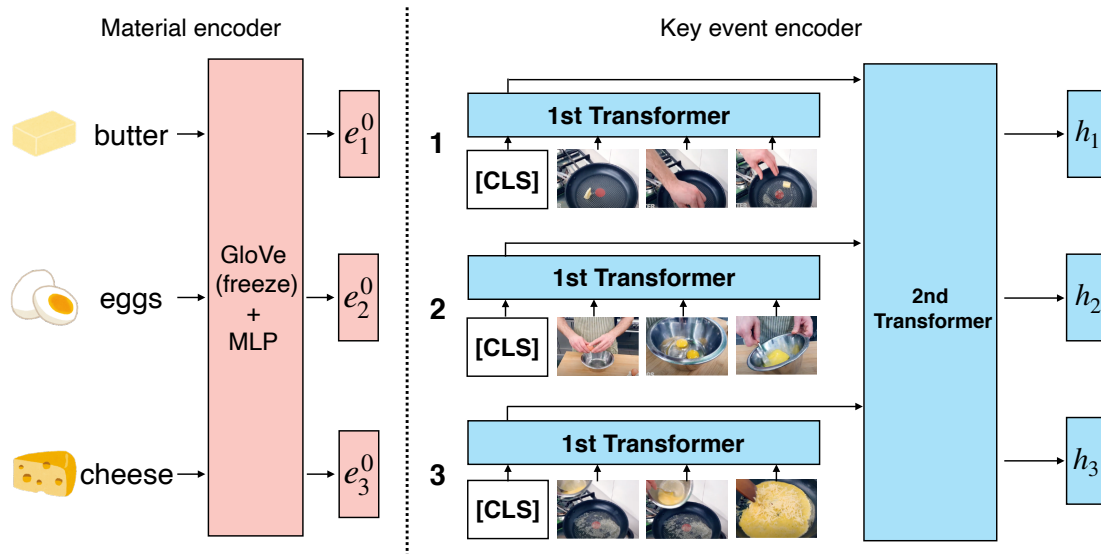


Fig. 3.4 An overview of the material and key event encoders.

state-aware step vectors as  $\mathcal{U} = R_v(\mathcal{H}, \mathcal{E}^0)$ . Then, the decoder  $D$  outputs procedural text conditioned on  $\mathcal{U}$ . We observe that this simple integration of the visual simulator is effective for the model to generate accurate procedural text.

In addition, based on our assumption that the state transition of materials should be consistently traceable from the generated procedural text, we attach a novel textual re-simulator  $R_t$ , encouraging the model to generate procedural text even more accurately. Specifically, the textual re-simulator  $R_t$  reasons the state transition of materials from the generated procedural text as  $R_t(\mathcal{S}, \mathcal{E}^0)$ , where  $\mathcal{S}$  represents the encoded sentence vectors. Note that the textual re-simulator is only used during the training phase and detached during the inference phase.

### 3.2.2 Encoder

The input has two components: a material list  $\mathcal{G}$  and key events  $\mathcal{V}$ . Thus, we develop a suitable encoder for each component. Fig. 3.4 shows an overview of the encoders.

#### Material encoder

To encode a material list, we input them to concatenated neural networks of pre-trained GloVe [92]<sup>1</sup> word embedding and multi-layer perceptrons (MLPs) with the ReLU activation

<sup>1</sup>We employ pre-trained 300D word embedding, which can be downloaded from <http://nlp.stanford.edu/data/glove.6B.zip>

function. Multi-word materials (e.g., parmesan cheese) are represented by the average embedding vector of words. As following the above procedure, we obtain the initial material vectors  $\mathcal{E}^0 = (e_1^0, \dots, e_m^0, \dots, e_M^0)$ .

### Key event encoder

The key events  $\mathcal{V}$  are hierarchical because they contain multiple events, and each event is composed of sequential frames. Thus, to encode key events effectively, we design a two-stage transformer suitable to encode a sequence of sequences. First, the former transformer encodes each event into a feature vector by extracting the vector, which corresponds to the [CLS] token as in [32, 113, 65]. Then the latter transformer is trained over the sequence to obtain the step-aware event vectors  $\mathcal{H} = (h_1, \dots, h_n, \dots, h_N)$  in the key events.

### 3.2.3 Visual simulator

Based on the encoded vectors of the key events and material list  $(\mathcal{H}, \mathcal{E}^0)$ , the visual simulator, shown in Fig. 3.3, reasons the state transition of materials at each step. Specifically, at the  $n$ -th step, given the  $n$ -th event  $h_n$  and  $(n-1)$ -th material list  $\mathcal{E}^{n-1}$ , the visual simulator predicts executed actions and involved materials in (1) the action and (2) material selector and then updates the state of materials in (3) the updater. After  $n$ -th reasoning, it outputs a state-aware step vector  $u_n \in \mathbb{R}^{3 \times d}$ , which concatenates the  $n$ -th event  $h_n$ , selected action  $\bar{f}_n$  and material vectors  $\bar{e}_n$  ( $d$  represents the dimension of these vectors). The visual simulator recurrently repeats the above process until processing the end element of the key events. This simulation process is the same as NPN except for the input modality. We replace the textual sentence and entity vectors for NLU with visual event  $\mathcal{H}$  and material vectors  $\mathcal{E}^0$  for our task.

#### Action selector

Given an event vector  $h_n$ , the action selector outputs the selected action vector  $\bar{f}_n$  by choosing actions executed in the event from the predefined action embedding  $\mathcal{F}$ . For example, in Fig. 3.3, the actions “crack” and “stir” are executed in the event, thus both  $f_{crack}$  and  $f_{stir}$  should be selected. To consider multiple actions, the action selector computes a soft selection  $w_p$  as an action probability for each action in  $\mathcal{F}$ . Then it outputs the selected action vector  $\bar{f}_n$  as a weighted sum of the action embedding  $\mathcal{F}$  and action probability  $w_p$ .

Specifically, the calculation of the action selector is written as follows:

$$w_p = \text{MLP}(h_n) \quad (3.1)$$

$$\bar{w}_p = \frac{w_p}{\sum_j w_p^j} \quad (3.2)$$

$$\bar{f}_n = \bar{w}_p^T \mathcal{F}, \quad (3.3)$$

where  $\text{MLP}(\cdot)$  represents two-layer MLPs with the sigmoid function and  $w_p \in \mathbb{R}^{|\mathcal{F}|}$  is the attention distribution over  $|\mathcal{F}|$  possible actions.

### Material selector

Based on the action probability  $w_p$  and event vector  $h_n$ , the material selector outputs a selected material vector  $\bar{e}_n$  by choosing materials that are involved in the event from the material list  $\mathcal{E}^{n-1}$ . For example, in Fig. 3.3, the raw “cheese” and manipulated “eggs” and “butter” should be selected. To consider such a combination of raw and manipulated material selection, the material selector has two attention modules: (1) event attention and (2) recurrent attention.

The event attention chooses relevant materials from the event vector  $h_n$  and action probability  $w_p$ :

$$\hat{h}_n = \text{ReLU}(W_1 h_n + b_1) \quad (3.4)$$

$$d_m = \sigma((e_m^{n-1})^T W_2 [\hat{h}_n; w_p]) \quad (3.5)$$

where  $W_1$  and  $W_2$  are linear and bilinear mapping,  $b_1$  and  $b_2$  are biases, and  $e_m^{n-1}$  and  $d_m$  represent the  $m$ -th material vector and its attention weight.

The recurrent attention selects materials based on information from both current and previous events. Using the result of event attention, it computes a soft selection  $a^n$  as a material probability for each material in the material list:

$$c = \text{softmax}(W_3 \hat{h}_n + b_3) \quad (3.6)$$

$$a_m^n = c_1 d_m + c_2 a_m^{n-1} + c_3 \mathbf{0} \quad (3.7)$$

where  $W_3$  is a linear mapping,  $c \in \mathbb{R}^3$  is the choice distribution,  $a_m^{n-1}$  is the attention weight of the previous event for each material,  $a_m^n$  is the final distribution for each material, and  $\mathbf{0}$  is a vector of zeros (providing the option not to select any materials). Finally, using the calculated attention weights, the selected material vector  $\bar{e}_n$  is computed as the normalized

weighted sum of the selected materials.

$$\alpha_m^n = \frac{a_m^n}{\sum_j a_j^n} \quad (3.8)$$

$$\bar{e}_n = \sum_m \alpha_m^n e_m^{n-1}. \quad (3.9)$$

### Updater

Based on the selected actions and materials, the updater represents the state transition of materials by computing a new material vector  $\hat{e}_m$ . To this end, it first calculates an action-aware proposal vector  $l_n$  of materials with a bilinear transformation of selected action and material vectors  $(\bar{f}_n, \bar{e}_n)$ :

$$l_n = \text{ReLU}(\bar{f}_n W_4 \bar{e}_n + b_4), \quad (3.10)$$

where  $W_4$  is a bilinear mapping.

Then, based on the material probability  $a^n$ , it computes the new material vector  $\hat{e}_m$  by interpolating the action-aware proposal vector  $l_n$  and the current material vector  $e_m^{n-1}$ .

$$\hat{e}_m = a_n^m l_n + (1 - a_n^m) e_m^{n-1}. \quad (3.11)$$

The new  $m$ -th new material vector  $\hat{e}_m$  is assigned to  $e_m^n$ , which is forwarded to the next  $(n+1)$ -th step.

### 3.2.4 Decoder

The decoder  $D$  outputs procedural text by recurrently generating sentences from the  $n$ -th output vector  $u_n$  of the visual simulator. Fig. 3.5 shows an overview of the decoder. As our decoder  $D$ , we use the transformer, which achieves state-of-the-art performance in video captioning tasks [148, 106, 107]. Our task allows the model to refer to a material list, thus to encourage the decoder to generate materials, we incorporate the copy mechanism [103] into the decoder  $D$ . When generating the  $k$ -th word in the  $n$ -th sentence, given the updated materials  $e_m^n \in \mathcal{E}^n$ , the copy mechanism first calculates the attention probability  $\beta_{n,k}^m$  using the bilinear dot product of the vectors of the decoder output  $o_{n,k}$  and each material  $e_m^n \in \mathcal{E}^n$  as:

$$\beta_{n,k}^m = \frac{\exp\{(o_{n,k})^\top W_c e_m^n\}}{\sum_i \exp\{(o_{n,k})^\top W_c e_i^n\}}, \quad (3.12)$$

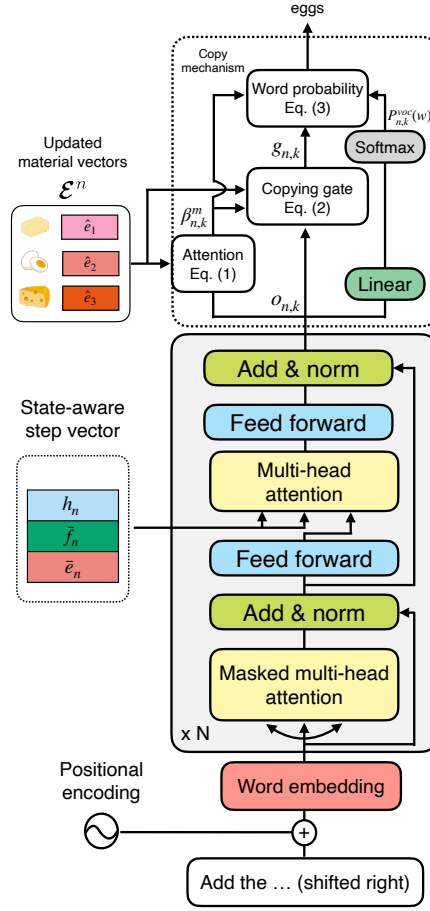


Fig. 3.5 An overview of the decoder.

where  $W_c$  represents a bilinear map.

Then it calculates the copying gate  $g_{n,k}$  ( $0 \leq g_{n,k} \leq 1$ ), which makes a soft choice between selecting a material from the material list and generating a word from the vocabulary:

$$g_{n,k} = \sigma(W_g[o_{n,k}; \sum_m \beta_{n,k}^m e_m^n] + b_g), \quad (3.13)$$

where  $[\cdot]$ ,  $\sigma(\cdot)$ ,  $W_g$  and  $b_g$  represent the concatenation function, sigmoid function, linear map and bias, respectively. Based on  $g_{n,k}$ , the final predicted word probability  $P_{n,k}(w)$  is computed as the weighted sum of the copy probability and generation probability as follows:

$$P_{n,k}(w) = (1 - g_{n,k}) P_{n,k}^{\text{voc}}(w) + g_{n,k} \left( \frac{1}{\|g_m\|} \sum_{i:w_i \in g_m} \beta_{n,k}^i \right), \quad (3.14)$$



where  $P_{n,k}^{\text{voc}}(w)$  and  $\|g_m\|$  represent the probability of the  $n$ -th sentence’s  $k$ -th word  $w$  in the vocabulary and the number of words of in the  $m$ -th material, respectively<sup>2</sup>.

### 3.2.5 Textual re-simulator

Because the state transition of materials is identical in the visual and textual worlds, it should be consistently traceable from the generated procedural text. Based on this assumption, we add a novel textual re-simulator  $R_t$ , encouraging the model to generate procedural text even more accurately.

The textual re-simulator consists of two sub-modules: (1) a textual encoder and (2) a textual simulator. The textual encoder converts generated procedural text into step-aware sentence vectors  $\mathcal{S} = (s_1, \dots, s_n, \dots, s_N)$ . First, it applies the straight-through version of Gumbel softmax resampling [52] to sample procedural text, preserving the differentiable chain. The sampled procedural text is further converted into feature vectors by computing the average vector of the word embedding at each step. Note that the word embedding is shared between the decoder and textual encoder. They are then converted into step-aware sentence vectors  $\mathcal{S}$  using a biLSTM encoder. Finally, based on  $\mathcal{S}$ , the textual simulator, another NPN described in Chapter 3.2.3, reasons the state transition of materials again as  $R_t(\mathcal{S}, \mathcal{E}^0)$ .

### 3.2.6 Loss functions

To train the model, we compute three types of losses: (1) sentence generation loss  $\mathcal{L}_{sent}$ , (2) visual simulation loss  $\mathcal{L}_{v\_sim}$  and (3) textual re-simulation loss  $\mathcal{L}_{t\_sim}$ .

#### Sentence generation loss $\mathcal{L}_{sent}$

This loss aims to train the decoder, and is computed as the summed negative log-likelihood for all input/output pairs  $\{(\mathcal{V}, \mathcal{G}), \mathcal{Y}\}$  in the training set.

#### Visual simulation loss $\mathcal{L}_{v\_sim}$

This loss aims to train the visual simulator and consists of two losses: (1) material selection and (2) action selection loss. These losses are computed as the summed binary cross-entropy loss based on whether the materials/actions are involved/executed in the event. To avoid costly human annotations, we compute the loss from distant supervision [77] following the original NPN training method [12]. For the material selection loss, labels are obtained

<sup>2</sup>To consider multiple words of materials, we divide the probability by the number of words.

whether or not each step contains materials in the material list. For the action selection loss, labels are obtained whether or not each step contains actions in the 384 actions defined by [12]. For example, from the sentence “crack the eggs and stir,” “eggs” is extracted as a material label, and “crack” and “stir” are extracted as an action label. As the action selection loss, the simple binary cross-entropy does not work in our preliminary experiment because the ratio of positive to negative actions is imbalanced; a few actions are positive and most of the actions are negative. Thus, as the action selection loss, we use asymmetric loss [142], which is a weighted binary cross-entropy loss, to defuse the imbalanced problem.

### Textual re-simulation loss $\mathcal{L}_{t\_sim}$

To train the textual re-simulator, we also compute the above visual simulation loss from the sampled procedural text.

### Total loss

Consequently, to train the entire model in an end-to-end manner, the total loss is computed as

$$\mathcal{L}_{total} = \mathcal{L}_{sent} + \mathcal{L}_{v\_sim} + \lambda \mathcal{L}_{t\_sim}, \quad (3.15)$$

where  $\lambda$  is a hyper-parameter used for weighting the importance of the textual re-simulation loss.

## 3.3 Experiments

We test the proposed method in the cooking domain and compare it with two state-of-the-art video captioning models on four evaluations: word-overlap evaluation, ingredient prediction, retrieval evaluation and qualitative analysis with thorough ablation. We also visualize the learned embedding of ingredients, demonstrating that the visual simulator effectively reasons the state transition of ingredients. Finally, we conduct experiments on the full prediction settings and discuss whether the proposed method performs well with the predicted materials or not.

Table 3.1 YouCook2-ingredient+ dataset statistics.

	train	val	test
#recipes	1,331	228	229
#steps / #recipes	7.8	7.6	7.7
#ingredients / #recipes	10.4	10.3	10.4

### 3.3.1 Experimental settings

#### Dataset construction

We use the YouCook2 dataset [147], which consists of 2,000 cooking videos from 89 recipes (=procedural text) categories. All of the videos have 3–16 events with a start/end timestamp annotated by humans, and each event is also annotated with an English sentence. Because ingredients (=materials) are not annotated in the original dataset, we prepared the YouCook2-ingredient+ dataset by annotating ingredients for recipes with 1,788 valid videos that are available online. Table 3.1 shows the statistics of the YouCook2-ingredient+ dataset.

**Annotation process.** To build the YouCook2-ingredient+ dataset, we hired three annotators to write ingredients that appear in the recipe using our developed web tool shown in Fig. 3.6. This annotation tool presents a recipe, corresponding video and text boxes for writing ingredients. In this paper, ingredients are defined as raw materials that are necessary to complete the dish. For example, “tomato” and “cucumber” should be written as ingredients, although “salad” should not be written because it represents a mixture of ingredients.

To annotate ingredients easily, “jump” buttons, which allow annotators to see an event corresponding to a step, are implemented based on the start/end timestamp from the original YouCook2 dataset. Moreover, to encourage annotators to write ingredients easily, this tool displays estimated ingredients using the named entity recognition (NER) model, flair [1] pre-trained on the English recipe flow graph corpus (E-rFG corpus) [138]<sup>3</sup>. If the estimated words are not appropriate for the ingredients, they can be deleted or rewritten.

#### Data preprocessing

As event features, we use concatenated features of appearance and optical flow provided by [147]. For the appearance, 2,048D feature vectors extracted from the “Flatten-673” layer in ResNet-200 [47] are used, and for the optical flow, 1,024D feature vectors extracted from the “global pool” layer in BN-Inception [51] are used. As in [65], we truncated sequences

<sup>3</sup>In the tag definitions of the E-rFG corpus, we display food entities as the estimated ingredients. These entities cannot be directly used for our dataset because the definition of food slightly differs from the ingredient definition in this paper (for example, “it” or “salad” are recognized as food in the E-rFG corpus). Therefore, we asked annotators to delete or rewrite ingredients if they were not appropriate.

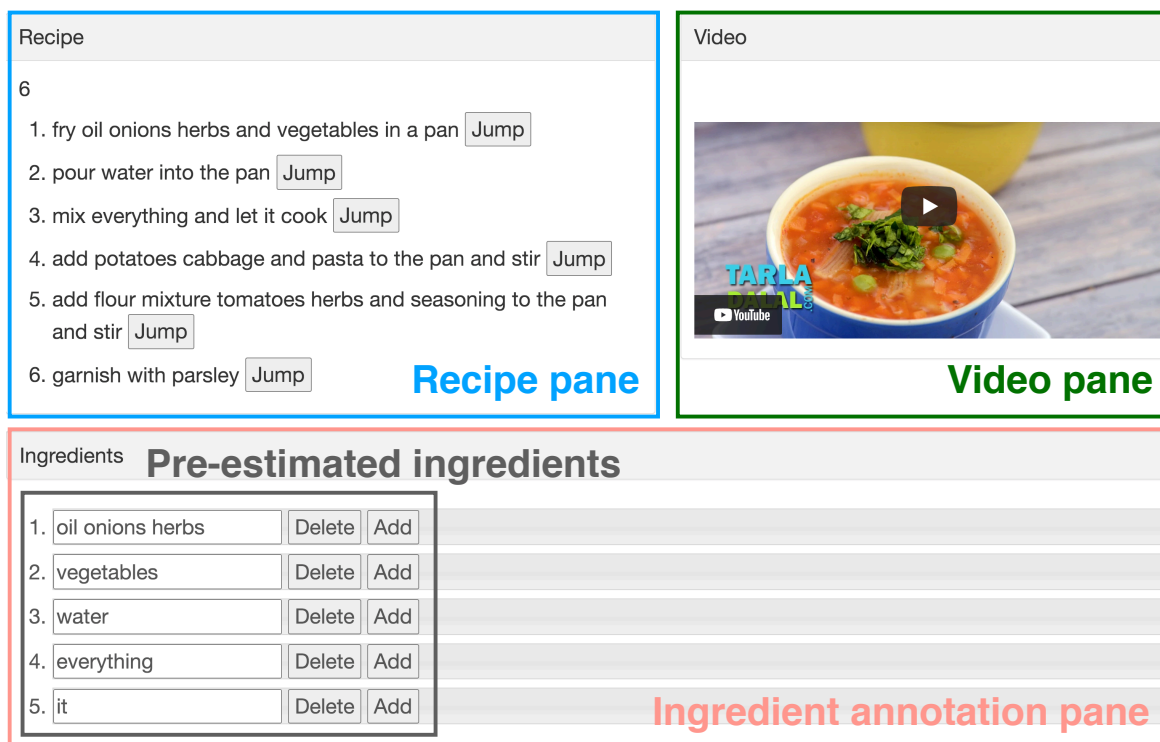


Fig. 3.6 A screen of our browser-based web annotation tool. Annotators write ingredients that appear in the recipes. To ease annotation, we estimated ingredients using the NER method [1] pre-trained on the English recipe flow graph corpus [138], and we set the default values of inputs.

longer than 100 for the event and 20 for the sentence and set the maximum length of the event sequence to 12. Finally, we built the vocabulary based on words that occurred three times or more, and the resulting vocabulary contained 991 words.

### Hyper-parameter settings

For both the encoder and decoder transformers, we set the hidden size to 768, the number of layers to two, and the number of attention heads to 12. We train the model following the optimization method described in [32, 65]; we use the Adam optimizer [56] with an initial learning rate of 0.0001,  $\beta_1 = 0.9$  and  $\beta_2 = 0.999$ . The L2 weight decay is set to 0.01, and the learning rate warmup is over the first five epochs. We set the batch size to 16, and continue training at most 50 epochs using early stopping with CIDEr-D. We tune  $\lambda$  with four different values  $\lambda \in \{0.25, 0.5, 0.75, 1.0\}$  and set  $\lambda$  to 0.5 in our experiments (for details, see Chapter 3.3.2).

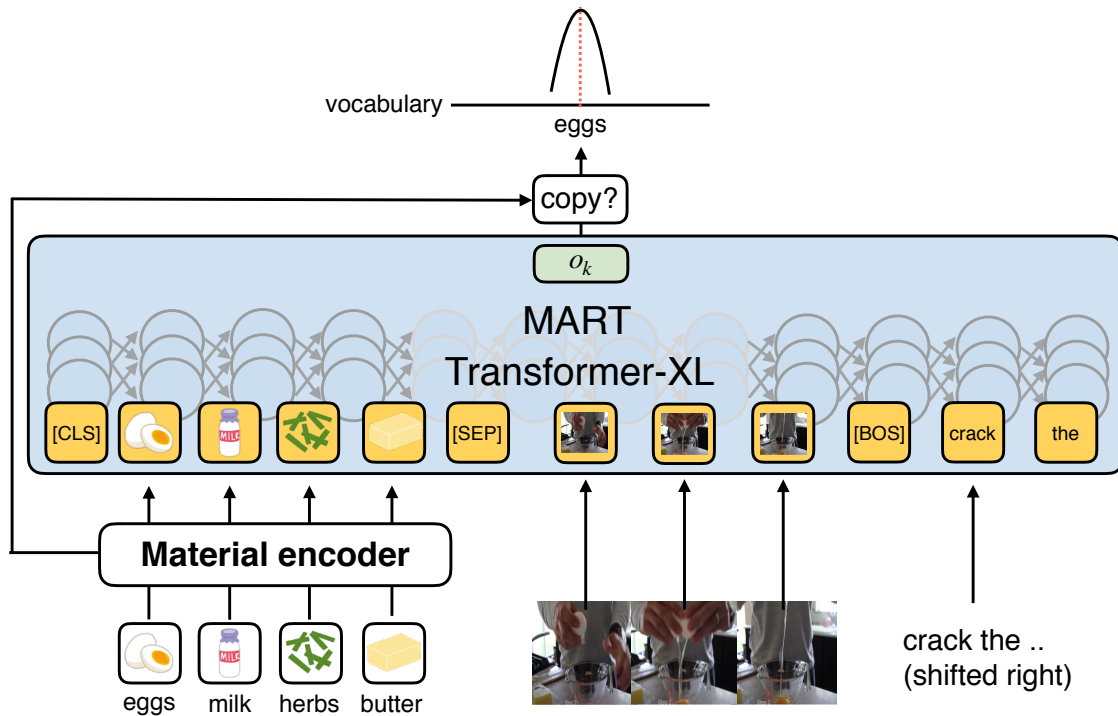


Fig. 3.7 An overview of our baseline +ingredients (-I) implementation. These models also incorporate the material encoder described in Chapter 3.2.2 and copy mechanism into the baselines.

### Baseline models

We test the proposed method by comparing it with two state-of-the-art video captioning models, as described below.

- **Transformer-XL** [25] is a powerful transformer-based language model that was originally proposed for capturing long-term dependency in natural language. As in [65], we adapt it for our task; the model directly uses all of the previous hidden states to generate a current sentence.
- **MART** [65] is a transformer-based video captioning model that achieves state-of-the-art performance on the video paragraph description. This model generates a sentence with a gated recurrent memory module, which does not pass all of the previous hidden states, but effectively summarize important information in the previous step.

**Ingredient-enriched models.** Note that these video captioning models originally have no ingredient set in the inputs and copy mechanism in the decoder. Thus for a fair comparison, we prepare for additional baselines, baseline + ingredient (-I) models, which incorporate the material encoder (Chapter 3.2.2) and the copy mechanism into the models.

Table 3.2 Paragraph- and sentence-level word-overlap evaluation for the baseline and the proposed models with ablation studies. The scores in bold are the best among the comparative models. “I” indicates whether the model uses ingredient information or not. B=BLEU, M=METEOR, C=CIDEr-D, RL=ROUGE-L.

Baseline	I	Paragraph-level							Sentence-level						
		B1	B2	B3	B4	M	C	RL	B1	B2	B3	B4	M	C	RL
Transformer-XL		39.0	22.0	12.1	6.7	15.2	22.7	30.9	26.5	13.5	5.3	1.5	10.6	57.3	27.8
+ Ingredients (Transformer-XL-I)	✓	37.7	22.5	13.4	8.2	15.4	35.4	34.2	29.6	17.0	8.2	3.5	12.6	69.6	31.3
MART		37.9	21.7	12.4	7.6	15.0	29.1	32.3	28.0	14.5	6.1	2.4	11.1	62.1	29.4
+ Ingredients (MART-I)	✓	42.3	26.2	16.1	9.9	17.6	48.2	36.2	31.3	18.5	9.4	4.4	13.7	81.0	32.9
Ours															
Video only (V)		43.2	24.5	14.0	8.1	16.6	32.4	31.9	28.0	13.9	6.0	1.9	11.4	60.7	28.3
V + Ingredients (VI)	✓	49.1	29.5	17.6	10.5	20.3	63.3	35.2	31.9	18.0	9.3	3.8	13.9	81.7	31.3
VI + Visual simulator (VIV)	✓	<b>49.4</b>	30.1	18.0	11.0	21.0	66.1	36.8	<b>33.7</b>	19.3	9.7	4.4	<b>15.2</b>	93.0	33.3
VIV + Textual re-simulator (VIVT)	✓	<b>49.4</b>	<b>30.9</b>	<b>18.3</b>	<b>11.3</b>	<b>21.1</b>	<b>67.1</b>	<b>37.1</b>	33.5	<b>19.4</b>	<b>10.1</b>	<b>4.9</b>	<b>15.2</b>	<b>96.7</b>	<b>33.7</b>

These models are based on the transformer that encodes sequential inputs and decodes a sentence by attending to all of the elements in the input sequence. Thus, to fit this characteristic, we concatenate the encoded ingredient and video vectors, and input them to the model, as shown in Fig. 3.7. When decoding, based on the output of the decoder  $o_k$  and ingredient vectors  $\mathcal{E}_0$ , the copy mechanism calculates the copying gate to make a soft choice between selecting an ingredient from the ingredient set or generating a word from the vocabulary.

### Ablations

To reveal the impact of the components in the proposed method, we conduct ablation studies on the following variations.

- **Video only (V)** encodes an event sequence with the event sequence encoder, then generates a procedural text.
- **V + Ingredient (VI)** incorporates the material encoder and copy mechanism in the model.
- **VI + Visual simulator (VIV)** incorporates the visual simulator into the VI model.
- **VIV + Textual re-simulator (VIVT)** additionally incorporates the textual re-simulator into the VIV model.

### 3.3.2 Word-overlap evaluation

To evaluate the captioning performance of the proposed method, we compute commonly used word-overlap metrics, such as BLEU [90], ROUGE-L [67], METEOR [8] and CIDEr-D [122] in the test set. We conduct two types of word-overlap evaluation: recipe (paragraph-level) [65, 91, 134] and step (sentence)-level evaluation [107].

Table 3.3 Change in paragraph-level word-overlap evaluation with controlled  $\lambda$ .

	B1	B2	B3	B4	M	C	RL
$\lambda = 0$ (VIV)	49.4	30.1	18.0	11.0	21.0	66.1	36.8
$\lambda = 0.25$	49.6	30.4	<b>18.3</b>	<b>11.3</b>	21.1	65.5	36.6
$\lambda = 0.5$	49.4	<b>30.9</b>	<b>18.3</b>	<b>11.3</b>	21.1	<b>67.1</b>	<b>37.1</b>
$\lambda = 0.75$	<b>50.3</b>	30.4	18.0	10.8	<b>21.2</b>	65.7	36.3
$\lambda = 1.0$	49.1	30.0	18.0	11.0	21.0	64.4	36.7

Table 3.4 Change in paragraph-level word-overlap evaluation on different viewpoints: egocentric, exocentric and mixed views.

	B1	B2	B3	B4	M	C	RL
VIV							
Exocentric	49.0	29.1	17.3	10.4	20.3	<b>68.3</b>	35.6
Egocentric	<b>50.5</b>	<b>31.3</b>	<b>18.9</b>	<b>11.8</b>	<b>21.7</b>	67.4	<b>37.6</b>
Mixed	47.3	28.6	16.7	10.0	20.5	65.8	36.8
VIVT							
Exocentric	48.7	29.3	17.3	10.5	20.4	62.2	35.1
Egocentric	<b>50.5</b>	<b>31.4</b>	<b>19.5</b>	<b>12.4</b>	<b>21.9</b>	<b>72.2</b>	<b>38.4</b>
Mixed	46.9	28.7	16.9	10.1	20.5	70.7	37.5

Table 3.2 shows the results of the word-overlap evaluation. We observe that the proposed method consistently outperforms the state-of-the-art captioning models by a significant margin in both paragraph- and sentence-level evaluation. Our ablation studies show that the VIV model performs better than the VI model, and the VIVT model further improves the VIV model. This indicates that both the visual simulator and the textual re-simulator are effective for generating a recipe accurately.

### Performance change of controlling the hyper-parameter $\lambda$

Table 3.3 shows the results by varying the  $\lambda \in \{0.0, 0.25, 0.5, 0.75, 1.0\}$ . Note that  $\lambda = 0$  is equivalent to the VIV model, which does not have a textual re-simulator. The results indicate that (1) the VIVT model achieves a performance that is better or comparable to the VIV model for any  $\lambda$  values, and (2)  $\lambda = 0.5$  performs the best among the three metrics (BLEU2-4, ROUGE-L, and CIDEr-D) and obtains competitive results in BLEU1 and METEOR. Thus, we set  $\lambda = 0.5$  in our experiments.

### Performance change on different viewpoints: egocentric, exocentric and mixed view

The YouCook2 dataset contains diverse videos featuring various viewpoints, including egocentric, exocentric, and mixed views. Fig. 3.8 shows an example of egocentric and

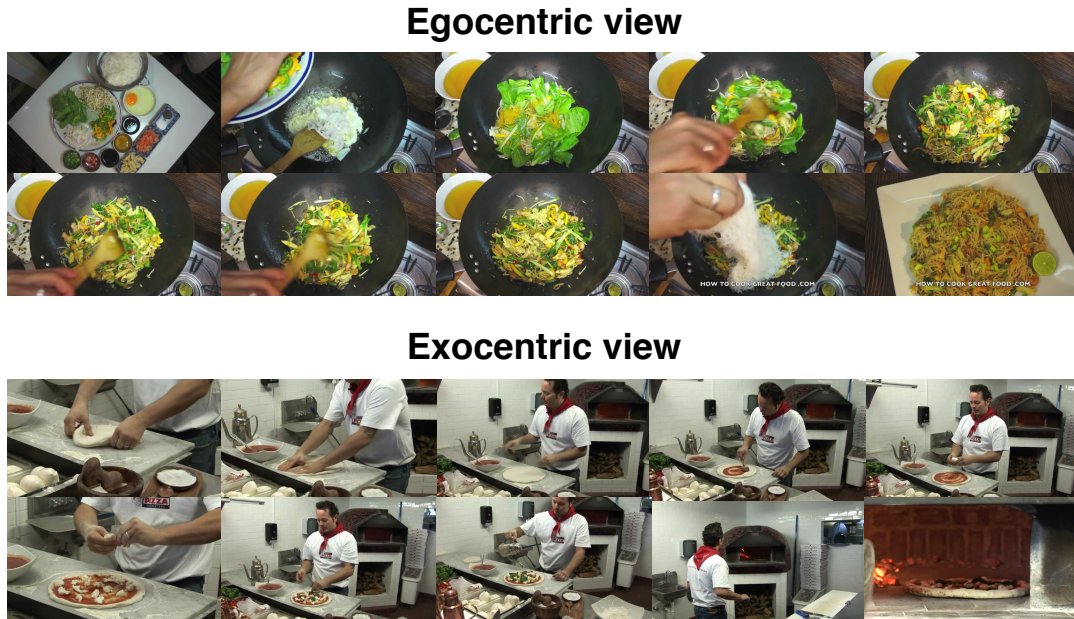


Fig. 3.8 An example of videos taken from different viewpoints: exocentric and egocentric views.

exocentric videos. Note that the mixed view represents that the videos have scenes taken from both egocentric and exocentric viewpoints. This arises a research question of whether the performance varies with viewpoints. To answer this, we partition the generated recipes in the test set into subsets based on their viewpoint types by annotating three types of view (egocentric, exocentric or mixed) with videos on the test set. As a result, we obtain 79 exocentric, 108 egocentric and 42 mixed-view videos. Then the paragraph-level word-overlap metrics are computed per each subset.

Table 3.4 shows the results of the word-overlap evaluation on VIV and VIVT models. We observe that both models achieve the best performance on the egocentric videos by a significant margin to other viewpoints. One of the possible reasons for this difference is that the egocentric videos highlight their work for the audience to understand, resulting in the success of the models to recognize the actions and ingredients accurately.

### 3.3.3 Ingredient prediction

To evaluate whether the models use the correct ingredients at each step without missing and hallucinating them, we design the ingredient prediction, which measures the step-level overlap of ingredients between generated and ground-truth recipes. To this end, we first construct an ingredient dictionary from all unique ingredients in the YouCook2-ingredient+



Ingredients	flour, eggs, baking soda, salt, pepper, water, shrimp, batter, breadcrumbs, oil		
	step 1	step 2	step 3
Key events			
MART + Ingredients (MART-I)	add <b>flour salt pepper</b> to a bowl ( <del>X</del> <b>baking soda, egg</b> )	add <b>flour salt pepper</b> and <b>milk</b> to the bowl and mix ( <del>X</del> <b>water</b> )	add <b>flour</b> and <b>shrimp</b> in the bowl ( <del>X</del> <b>batter, breadcrumbs</b> )
V + Ingredients (VI)	mix <b>flour salt pepper</b> and <b>breadcrumbs</b> ( <del>X</del> <b>baking soda, eggs</b> )	mix <b>flour salt pepper</b> and <b>breadcrumbs</b> ( <del>X</del> <b>water</b> )	coat the <b>shrimp</b> in the <b>batter</b> ( <del>X</del> <b>batter, breadcrumbs</b> )
VI + Visual simulator (VIV)	mix <b>flour baking soda salt pepper</b> and <b>breadcrumbs</b> ( <del>X</del> <b>eggs</b> )	mix the <b>eggs</b> with <b>eggs</b> ( <del>X</del> <b>water</b> )	coat the <b>shrimp</b> in the <b>batter</b> ( <del>X</del> <b>breadcrumbs</b> )
+ VIV + Textual re-simulator (VIVT)	mix <b>flour baking soda salt and pepper</b> together ( <del>X</del> <b>eggs</b> )	mix the <b>eggs</b> with the <b>water</b>	coat the <b>shrimp</b> in the <b>batter</b> ( <del>X</del> <b>breadcrumbs</b> )
Ground truth	add <b>flour eggs baking soda salt and pepper</b> to the bowl and stir	add cold <b>water</b> to the bowl and stir	cover the <b>shrimp</b> in the <b>batter</b> and <b>breadcrumbs</b>
	step 4	step 5	
Key events			
MART + Ingredients (MART-I)	fry the <b>shrimp</b> in <b>oil</b>	remove the <b>shrimp</b> from the <b>oil</b>	
V + Ingredients (VI)	fry the <b>shrimp</b> in a pan with <b>oil</b>	remove the <b>shrimp</b> from the <b>oil</b>	
VI + Visual simulator (VIV)	heat <b>oil</b> in a pan and fry the <b>shrimp</b>	remove the <b>shrimp</b> from the <b>oil</b>	
VIV + Textual re-simulator (VIVT)	heat <b>oil</b> in a pan and fry the <b>shrimp</b>	remove the <b>shrimp</b> from the <b>oil</b>	
Ground truth	place the <b>shrimp</b> into a pan of hot <b>oil</b>	remove the <b>shrimp</b> from the pan	

Ingredients	soy sauce, brown sugar, water, garlic, green onions, sesame oil, ribs		
	step 1	step 2	step 3
Key events			
MART + Ingredients (MART-I)	add <b>soy sauce sesame oil soy sauce sesame oil</b> and <b>sesame oil</b> to a pan ( <del>X</del> <b>brown sugar, garlic, green onions, water</b> )	cut the <b>ribs</b> into small pieces	add <b>soy sauce sesame oil soy sauce sesame oil</b> and <b>sesame oil</b> to a bowl ( <del>X</del> <b>marinade</b> )
V + Ingredients (VI)	add <b>chopped garlic sesame oil brown sugar</b> and <b>red pepper flakes</b> to a pan and mix ( <del>X</del> <b>green onions, water</b> )	put the <b>marinade</b> in the <b>ribs</b> and <b>soy sauce</b>	add <b>sesame oil garlic</b> and the bowl ( <del>X</del> <b>marinade</b> )
VI + Visual simulator (VIV)	add <b>garlic soy sauce sugar garlic green onions</b> and <b>garlic</b> to a bowl and mix ( <del>X</del> <b>sesame oil, water</b> )	cover the <b>ribs</b> with plastic wrap and place in a paper towel	pour the <b>marinade</b> over the ribs
+ VIV + Textual re-simulator (VIVT)	add <b>garlic sesame oil sugar soy sauce garlic</b> and <b>water</b> to a blender ( <del>X</del> <b>green onions</b> )	place the <b>ribs</b> in a bag	put the <b>ribs</b> into a bag ( <del>X</del> <b>marinade</b> )
Ground truth	combine <b>soy sauce brown sugar water garlic green onions</b> and <b>sesame oil</b>	place the <b>ribs</b> in the bag	pour the <b>marinade</b> into the bag
	step 4	step 5	
Key events			
MART + Ingredients (MART-I)	cook the <b>ribs</b> in the pan	cook the <b>ribs</b> on a grill	
V + Ingredients (VI)	flip the <b>pancakes</b> over	place the <b>ribs</b> on the grill	
VI + Visual simulator (VIV)	flip the <b>ribs</b> over	cook the <b>ribs</b> on a grill	
VIV + Textual re-simulator (VIVT)	grill the <b>ribs</b>	place the <b>ribs</b> on the grill and cook them until it is golden brown on both sides	
Ground truth	oil the grate of the grill	cook the <b>ribs</b> on the grill	

Fig. 3.9 Examples of generated recipes. Here, we compare four models, MART-I (baseline), VI, VIV and VIVT with the ground truth. Green bold and red words represent semantically correct and incorrect ingredients, respectively. Words in parentheses indicate missing ingredients, which should be included in the sentence. Note that parallel words in a sentence are not separated from the commas in the YouCook2 dataset (see step 1 in (a) in the ground truth).

Table 3.5 Results of ingredient prediction.

Baseline	Recall	Precision	F1
Transformer-XL	12.6	19.1	15.2
+ Ingredients (Transformer-XL-I)	17.3	30.4	22.0
MART	11.3	20.2	14.5
+ Ingredients (MART-I)	21.9	34.0	26.7
Ours			
Video only (V)	13.5	19.8	16.0
V + Ingredients (VI)	24.3	33.1	28.1
VI + Visual simulator (VIV)	28.9	<b>43.2</b>	34.7
VIV + Textual re-simulator (VIVT)	<b>29.7</b>	<b>43.2</b>	<b>35.2</b>

Table 3.6 Results of retrieval evaluation. ↓ indicates that lower is better.

Baseline	MedR (↓)	R@1	R@5	R@10
Transformer-XL	162.5	2.0	6.3	11.0
+ Ingredients (Transformer-XL-I)	139	1.6	8.1	13.6
MART	138.5	1.9	7.1	11.4
+ Ingredients (MART-I)	79	3.1	11.9	19.4
Ours				
Video only (V)	134	2.2	8.4	13.6
V + Ingredients (VI)	65	4.6	14.5	21.2
VI + Visual simulator (VIV)	<b>49</b>	4.6	<b>17.3</b>	25.1
VIV + Textual re-simulator (VIVT)	<b>49</b>	<b>5.2</b>	<b>17.3</b>	<b>25.2</b>
Ground truth	7	19.0	45.2	59.7

dataset. Then, at each step, we extract ingredients that are exact-matched between generated recipes and the ingredient dictionary. The same process is performed to extract ingredients in ground-truth recipes. Finally, based on the extracted ingredient sets, we compute the micro-recall, precision and F1 scores, respectively.

Table 3.5 shows the results of the ingredient prediction. This result shows that the proposed method outperforms the state-of-the-art video captioning models. In our ablation, we observe the same tendency of performance change to the word-overlap evaluation. We note that the VIV model performs much better than the VI model by 6.6% in F1, indicating that not only the copy mechanism but also the visual simulator are important for generating ingredients correctly. We also notice that the VIVT model improves the VIV model by 0.5% in F1, demonstrating the effectiveness of the textual re-simulator.

### 3.3.4 Retrieval evaluation

To evaluate whether the generated procedural text is sufficiently concrete to describe the input events, we design a step-level zero-shot sentence-to-event retrieval evaluation. As a retrieval model, we employ the MIL-NCE model [74] pre-trained on the HowTo100M dataset [75], achieving state-of-the-art performance. In this task, given a generated step-level sentence as a query, the MIL-NCE model embeds it and computes the cosine similarity as a score between the query vector and all the 1,768 event vectors from the test set. Then, we sort scores with events in descending order and calculate the median rank (MedR) and recall rate at the top  $K$  ( $R@K$ ). The median rank represents the median ranking of retrieved corresponding events, hence a lower is better; in contrast,  $R@K$  represents the percentage of all the step-level sentence queries where the corresponding event is retrieved in the top  $K$ , hence a higher is better.

Table 3.6 shows the results of the retrieval evaluation. We observe that the proposed method significantly outperforms the state-of-the-art video captioning models. In MedR, the VIVT model achieves 49, which is marginally lower than that of Transformer-XL-I 139 and MART-I 79. This indicates that the proposed method generates a more concrete recipe based on events than the state-of-the-art video captioning models. In our ablation, the VIV model dramatically improves the VI model, indicating that the visual simulator is essential for generating concrete recipes. In addition, the VIVT model shows a steady improvement from the VIV model, indicating the effectiveness of the textual re-simulator.

### 3.3.5 Qualitative analysis

Fig. 3.9 shows two examples of the generated recipes. We discuss the insights and limitations of the proposed method.

#### Insights

For our task, it is important to generate the correct ingredients that are manipulated in an event. MART-I fails to generate ingredients correctly. The model tends to miss and hallucinate ingredients (e.g., “flour” and “milk” in step 2 (a) and “garlic” in step 1 (b)). A similar tendency can be observed in the VI model, indicating that these models superfluously generate ingredients listed in the ingredient set.

The VIV model suppresses these problems (e.g., “baking soda” and “batter” in steps 1 and 3 in (a) and “marinade” in step 3 in (b)). In addition, owing to the textual re-simulator, the VIVT model can generate ingredients that are missed or unrecognized in the VIV model (e.g., “water” in steps 1 and 2 in (a) and in step 1 in (b)).

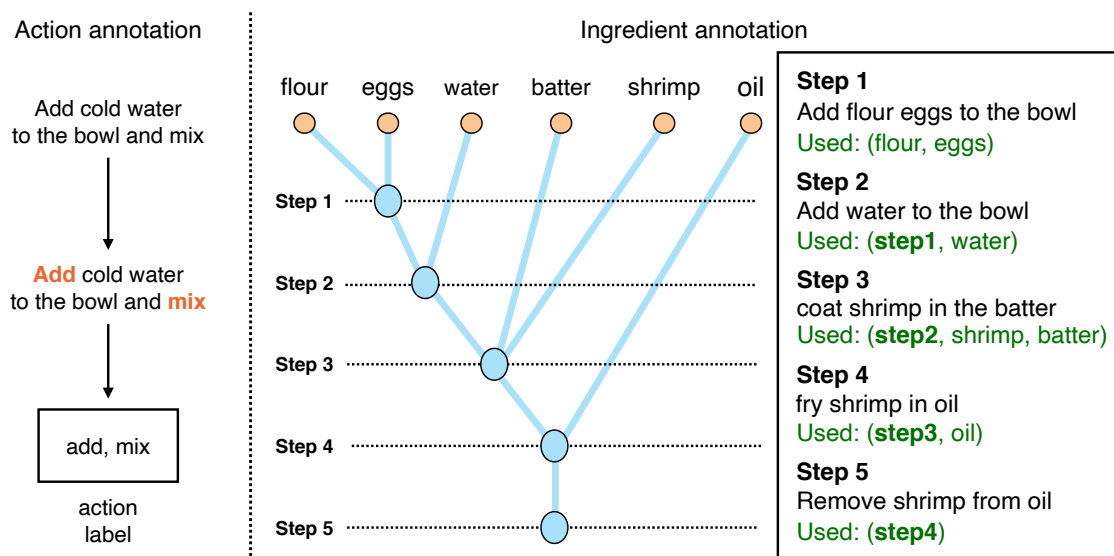


Fig. 3.10 An annotation example of the visual simulator.

Table 3.7 Quantitative evaluation of the visual simulators.

	Ingredient			Action		
	Recall	Precision	F1	Recall	Precision	F1
VIV	22.6	77.5	35.0	49.5	18.7	27.1
VIVT	21.4	75.9	33.4	49.1	19.9	28.3
Distant supervision	30.5	97.8	46.5	94.7	57.1	71.2

## Limitations

Although the proposed method generates recipes more accurately than the baseline models, we still found some differences from the ground truth. For example, in step 2 in (a), the VIV and VIVT models refer to “eggs,” superfluously, but only “water” is added to the ground truth. In addition, in step 3 in (b), the VIV and VIVT models generate “ribs” but it is not used. To solve these problems, we believe that incorporating fine-grained ingredient recognition modules [19] would help the model to generate a recipe more precisely.

## 3.3.6 Discussion of the learned embedding

### Quantitative evaluation of visual simulators

We evaluate the visual simulators by measuring the performance of the action and material selectors. These models are trained from distant supervision, as described in Chapter 3.2.6, which does not always agree with human judgment. Thus, we manually annotated the action and material labels for 50 recipes that were randomly sampled from the test set.

Fig. 3.10 shows an annotation example. For actions, we manually extracted action words from sentences at each step (e.g., “add” and “mix” were extracted in Fig. 3.10). For ingredients, step sentences do not explicitly include all of the ingredients manipulated in the real world because they mention only the difference from the previous steps. For example, in step 2 in Fig. 3.10, a mixture of “flour” and “eggs” is used, but it is not mentioned in the sentence. To address this, we annotated the ingredient tree structure [53, 85], where each step node indicates a mixture of ingredients. The tree structure allows us to determine which ingredients are used at each step by back-tracking its leaf nodes.

We compute micro-recall, precision and F1 as evaluation metrics for the visual simulator. Table 3.7 shows that VIV and VIVT achieve competitive results; while VIV performs better in terms of ingredient selection, VIVT performs better for action selection. We also note that even distant supervision performs only F1=46.5% in the ingredient selection. This happens because recipes mention only the difference from the previous steps; “flour” and “eggs” are manipulated in step 2 in Fig. 3.10, but are not mentioned. Therefore, distant supervision is not perfect for simulating human material manipulation in the real world. However, it effectively guides the model to predict incremental ingredients and is adequate for improving the captioning performance of the task.

### Visualization of the learned embedding

To investigate how the visual simulator represents the state transition of ingredients, we visualize the ingredient embedding by projecting it to 2D space using t-SNE [120]. Fig. 3.11 shows the projected learned embedding of ingredients obtained by the VIVT model. Note that only raw (red) and updated (blue)<sup>4</sup> ingredients are shown in the figure. This result shows that the raw and updated points are located on the outer and inner sides of the distribution, respectively<sup>5</sup>.

We also investigate the ingredients’ trajectory with the retrieved top-2 nearest ingredient vectors from updated ingredient vectors (see the zoomed parts of the figure). The retrieved ingredients indicate their state awareness. The ingredients with similar states are embedded into the same cluster in the vector space regardless of the difference in their recipe categories defined by [147]. For example, the near vectors of updated “milk” with the “add” state are also updated “milk” with “add”-like states (e.g., mix and pour). “tomatoes” also have the same tendency.

<sup>4</sup>The attention weight in the material selector was higher than 0.5.

<sup>5</sup>The raw and updated ingredients correspond to an embedding  $\mathcal{E}^0$  by the material encoder and an embedding  $\mathcal{E}^n$  updated from  $\mathcal{E}^0$  by the visual simulator, respectively.

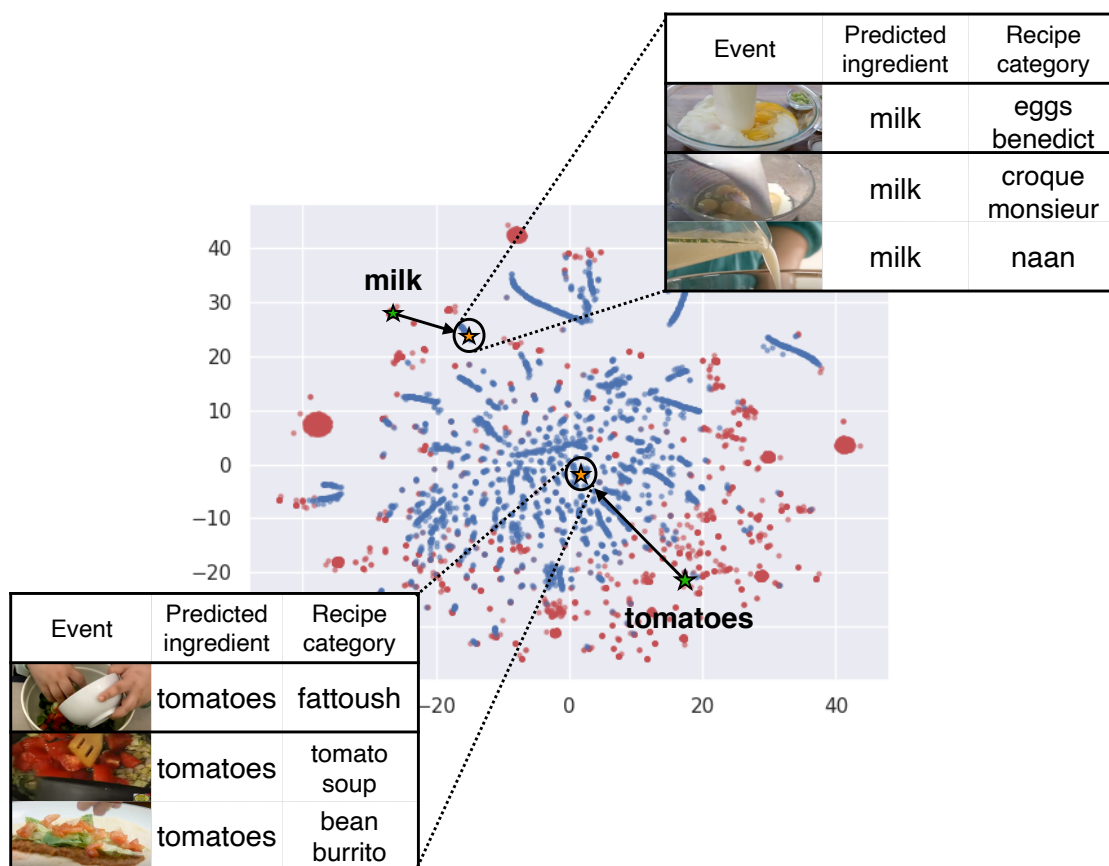


Fig. 3.11 Learned embedding of ingredients obtained by the VIVT model. Note that only raw and updated (the attention weight in the material selector is higher than 0.5) ingredients are transformed by t-SNE [120]. Red and blue colors represent the raw and updated ingredients, respectively.

### Semantic vector arithmetic

To demonstrate the state-awareness of learned embedding, we attempt to apply simple arithmetic operations as performed in the literature [76, 95, 101]. In the context of our task, the state transition of ingredients is expected to be computed as  $v(\text{cut potatoes}) = v(\text{potatoes}) + v(\text{cut tomatoes}) - v(\text{tomatoes})$ , where  $v$  represents the map in the embedding space.

Fig. 3.12 shows seven examples of arithmetic operations. From (a) to (d), first-order transformations (raw-to-updated) are described, and from (e) to (g), second-order transformations (updated-to-updated) are described. We can see that the learned embedding simulates the state transition of ingredients by specific actions in both transformations. For example, in (a) and (e), “canadian bacon” and “drained spaghetti” are converted into “cut canadian bacon” and “mixed drained spaghetti,” respectively. However, we observe some failure cases, where


















	Ingredient	+	Updated ingredient	-	Raw ingredient	=	Updated ingredient (nearest vector)
(a)	canadian bacon	+	 cut tuna	-	tuna	=	 cut canadian bacon
(b)	lettuce	+	 add tomatoes	-	tomatoes	=	 added lettuce
(c)	beef	+	 fry chicken	-	chicken	=	 fried beef
(d)	chicken	+	 cut tuna	-	tuna	=	 added chicken (fail)
(e)	 drained spaghetti	+	 mix rice	-	rice	=	 mixed drained spaghetti
(f)	 cut dogs	+	 fry chicken	-	chicken	=	 fried cut dogs
(g)	 cut potatoes	+	 fry chicken	-	chicken	=	 mixed potatoes (fail)

Fig. 3.12 Arithmetics using the learned embedding of ingredients. Examples (a) to (d) and (e) to (g) represent the first-order (raw-to-updated) and second-order (updated-to-updated) transformations, respectively. (d) and (g) show the failure cases for each transformation.

(d) and (g), “raw chicken” is transformed into “added chicken” via “cut” and “cut potatoes” into “mixed potatoes” via “fry.” In these examples, ingredients are consistent before and after, but executed actions are different because of the failure in action selection. We believe that noisy action labeling causes this failure, and a sophisticated action selection would ease this problem.

### 3.3.7 Experiments on the full prediction setting

This work discusses the proposed method under the condition that the input has ground-truth ingredients. Then, a natural question arises: is the proposed method effective with the predicted ingredients? To answer this question, we conduct experiments on the full prediction setting, where the material set is not given, but is predicted from the video events in advance. To achieve this, we add an ingredient decoder of the multi-label classifier and train the entire model as multi-task learning.

Fig. 3.13 shows how to integrate the ingredient decoder into the model. The ingredient decoder consists of a two-layered MLP with a sigmoid function and converts  $\hat{h}$  of a max-pooled vector of event vectors into a probability vector  $p \in \mathbb{R}^q$  of materials, where  $q$  indicates

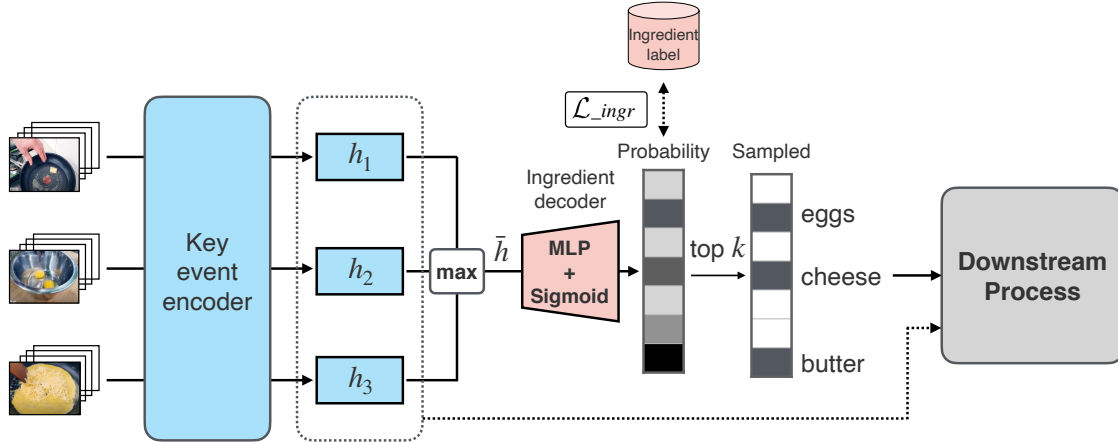


Fig. 3.13 An overview of how to integrate ingredient decoder into the model.

Table 3.8 Paragraph- and sentence-level word-overlap evaluation on the full prediction setting.

	I	B1	B2	B3	B4	M	C	RL	B1	B2	B3	B4	M	C	RL
Video only (V)		43.2	24.5	14.0	8.1	16.6	32.4	31.9	28.0	13.9	6.0	1.9	11.4	60.7	28.3
w/ Predict ingredients															
V + Ingredients (VI)	✓	42.5	24.3	13.9	8.1	16.5	28.4	32.3	28.5	14.6	5.9	2.2	11.8	62.1	28.7
VI + Visual simulator (VIV)	✓	42.5	24.3	13.9	8.1	16.7	33.9	32.5	28.9	14.6	6.2	2.3	11.9	63.8	28.8
VIV + Textual re-simulator (VIVT)	✓	42.7	24.0	13.3	7.7	16.7	31.6	31.9	28.3	14.0	5.5	2.3	11.7	60.8	28.3
w/ Ground-truth ingredients (Table 3.2)															
V + Ingredients (VI)	✓	49.1	29.5	17.6	10.5	20.3	63.3	35.2	31.9	18.0	9.3	3.8	13.9	81.7	31.3
VI + Visual simulator (VIV)	✓	49.4	30.1	18.0	11.0	21.0	66.1	36.8	33.7	19.3	9.7	4.4	15.2	93.0	33.3
VIV + Textual re-simulator (VIVT)	✓	49.4	30.9	18.3	11.3	21.1	67.1	37.1	33.5	19.4	10.1	4.9	15.2	96.7	33.7

the number of unique ingredients appearing more than three times in the training set (we obtained  $q = 668$  in the experiment). During training, we compute the ingredient decoder loss  $\mathcal{L}_{ingr}$ , which is an asymmetric loss [142] on the multi-label classification settings, and add it to the total loss. Note that we adopt teacher-forcing [132] to stabilize the training; while the models learn using the ground-truth ingredients for the downstream process in the training phase, they generate a recipe based on predicted ingredients at the inference phase (we sample the top  $k = 15$  ingredients from the probability). Another modification of the model is to remove the copy mechanism because we find that it degrades the captioning performance with this setting.

**Results.** We conduct three types of evaluations: word-overlap evaluation, ingredient prediction and retrieval evaluation. Table 3.8, 3.9 and 3.10 shows their results. Compared with the video-only (V) model, the performance of VI, VIV and VIVT models is competitive in the word-overlap evaluation and ingredient prediction and is worse for retrieval evaluation. Compared with the models with ground-truth ingredients, we observe a performance drop by a significant margin. This performance degradation likely occurs because of the accumu-



Table 3.9 Results of ingredient prediction on the full prediction setting.

	Recall	Precision	F1
Video only (V)	13.5	19.8	16.0
w/ Predict ingredients			
V + Ingredients (VI)	14.8	21.5	17.5
VI + Visual simulator (VIV)	16.2	22.5	18.8
VIV + Textual re-simulator (VIVT)	15.8	21.9	18.3
w/ Ground-truth ingredients (Table 3.5)			
V + Ingredients (VI)	24.3	33.1	28.1
VI + Visual simulator (VIV)	28.9	43.2	34.7
VIV + Textual re-simulator (VIVT)	29.7	43.2	35.2

Table 3.10 Results of retrieval evaluation on the full prediction setting.

	MedR ( $\downarrow$ )	R@1	R@5	R@10
Video only (V)	134	2.2	8.4	13.6
w/ Predict ingredients				
V + Ingredients (VI)	138.5	2.1	7.8	13.1
VI + Visual simulator (VIV)	133.5	1.6	7.2	12.1
VIV + Textual re-simulator (VIVT)	141	2.0	7.0	12.6
w/ Ground-truth ingredients				
V + Ingredients (VI)	65	4.6	14.5	21.2
VI + Visual simulator (VIV)	49	4.6	17.3	25.1
VIV + Textual re-simulator (VIVT)	49	5.2	17.3	25.2

lated errors of the ingredient decoder and visual simulators. Table 3.11 indicates that the performance of the ingredient decoder is around F1=34% to 35% on average. Therefore, we conclude that the proposed method performs well with the input of ground-truth ingredients, rather than the predicted ingredients. This is another limitation of our work.

## 3.4 Conclusion

In this chapter, we tackled a task for generating a procedural text from key events and materials. To generate accurate procedural text, it is essential for models to track material states in key events. To achieve this, we proposed a novel state-aware method, which modifies the existing simulator as a visual simulator and incorporates it into the encoder-decoder architecture. Our experimental results demonstrated the effectiveness of the proposed method, which outperforms the state-of-the-art video captioning models. The learned embedding of materials showed that the simulator effectively captures their state transition. Finally, we

Table 3.11 Performance of an ingredient decoder on the full prediction setting. Note that we compute the micro-recall, precision, and F1 on the multi-label classification setting.

	Recall	Precision	F1
VI	45.2	27.2	34.0
VI + Visual simulator (VIV)	46.0	27.8	34.7
VIV + Textual re-simulator (VIVT)	45.3	27.3	34.1

conducted experiments on the full prediction settings and found that the proposed method performs well on the ground-truth ingredients, rather than the predicted ingredients.

# Chapter 4

## Multimodal Recurrent Learning of the Event Selector and Sentence Generator

### 4.1 Introduction

In this chapter, we tackle the full prediction task of generating procedural text from unsegmented instructional videos: (1) extracting events from the videos and (2) generating descriptive sentences for them. As discussed in Chapter 1, this task is similar to DVC [57, 129, 30], which aims at detecting events *densely* from videos and generating sentences for them. Although the input/output pairs of our task (video/(events, sentences)) and that of the DVC are of the same format, our task is different from the perspective of the story awareness of instructional videos. DVC allows computers to exclusively detect false-positive events in its evaluation metrics. Our task, on the other hand, requires them to extract the accurate number of events in the correct order and generate sentences for them. Fujita *et al.* [37] reported that DVC models produce more than 200 redundant events per video on average on the ActivityNet captions dataset [57] while the number of manually-annotated events is only 3-4. Such redundant events and sentences make it difficult for users to grasp an overview of the video content.

The reason for the redundant outputs is that existing DVC approaches employed parallel prediction, wherein events and sentences are estimated independently (Fig. 4.1 top). These approaches consist of two modules: event prediction and sentence generation modules. Both modules do not take into account the surrounding events and sentences when predicting the current ones, leading to duplicated outputs. The choice of parallel prediction is reasonable for the DVC objective of thoroughly detecting events and generating sentences for multimedia

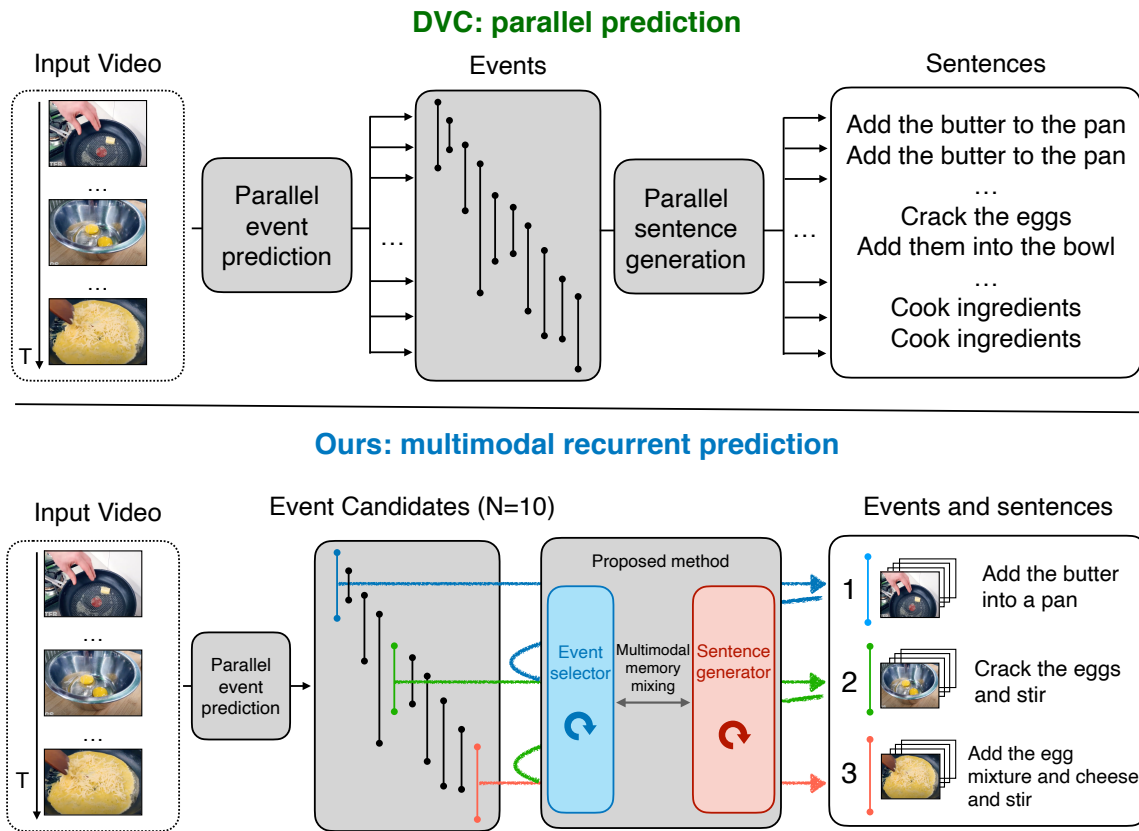


Fig. 4.1 A conceptual comparison of our approach and existing DVC studies. While the existing DVC models adopted parallel prediction, our approach employ multimodal recurrent prediction, which estimates events and sentences by memorizing and fusing the previously prediction results.

retrieval. However, it is not suitable for our procedural text generation task because we place high importance on providing an overview of videos that can be easily grasped by humans.

Although the events predicted by DVC models are redundant, we observed that (1) several events are adoptable as a story, but (2) the generated sentences for such events are not grounded well to the visual contents (i.e., materials and actions in the sentences are incorrect). We confirm this by analyzing the outputs of the state-of-the-art DVC model. We refer to the DVC output events as event candidates. Through an approach that we refer to as oracle selection, we select events that have the maximum temporal IoU (tIoU) to ground-truth events (i.e., manually-annotated events) and compute the DVC scores [37, 57]. The results support our observations. Although oracle selection ensures that events that are adoptable as the story of a procedural text are selected, the obtained sentences are not grounded in the visual content.

Based on this analysis, we set our goal to obtain correct procedural text by selecting oracle events from the event candidates and re-generating sentences for them. To realize this, the crucial idea is multimodal recurrent prediction, which estimates the next step by understanding previously predicted events and generated sentences. The bottom of Fig. 4.1 shows a concept of our idea. To predict the events and sentences in step 3, the models should understand both the visual and textual content of step 2. To achieve this, we propose a transformer[121]-based recurrent learning approach of an event selector and sentence generator. As with [65], both of the modules have memory representations, which remember previously selected events and generated sentences to predict events and sentences accurately. In addition, the proposed multimodal memory mixing method enables the modules to share the history of the previous prediction results, contributing the better performance. The overall model is designed to be trainable in an end-to-end manner because the modules of the model are connected without breaking a differentiable chain. We refer to this model as the **base** model.

We also propose an **extended** model in the settings where the inputs are videos with materials as in [84, 133]. The motivation behind this extension is to enhance the model’s ability to generate more accurate procedural text by incorporating the actual materials depicted in the videos. Relying solely on video inputs is problematic since even humans struggle to verbalize precise materials without additional contextual information. For example, the kinds of seasoning (e.g., salt and sugar) or meat parts (e.g., chuck and rib) are the exact cases. To develop a reliable procedural text generation system, it is essential for the models to accept supplementary material inputs.

This task requires agents to describe detailed manipulations of materials from instructional videos. To achieve this goal, our extended model has two additional modules: (1) a dot-product visual simulator and (2) a textual attention module. The dot-product visual simulator is introduced by the insights from Chapter 3, where we discovered that learning the material state changes contributes to generating accurate procedural text. The motivation for the extension is the difference in the inputs. Although the pre-segmented ground-truth events are given in Chapter 3, the inputs in this work are event candidates. The dot-product visual simulator accepts them and reasons about the state transition of materials. Furthermore, we also introduce a textual attention module that verbalizes grounded materials and actions in the procedural text. These modules are effective for grounded procedural text generation from instructional videos.

In our experiments, we use the YouCook2 [147] dataset and test the proposed method. It outperforms the state-of-the-art DVC approaches based on commonly used DVC metrics. In addition, the extended models boost the model’s performance. We also show that the

proposed models can select the correct number of events that effectively reflects the ground-truth events. In addition, our qualitative evaluation reveals that the proposed approaches can select events in the correct order and generate procedural text that reflects on the video contents. We discuss the detailed experiment settings for optimal procedural text generation.

In summary, our contributions are summarized as follows:

- This study focuses on enhancing the story awareness of procedural text, where agents need to accurately extract the number of events in the correct order and generate corresponding sentences. Based on the analysis of the DVC models, we set our goal to obtain accurate procedural text by selecting oracle events from the DVC output events and re-generating sentences for them.
- To achieve this goal, we propose transformer-based multimodal recurrent learning of an event selector and sentence generator. Building upon the predicted events from DVC models, our method effectively recurrently estimates the next step by memorizing and mixing previously predicted events and sentences.
- Furthermore, we extend our model to handle inputs comprising videos and materials. The motivation behind this extension is to enhance the model’s ability to generate more precise procedural text by incorporating actual materials observed in cooking videos. We introduce two additional modules to the base model, allowing it to verbalize detailed manipulations of materials in the instructional videos.
- Our experiments demonstrate that the proposed methods obtain the story awareness of procedural text. They outperform the DVC approaches especially on the story-oriented metrics, select the accurate number of events in the correct order, and generate accurate sentences for them. In addition, we investigate the detailed experiment settings for optimal procedural text generation.

## 4.2 Oracle-based analysis of the existing DVC model

As Fujita *et al.* [37] reported, the outputs of DVC models are redundant. However, we observed that although (1) several events are adoptable as a story, (2) the generated sentences for such events are not grounded well in the visual contents. To verify this, we analyze the outputs of the state-of-the-art DVC model. Specifically, we select events with the maximum tIoU scores to ground-truth events from event candidates and compute the DVC scores, an approach referred to as oracle selection hereinafter. The analyzed results demonstrate that the oracle selection boosts the performance, especially on story-oriented DVC metrics.

Table 4.1 Word-overlap metrics of the oracle selection on the YouCook2 dataset.  $N$  represents the number of candidate events, a hyper-parameter of PDVC. The bold scores are the best among the comparative settings.

	dvc_eval			SODA		
	BLEU	METEOR	CIDEr-D	METEOR	CIDEr-D	tIoU
PDVC (reproduced)	0.89	4.52	21.50	3.98	25.30	27.80
Oracle						
N=25	0.58	6.09	27.12	7.62	26.32	56.55
N=50	0.84	6.92	31.63	8.83	29.93	64.58
N=100	0.97	7.68	36.26	9.64	35.08	71.16
N=200	<b>1.10</b>	<b>8.15</b>	<b>38.60</b>	<b>10.43</b>	<b>36.89</b>	<b>76.71</b>

We use the YouCook2 dataset [147], one of the largest cooking video-and-language datasets. For the DVC model, we employ PDVC [129], the state-of-the-art DVC model. It detects  $N$  events densely and then re-ranks the top  $K$  of them for its outputs. Note that  $N$  is a hyper-parameter<sup>1</sup> and  $K$  is the prediction target. We adopt this model because it achieves the best performance on DVC tasks and it is easy to control  $N$  before training the model. We use the  $N$  detected events for the analysis, not the re-ranked  $K$  events.

### Evaluation metrics

We use two commonly-used DVC metrics: dvc\_eval [57] and SODA [37].

- **dvc\_eval** firstly computes tIoU of all the combinations between the prediction and ground-truth events and then computes word-overlap metrics (e.g., METEOR or CIDEr-D), if the tIoU scores are over the threshold  $\theta$ .  $\theta$  ranges from 0.3 to 0.9 by 0.2. Their average is the output score of these metrics.
- **SODA** stands for story-oriented dense video captioning evaluation, whereby the story awareness of the output events is evaluated. Specifically, it uses dynamic programming to explore an alignment of events between prediction and ground truth for obtaining the maximum story scores. The story scores are computed as a product of tIoU and word overlap metrics. Because SODA evaluates the predicted events by penalizing redundant outputs, it is suitable for computing story awareness.

This study thinks highly of SODA because it evaluates whether the output event/sentence pairs are appropriate as a story. As word-overlap metrics, we use BLEU [90], METEOR [8]

<sup>1</sup> $N$  is set to be a sufficiently large number. For YouCook2, the authors of [129] set  $N$  to be 100 as a default parameter.

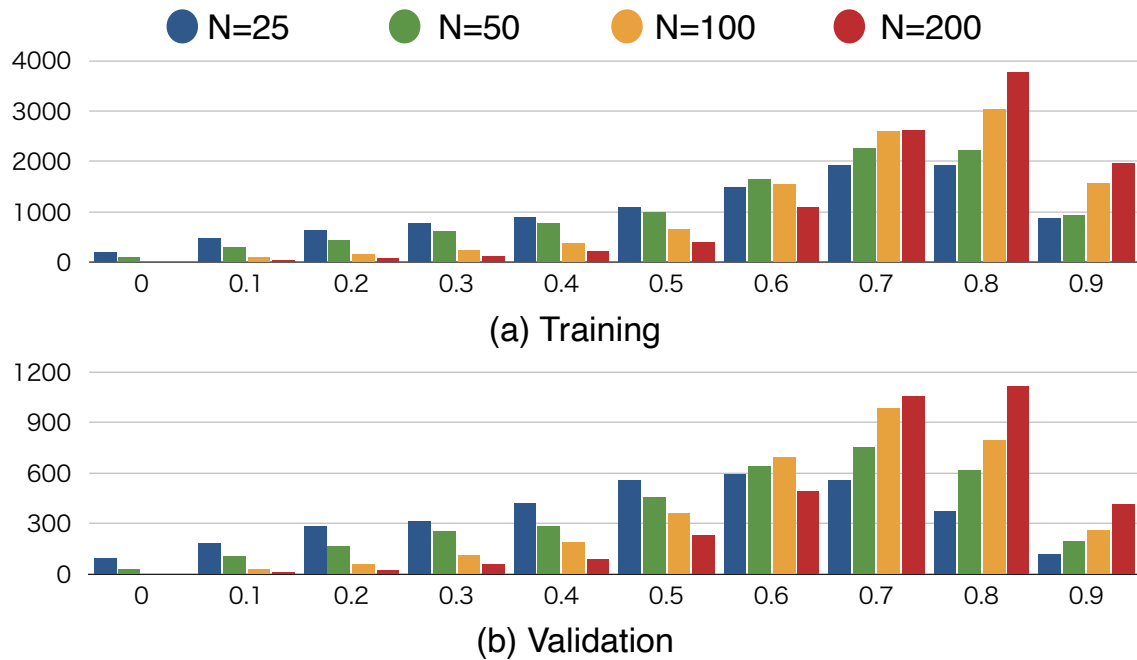


Fig. 4.2 tIoU distribution of oracle events on the training and validation sets of YouCook2.

and CIDEr-D [122], which are commonly-used metrics for text generation tasks. We also introduce SODA tIoU, which computes story scores as tIoU scores, rather than a product of tIoU and word overlap metrics. These metrics can evaluate whether the selected events are appropriate as components of the generated procedural text. In this analysis, we range  $N$  from 25, 50, 100, and 200.

#### 4.2.1 Quantitative evaluation

Table 4.1 shows the results of the oracle selection on `dvc_eval` and SODA metrics, indicating that the oracle selection outperforms the PDVC. Specifically, the SODA scores of the oracle selection are quite better than that of the PDVC, demonstrating that the generated procedural text is more suitable as a story than the ones generated by the PDVC.

Fig. 4.2 shows the distribution of tIoU scores of the oracle events on training and validation sets. The number of candidate events  $N$  was directly proportional to the average tIoU. This is because the more  $N$  was, the more suitable oracle events appear in the candidate. Both the training and validation sets confirm this tendency.



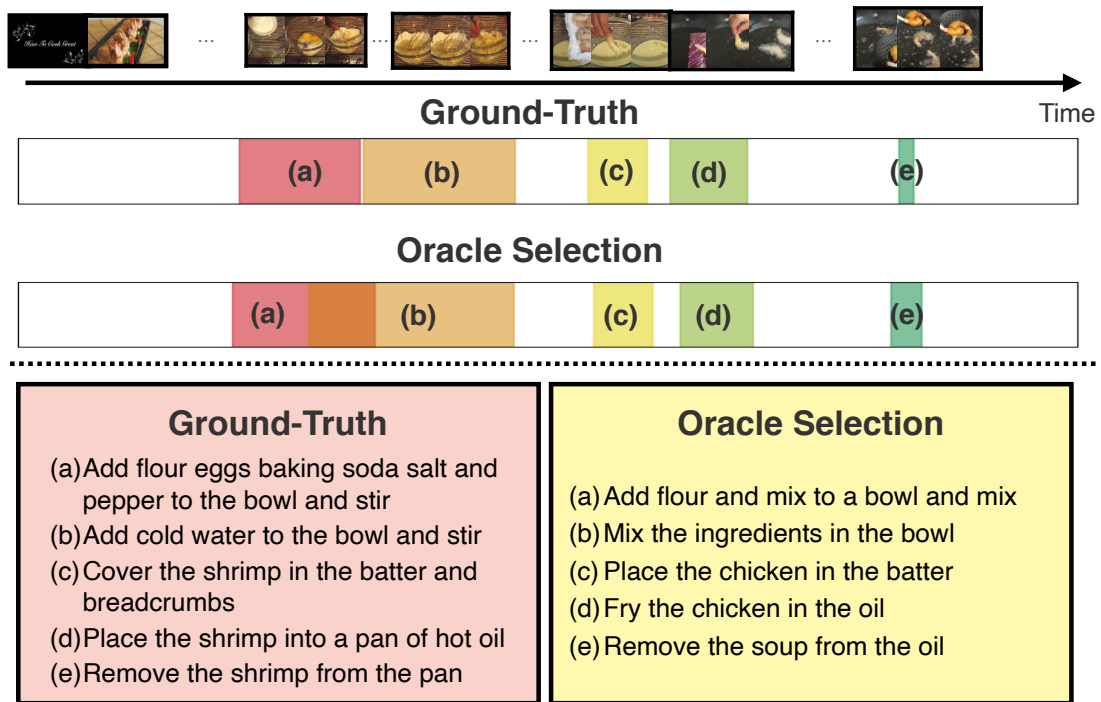


Fig. 4.3 Comparison of the procedural texts generated by the oracle selection and ground truth.  $N = 100$ , which is a default hyper-parameter of PDVC, is used in this example.

### 4.2.2 Qualitative evaluation

Fig. 4.3 shows a comparison of the procedural text generated by the oracle selection and ground truth. The selected event timestamps are close to the ground-truth events, indicating that the appropriate selection can construct the correct procedural text. However, the sentences are different from the ground truth (e.g., “baking soda”, “salt”, and “pepper” are missing in step (1)).

The main reason for this error is parallel prediction, where the events and sentences are predicted independently. This causes the model to miss or hallucinate materials and motivates us to propose a model that re-generates sentences for the predicted events, rather than using the corresponding generated sentences without modification.

## 4.3 Proposed method

Based on the oracle-based analysis, we set our goal to obtain correct procedural text by selecting oracle events from the output events of the DVC model and re-generating sentences for them. To achieve this goal, multimodal recurrent prediction is essential, which memorizes

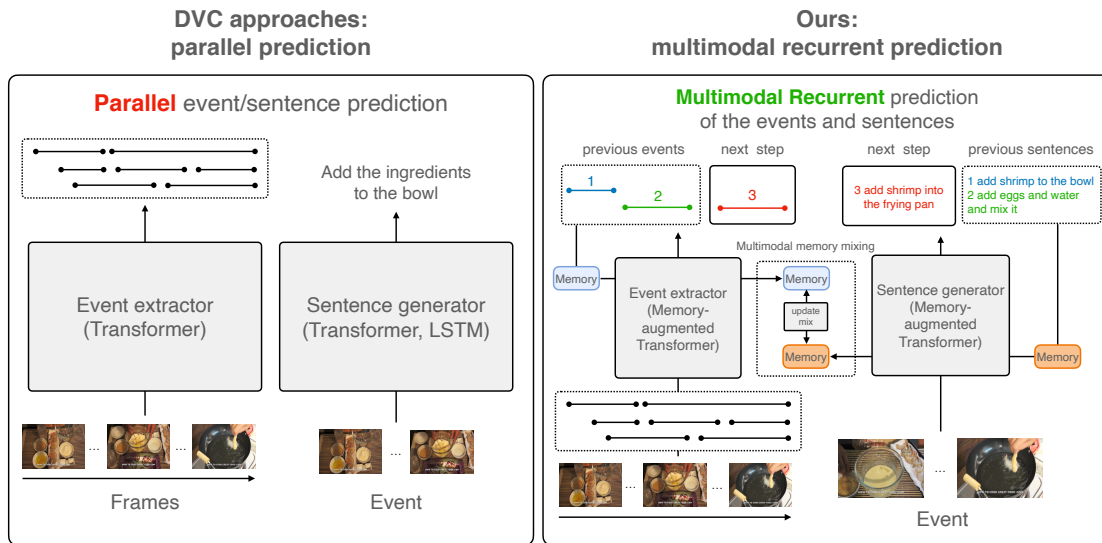


Fig. 4.4 An introductory overview of our approach. Unlike the previous DVC approaches, we propose a multimodal recurrent learning approach to train the event selector and sentence generator. Both modules represent the previously predicted events and sentences as memory vectors and predict the next step. These memory vectors are updated and mixed to effectively share the previous prediction belonging to different modalities.

and mixes the previously predicted events and generated sentences to estimate the next step (Fig. 4.4).

We realize this idea by proposing transformer-based models, consisting of event selection and sentence generation modules (Fig. 4.5), which we refer to them as an event selector and sentence generator. The event selector chooses oracle events from event candidates repeatedly (Chapter 4.3.1) and the sentence generator outputs sentences for the selected events (Chapter 4.3.1). Both encoders are based on memory-augmented recurrent transformers (MART) [65], which contain memory vectors to learn the recurrence by remembering the previous prediction to estimate the next step. In addition, the proposed multimodal memory mixing approach enables the model to share the previous prediction belonging to different modalities by fusing their memory vectors (Chapter 4.3.3).

It is worth noting the advancements over MART. This study presents two improvements:

- **Task extension.** We extend the scope of the model to verify both event selection and sentence generation tasks, whereas the original paper only validated its effectiveness in generating sentences from pre-segmented key events.

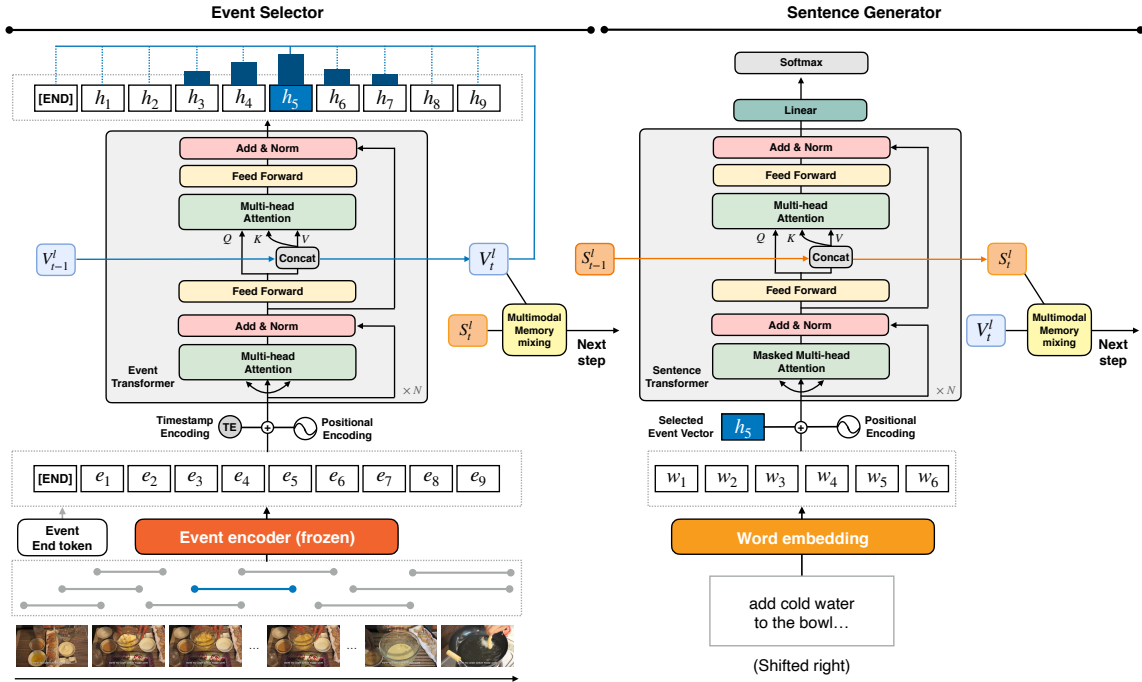


Fig. 4.5 Multimodal recurrent learning approach of the event selector and sentence generator for recipe generation from unsegmented cooking videos. The event selector chooses oracle events from event candidates repeatedly (Chapter 4.3.1) and the sentence generator outputs sentences for the selected events (Chapter 4.3.1). The memories are updated and mixed to effectively remember the history of the events/sentences for predicting the next step (Chapter 4.3.3).

- **Multimodal memory mixing.** We introduce a novel memory mixing method that integrates memory vectors from both modules to ensure coherent procedural text generation (Chapter 4.3.3).

We formulate this task using notations. Let  $X = (x_1, x_2, \dots, x_n, \dots, x_N) \in \mathbb{R}^{2 \times N}$  be event candidates, where  $x_n$  consists of start and end timestamps. Note that  $X$  is sorted on the start time of the events in chronological order. Given  $X$ , the model generates pairs  $(C, Y) = ((c_1, y_1), \dots, (c_t, y_t), \dots, (c_T, y_T))$ , where  $c_t$ ,  $y_t$  and  $T$  represent an index of the oracle event candidates, corresponding generated sentences and the number of the selected events, respectively. The memories  $V_t^l$  in the event selector and  $S_t^l$  in the sentence generator are updated at each  $t$  step, where  $l$  represents the layer number of transformers.

### 4.3.1 Event selector

The event selector can be divided into two main components: event encoder and event transformer. The event encoder converts given candidate events  $X$  into event-level representations  $E = (e_1, e_2, \dots, e_n, \dots, e_T)$ , based on which the event transformer outputs a holistic representations of events  $H = (h_1, h_2, \dots, h_n, \dots, h_T)$ . They are used for computing event probabilities, which represent the likelihood of oracle events. We apply Gumbel softmax resampling [52] to select events and forward them to the sentence generator without breaking a differentiable chain during the training phase. In the inference phase, events are deterministically selected by applying argmax to event probabilities.

#### Event encoder

The event encoder converts events  $x_n$  into representations  $e_n$ . First, we extract events from the video, according to the start and end time of  $x_n$ , and input them into pre-trained visual encoders. One of the straightforward ways to encode events is to average frame-level image representations extracted by ResNet [47]. However, we find that this approach is unable to capture fine-grained event semantics for overlapped events. For example, assume that the ground-truth event is a scene of cutting potatoes, and the event candidates have two scenes, where one is a scene of cutting potatoes and tomatoes, and another is of cutting only potatoes. We expect the model to select the latter scene because the former contains extra information about cutting tomatoes. Representing events based on the average of frame-level features cannot capture the semantic difference of these events effectively. Thus, it is necessary to employ an event encoder that focuses on extracting fine-grained event-level semantics.

To achieve this, we focus on the multiple instances learning-noise contrastive estimation (MIL-NCE) model [74] pre-trained on Howto100M [75]. The model is trained on more than 100M pairs of an event with narration, and it can capture event-level semantics of overlapped events. Using this encoder, we obtain event-level representations as  $E = (e_1, e_2, \dots, e_n, \dots, e_T)$ . Then, the positional encoding (PE) [121] and relative encoding of events [130] are added to  $e_n$ .

#### Event transformer

Given  $E$ , the event transformer outputs holistic representations of events  $H = (h_1, h_2, \dots, h_n, \dots, h_T)$ . This module is based on the memory-augmented recurrent transformer [65], where each  $l$  layer has the memories  $V_t^l$  to remember the history of the selected events. The output vectors  $H = (h_1, h_2, \dots, h_n, \dots, h_T)$  and memories  $V_t^l$  are used to compute  $n$ -th event probability

$p(c_t = n|V_t^l, X)$  as follows:

$$V_t = \max(V_t^1, \dots, V_t^l, \dots, V_t^L), \quad (4.1)$$

$$p(c_t = n|V_t^l, H) = \frac{\exp\{(h_n^t)^\top V_t\}}{\sum_i \exp\{(h_i^t)^\top V_t\}}, \quad (4.2)$$

where  $\max(\cdot)$  represents an element-wise max-pooling of vectors. During training, the event selector selects events through a Gumbel softmax resampling [52], which enables us to train the model in an end-to-end manner without breaking a differentiable chain. This indicates that the loss computed on the sentence generator is backpropagated to the event selector. During inference, the model selects the event index based on the argmax of  $p(c_t = n|V_t^l, H)$ .

### 4.3.2 Sentence generator

Based on the selected event representations, the sentence generator outputs sentences grounded for them. Let the selected event index be  $\hat{c}_t$ , and the selected event representation can be written as  $h_{\hat{c}_t}$ . This vector is added to the word vectors  $W = (w_1, w_2, \dots, w_k, \dots, w_K)$  from the word embedding layer, which is the concatenated neural networks of the pre-trained global vectors for word representation (GloVe) [92]<sup>2</sup> and one-layer perceptron with ReLU activation. We also add PE to word vectors  $W$  and input them into the sentence transformer, another memory-augmented transformer. The model generates words repeatedly by applying softmax and argmax operations to the output vectors of the sentence transformer.

### 4.3.3 Multimodal memory mixing

One of the important contributions of this study is to mix the memories for sharing the prediction results between the event selector and sentence generator. The motivation of this approach is that mixing memories is intuitively effective for coherent procedural text generation because the history of the selected events contributes to sentence generation and vice versa. Specifically, the memory vectors  $V_t^l, S_t^l$  are first separately updated by following the equations described in Section 3.2 in MART paper, and then mixed as follows:

$$\hat{V}_t = f_1(V_t) \odot \sigma(g_2(g_1(S_t))), \quad (4.3)$$

$$\hat{S}_t = g_1(S_t) \odot \sigma(f_2(f_1(V_t))), \quad (4.4)$$

<sup>2</sup>We employ pre-trained 300D word embedding, which can be downloaded from <http://nlp.stanford.edu/data/glove.6B.zip>

where  $f_*(\cdot), g_*(\cdot)$  represents a single linear layer and  $\odot, \sigma$  represents the Hadamard dot product and sigmoid function, respectively. The obtained  $\hat{V}_t$  and  $\hat{S}_t$  are forwarded into the next  $t + 1$  step. Note that we compare this with the original MART algorithm that updates memory vectors separately, and confirm its effectiveness in Chapter 4.5.4.

### 4.3.4 Loss functions

To train the model in an end-to-end manner, we sum up two types of losses: event selection and sentence generation loss. These losses are formulated as a negative log-likelihood of the event selection and sentence generation tasks as follows:

$$L_{base} = L_e + L_s, \quad (4.5)$$

$$L_e = - \sum_{(X,C)} \sum_t \log p(c_t | X, V_{<t}^l), \quad (4.6)$$

$$L_s = - \sum_{(X,C,Y)} \sum_t \log p(y_t | h_{\hat{c}_t}, S_{<t}^l), \quad (4.7)$$

where  $L_{base}, L_e$ , and  $L_s$  represent the base, event selection and sentence generation losses, respectively.

## 4.4 Extended model

In addition to the base model, we also propose an extended model that can generate more accurate procedural text in the settings where the inputs are videos with materials. The motivation behind this extension is to enhance the model’s ability to generate more accurate procedural text by incorporating the actual materials depicted in the videos. Relying solely on video inputs is problematic since even humans struggle to verbalize precise material names without additional contextual information. In Fig. 4.10, for instance, the correct material is “pork,” yet the base model mistakenly generates “meat” and “veal.” To develop a reliable procedural text generation system, it is essential for the models to accept supplementary material inputs.

This task requires computers to describe detailed manipulations of materials from instructional videos. To achieve this goal, our extended model has two additional modules: (1) dot-product visual simulator and (2) textual attention module (Fig. 4.6). The dot-product visual simulator is introduced based on the insights from Chapter 3, where we discovered that learning the material state changes is effective for procedural text generation. Although pre-segmented ground-truth events are regarded as input in our previous work, the event

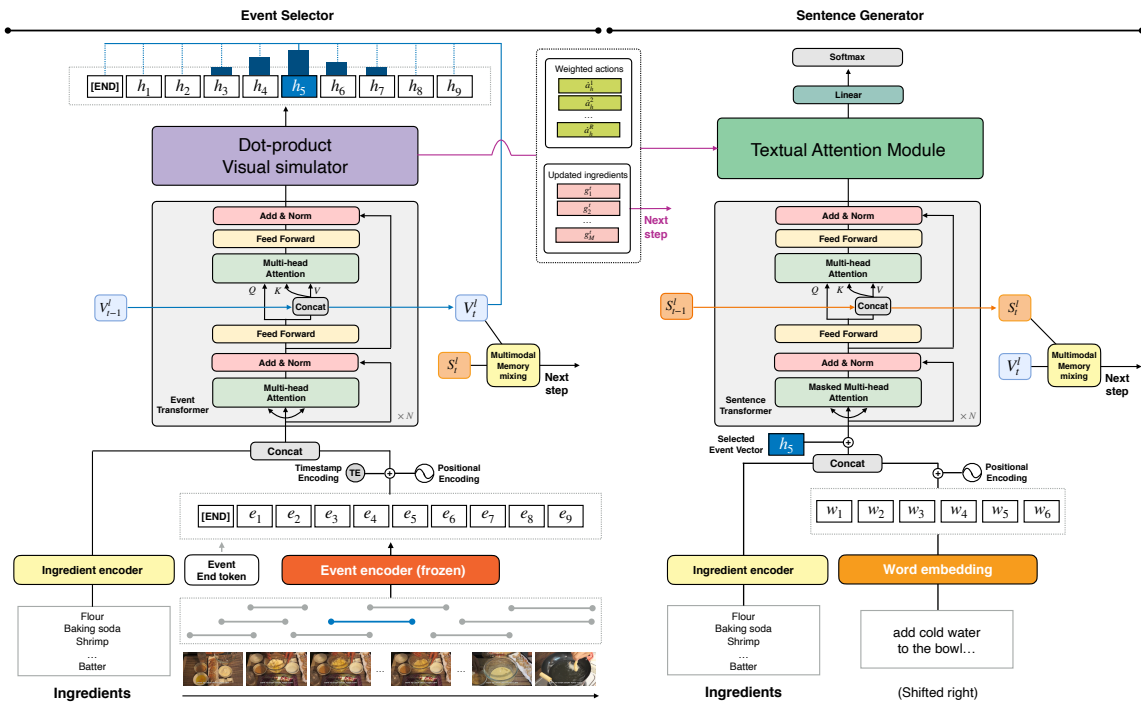


Fig. 4.6 An overview of the extended model for procedural text generation from unsegmented instructional videos. To generate more accurate procedural text, it has additional two modules: (1) dot-product visual simulator and (2) textual attention. The dot-product visual simulator is introduced to learn the state transition of materials. The textual attention module encourages the sentence generator to verbalize actions and materials more accurately.

candidates should be considered in this study. To this end, we extend it to the dot-product visual simulator described in Chapter 4.4.1. In addition, we also introduce a textual attention module that verbalizes grounded materials and actions in the procedural text. These modules are effective for grounded procedural text generation from instructional videos.

Another extension is to add a material encoder to convert materials into representations  $G^0$ . The material encoders are concatenated neural networks of pre-trained GloVe [92] word embedding and MLPs with ReLU activation function and are added to the event selector and sentence generator without sharing their parameters. Note that multi-word materials (e.g., parmesan cheese) are represented by the average embedding vector of the words. They are concatenated with event/word representations and inputted to the event/sentence transformers as shown in Fig. 4.6.

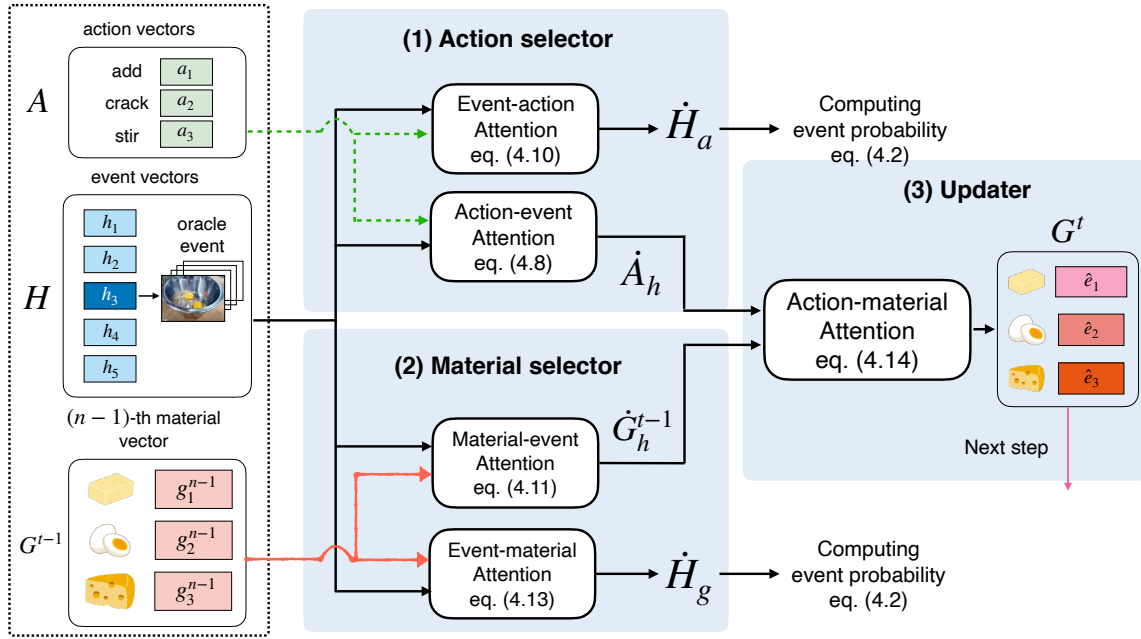


Fig. 4.7 An overview of the extended dot-product visual simulator that reasons about the state transition of materials. It has three components: (1) action selector, (2) material selector and (3) updater. The dot-product attention is employed to treat the event candidates.

#### 4.4.1 Dot-product visual simulator

We first revisit the original visual simulator and then describe how we extend it to the dot-product visual simulator.

##### Visual simulator revisit

In Chapter 3, the inputs are the pairs of materials and ground-truth events. Let  $G^0 = (g_1^0, g_2^0, \dots, g_m^0, \dots, g_M^0)$  and  $\hat{H} = (\hat{h}_1, \hat{h}_2, \dots, \hat{h}_{\hat{t}}, \dots, \hat{h}_{\hat{T}})$  be the material and ground-truth event vectors encoded by the material encoder (e.g., GloVe) and event encoder (e.g., MIL-NCE), respectively. Given  $(G^0, \hat{H})$ , the visual simulator reasons about the state transition of the materials by updating them at each  $\hat{t}$ -th step. To this end, it consists of three components: (1) action selector, (2) material selector and (3) updater. The action selector and material selector predict executed actions and used materials at  $\hat{t}$ -th step. Based on the selected actions/material, the  $(\hat{t} - 1)$ -th material vectors  $G^{(\hat{t}-1)}$  are updated into  $\hat{t}$ -th new proposal material vectors  $G^{(\hat{t})}$ , which are forwarded into the next step. The visual simulator recurrently repeats the above process until processing the end element of the ground-truth events.



### Proposed extension

The proposed model is similar to the visual simulator because it also has the same three components to update the material vectors. However, instead of the ground-truth event vectors  $\hat{H}$ , the event candidates  $H$  are assumed to be the inputs in this study, that is, the model needs to predict not only the actions/materials but also the events that constitute the story.

To achieve this, we extend the visual simulator into the dot-product visual simulator shown in Fig. 4.7. It computes the relationships between action-to- and material-to-events as attention matrices and outputs four vectors: action-weighted and material-weighted event vectors, event-weighted action and material vectors. The former two vectors are used to calculate the event probabilities and the latter two vectors are forwarded to the textual attention module.

### Action selector

The action selector outputs event-weighted action and action-weighted event vectors by predicting the executed actions and related events. Let the action vectors be  $A = (a_1, a_2, \dots, a_R)$ , that is, the pre-defined action embedding, where  $a_r$  represents  $r$ -th actions and  $R$  is the number of the actions. In Fig. 4.7, the actions “crack” and “stir” are executed at the oracle event  $h_3$ ; thus  $a_*$  that corresponding crack and stir indices should be selected with the relation to  $h_3$ . This computation can be formulated as the dot-product attention as follows:

$$\hat{A}_h = \text{softmax}\left\{\frac{(W_a^Q A)^T (W_h^K H)}{\sqrt{d}}\right\}, \quad (4.8)$$

$$\dot{A}_h = \hat{A}_h (W_h^V H)^T, \quad (4.9)$$

where  $W_a^Q, W_h^K, W_h^V$  represents a linear layer and  $d$  represents the dimension size of  $A$  and  $H$ . Note that  $\text{Softmax}(\cdot)$  represents the row-wise softmax operation on a matrix. We also acquire action-weighted event vectors  $\dot{H}_a$  as follows:

$$\dot{H}_a = \text{softmax}\left\{\frac{(W_h^Q H)^T (W_a^K A)}{\sqrt{d}}\right\} (W_a^V A)^T, \quad (4.10)$$

where  $W_h^Q, W_a^K, W_a^V$  represents a linear layer.

### Material selector

The material selector outputs the event-weighted material and material-weighted event vectors by predicting the materials used and related events. For example, in Fig. 4.7, the raw “eggs”

should be selected at  $h_3$ . This computation is achieved by replacing  $A$  with  $G^{t-1}$  in Eq (4.9) and Eq (4.10) as follows:

$$\hat{G}_h^{t-1} = \text{softmax}\left\{\frac{(W_g^Q G^{t-1})^T (W_h^K H)}{\sqrt{d}}\right\}, \quad (4.11)$$

$$\dot{G}_h^{t-1} = \hat{G}_h^{t-1} (W_h^V H)^T, \quad (4.12)$$

$$\hat{H}_g = \text{softmax}\left\{\frac{(W_h^Q H)^T (W_g^K G^{t-1})}{\sqrt{d}}\right\} (W_g^V G^{t-1})^T, \quad (4.13)$$

where  $W_g^Q, W_g^K, W_g^V$  represents a linear layer.

### Updater

The updater represents the state transition of the materials by updating the material vectors  $G^{t-1}$ . Based on the selected actions and materials  $(\dot{A}_h, \dot{G}_H^{t-1})$ , the updater computes the updated material representations as follows:

$$G^t = G^{t-1} + \dot{G}_H^{t-1} \odot \text{repeat}(\max(\dot{A}_h)), \quad (4.14)$$

where  $\text{repeat}(\cdot)$  expands the max-pooled action vector by repeating it  $M$  times ( $M$  is the number of materials).

### Output representations

The action- and material-weighted event representations  $(\dot{H}_a, \dot{H}_g)$  are used to compute the event probability by replacing  $H$  with  $\dot{H} = H + \dot{H}_a + \dot{H}_g$  in Eq (4.2). The event-weighted action vectors  $\dot{A}_h$  and updated materials  $G^t$  are forwarded to the textual attention module.  $G^t$  is also set to be the material vectors at  $(t+1)$ -th prediction.

#### 4.4.2 Textual attention

The textual attention module encourages the sentence generator to output accurate procedural based on the actions and materials. Let  $\hat{W} = (w_1, w_2, \dots, w_k, \dots, w_K)$  be the output word vector sequence from the sentence transformer. Given  $(G^t, \dot{A}_h)$  and  $\hat{W}$ , the textual attention module computes two attention matrices: (1) word-action attention and (2) word-material

attention. Then it outputs the context vectors  $\hat{U}$  as follows:

$$Z_m = \text{softmax}\left(\frac{\hat{W}^T G^t}{\sqrt{d}}\right), \quad (4.15)$$

$$U_m = Z_m (G^t)^T, \quad (4.16)$$

$$Z_a = \text{softmax}\left(\frac{\hat{W}^T \dot{A}_h}{\sqrt{d}}\right), \quad (4.17)$$

$$U_a = Z_a (\dot{A}_h)^T, \quad (4.18)$$

$$\hat{U} = \text{concat}(\hat{W}, (U_m)^T, (U_a)^T), \quad (4.19)$$

where  $\text{concat}(\cdot)$  indicates a concatenation function of vectors.  $\hat{u}_k$  is forwarded to the linear layer and softmax activation to obtain the word probability across the vocabulary.

### 4.4.3 Loss functions

In addition to the losses described in Chapter 4.3.4, we introduce the following two types of loss functions: (1) visual simulator loss  $L_{vsim}$  and (2) textual attention loss  $L_{tattn}$ . Fig. 4.8 shows an overview of the loss computation for these two losses.

#### Visual simulator loss

This loss aims to train the visual simulator, consisting of two losses: (1) material selection and (2) action selection loss. Given the action- and material-event attention matrices in Eq (4.9) and Eq (4.11), they are computed as the summed negative log-likelihood based on the materials/actions and events that constitute the story.

To avoid costly human annotations, we compute the loss through distant supervision [77] following our previous work. For the material selection loss, labels are obtained on whether the sentence corresponding ground-truth event contains materials and whether the events are oracle or not at each step. For the action selection loss, labels are obtained whether the sentence has actions in the 384 actions defined by [12] and whether the events are oracle or not at each step. For example, in Fig. 4.8, “eggs” at the oracle event index of three are extracted as a material label, and “crack” and “stir” are extracted at the same event index as an action label. Mathematically, let  $\ddot{A}_h, \ddot{G}_h^{t-1}$  be the action- and material-label binary matrices respectively, where each element represents 0 or 1. Based on  $\hat{A}_h, \hat{G}_h^{t-1}, \ddot{A}_h, \ddot{G}_h^{t-1}$ , the material

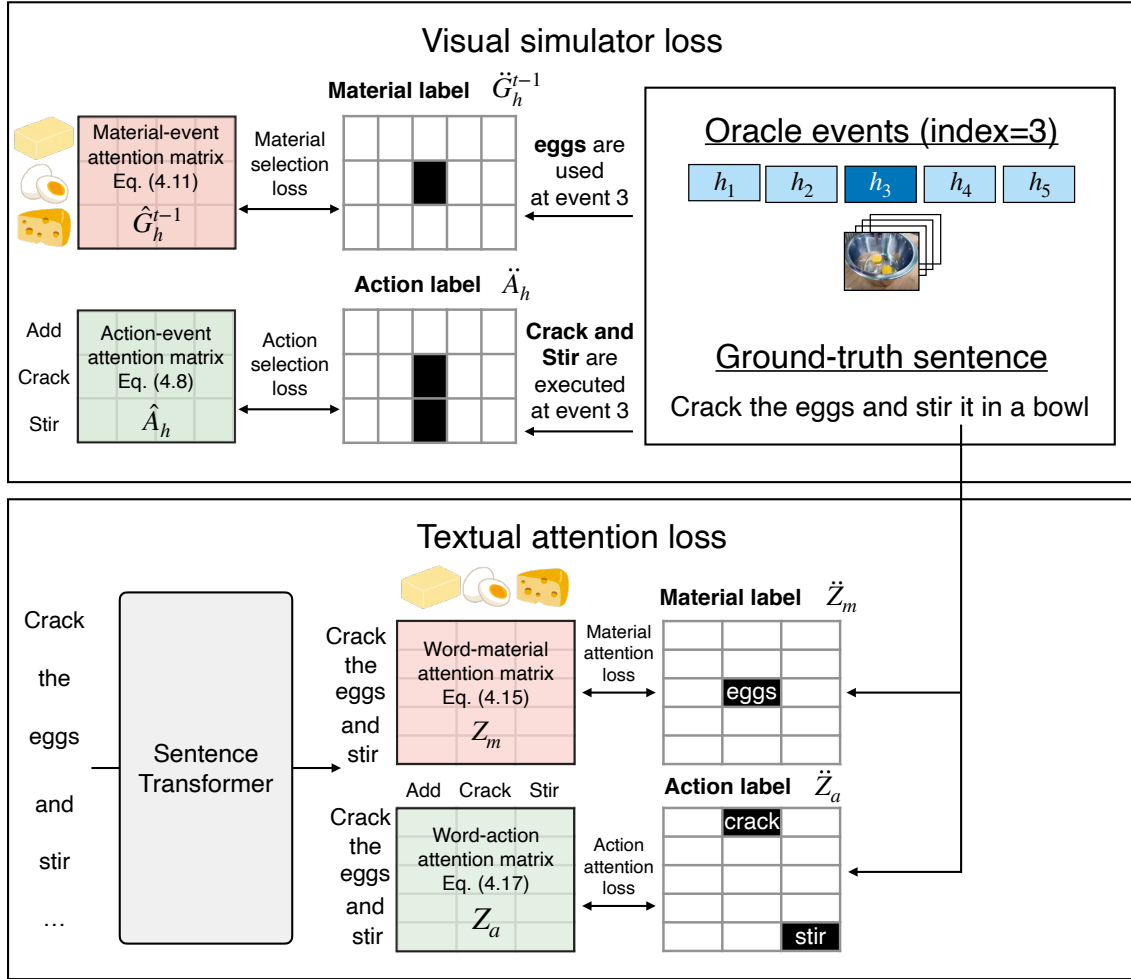


Fig. 4.8 An overview of loss computation of visual selector loss and textual attention loss.

and action selection losses are computed as:

$$L_{is} = -\text{SumMat}(\log(\hat{G}_h^{t-1} \odot \ddot{G}_h^{t-1})), \quad (4.20)$$

$$L_{as} = -\text{SumMat}(\log(\hat{A}_h \odot \ddot{A}_h)), \quad (4.21)$$

$$L_{vsim} = L_{is} + L_{as}, \quad (4.22)$$

where  $\text{SumMat}(\cdot)$  represents the sum of all of the elements of matrix.

### Textual attention loss

This loss aims to train textual attention, consisting of two losses: (1) material attention loss and (2) action attention loss. Given the word-material/action attention matrices, they are computed as the sum of the negative log-likelihood based on whether the  $k$ -th word is the

same as the materials/actions. Mathematically, let  $\check{Z}_a, \check{Z}_m$  be the action- and material-word binary matrices, which represent whether the  $k$ -th word is the same as the actions/materials. Given the word-material/action attention matrices  $Z_m, Z_a$  and  $\check{Z}_a, \check{Z}_m$ , they are computed as:

$$L_{ia} = -\text{SumMat}(\log(Z_m \odot \check{Z}_m)), \quad (4.23)$$

$$L_{aa} = -\text{SumMat}(\log(Z_a \odot \check{Z}_a)), \quad (4.24)$$

$$L_{tattn} = L_{ia} + L_{aa}. \quad (4.25)$$

### Total loss

We simply add these losses to the loss defined in Chapter 4.3.4 as follows:

$$L_{extended} = L_{base} + L_{vsim} + L_{tattn}. \quad (4.26)$$

## 4.5 Experiments

We use the YouCook2-ingredient+ dataset [147], which consist of 1,788 cooking (= instructional) videos, recipes (= procedural text) and ingredients (= materials). We use the official split proposed by [147] for evaluation. Because the test set is not available online, we use the validation set for evaluation.

### Event encoder

We employ different two encoders described below:

- **TSN** [128] converts the appearance and optical flow into frame-level representations, and then outputs the event-level representations by averaging them. For appearance, we use 2,048D feature vectors extracted from the “Flatten-673” layer in ResNet-200 [47]. For the optical flow, 1,024D feature vectors are extracted from the “global pool” layer in BN-Inception [51]. Note that these models are pre-trained on only vision resources (e.g., ImageNet [31]).
- **MIL-NCE** [74] converts the events into representations. The model is pre-trained on Howto100M [75], which consists of automatically constructed 100M clip-narration pairs. We expect the MIL-NCE to yield better event representations than the TSN because this model is pre-trained on instructional vision-and-language resources.

### Data preprocessing

As in [65], we truncated sequences longer than 100 for the event and 20 for the sentence and set the maximum length of the event sequence to 12. Finally, we built the vocabulary based on words that occurred at least three times. The resulting vocabulary contained 991 words.

### Hyper-parameter settings

For both the encoder and decoder transformers, we set the hidden size to 768, the number of layers to two and the number of attention heads to 12. We train the model using the optimization method described in [32, 65]; we use the Adam optimizer [56] with an initial learning rate of 0.0001,  $\beta_1 = 0.9$  and  $\beta_2 = 0.999$ . The L2 weight decay is set to 0.01, and the learning rate warmup is over the first five epochs. We set the batch size to 16, and continue training at most 50 epochs using early stopping with SODA:CIDEr-D.

### Models

We test the proposed method by comparing it with four state-of-the-art dense video captioning models, as described below:

- **Masked Transformer (MT)** [148] is a transformer-based encoder-decoder DVC model that can be trained in an end-to-end manner by using a differentiable mask.
- **Event-centric Hierarchical Representation for DVC (ECHR)** [130] is an event-oriented encoder-decoder architecture for DVC. The ECHR incorporates temporal and semantic relations into the output events for generating captions accurately.
- **SGR** [30] is a top-down DVC model consisting of four processes. The model (1) generates an overall paragraph from the input video, (2) grounds the sentences with events in the video, (3) refines captions based on the grounded events and (4) refines events, referring to the refined captions.
- **PDVC** [129] is the state-of-the-art DVC model. It detects  $N$  events densely, re-ranks the top  $K$  of them and generates sentences for the re-ranked top  $K$  events. Note that  $K$  is the prediction target. This model is used in our preliminary experiments described in Chapter 4.2.

### Ablations

We examine the impact of the components of the proposed method through ablation studies on the following variations:

Table 4.2 Word-overlap metrics for the baseline and proposed method. The bold scores are the best among the comparative methods.

	Input modality	Video feature	dvc_eval			SODA		
			BLEU4	METEOR	CIDEr-D	METEOR	CIDEr-D	tIoU
MT	Video (V)	TSN	0.30	3.18	6.10	-	-	-
ECHR	V	TSN	-	3.82	-	-	-	-
SGR	V	TSN	-	-	-	4.35	-	-
PDVC (reported)	V	TSN	0.80	4.72	22.71	4.42	-	-
PDVC (reproduced)	V	TSN	0.56	5.80	21.47	3.99	15.10	27.80
Base (B)	V	MIL-NCE	1.04	6.03	24.98	5.45	25.09	33.23
Base + Ingredients (BI)	Video + Ingredients (VI)	MIL-NCE	1.39	7.18	31.07	6.44	31.69	<b>35.10</b>
BI + Visual simulator (BIV)	VI	MIL-NCE	1.40	7.27	32.67	6.46	32.95	34.13
BIV + Textual attention (BIVT)	VI	MIL-NCE	<b>1.92</b>	<b>8.04</b>	<b>37.24</b>	<b>7.29</b>	<b>38.93</b>	35.06
Oracle	V	TSN	0.97	7.68	36.30	9.64	35.09	71.16

- **Base model (B)** is the model introduced in Chapter 4.3.
- **B + Ingredient (BI)** incorporates the ingredient encoder into the base model.
- **BI + Visual simulator (BIV)** incorporates the visual simulator into the BI model.
- **BIV + Textual attention (BIVT)** additionally incorporates the textual attention module into the BIV model.

### 4.5.1 Word-overlap evaluation

Table 4.2 demonstrates the results of the word-overlap evaluation on both dvc\_eval and SODA with BLEU, METEOR and CIDEr-D. We observe that the base model consistently outperforms state-of-the-art captioning models by a significant margin in both evaluations. In the ablation, the BIV model outperforms the BI model, and the BIVT model further improves the BIV model. This indicates that both the dot-product visual simulator and the textual attention module are effective for accurate recipe generation.

### 4.5.2 Discussion on the number of predicted events

In addition to the word-overlap metrics, we discuss the generated recipes from the perspective of the number of predicted events. Table 4.3 shows the percentage of recipes that satisfy  $|p - q| \leq \eta$ , where  $p, q, \eta$  represents the number of predicted events, ground truth and a threshold, respectively. In this experiment, we change  $\eta$  from 0 to 3. This result demonstrates that the proposed models consistently predict a more precise number of events than the PDVC. Fig. 4.9 shows the histogram of the number of the predicted events. While the PDVC outputs the specific number of events (i.e., 5, 7, 10), the histogram of the proposed method draws a similar curve to that of the ground truth.

Table 4.3 Percentage of procedural texts that satisfy  $|p - q| \leq \eta$ , where  $p, q, \eta$  represents the number of predicted events, ground-truth events, and a threshold, respectively. In this experiment, we change  $\eta$  from 0 to 3.

$\eta$	0	1	2	3
PDVC	14.4	40.0	63.0	76.4
Model				
B	18.6	<b>52.1</b>	71.7	83.4
BI	18.8	51.6	73.5	87.3
BIV	<b>20.5</b>	51.8	<b>74.2</b>	86.2
BIVT	19.7	51.8	73.5	<b>87.5</b>

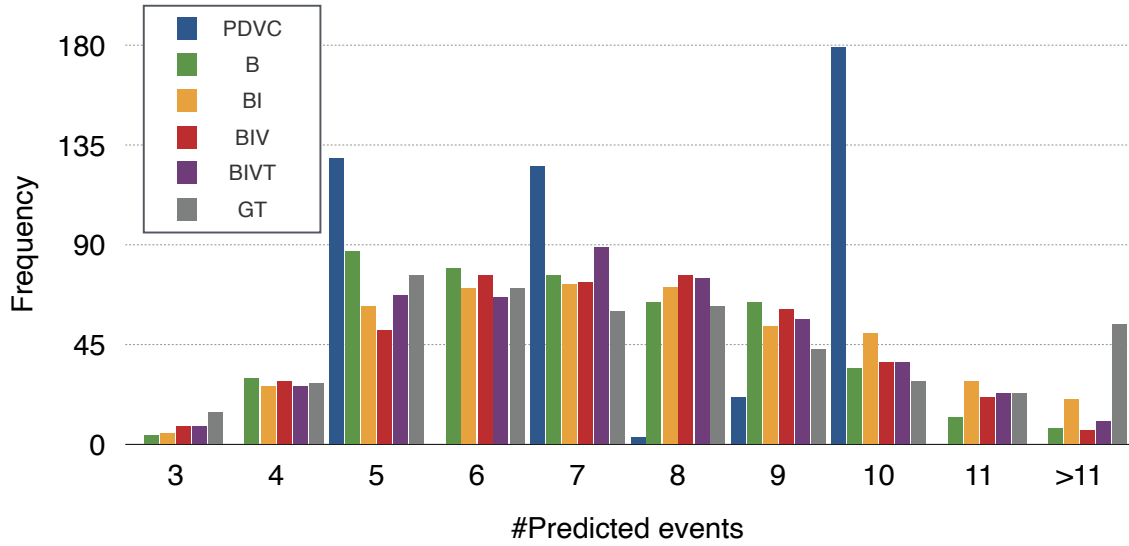


Fig. 4.9 Histogram of the number of predicted events and ground truth.

### 4.5.3 Qualitative analysis

Fig. 4.10 shows the predicted events and generated recipes from the PDVC, B and BIVT models, in comparison to that of the ground truth. In terms of the predicted events, PDVC outputs highly overlapped events. The proposed method, on the other hand, predicts events in the correct order, with minimal overlap. This story-oriented sequential event prediction is an advantage of the proposed method. In terms of the generated recipes, the PDVC repeatedly generates the same contents, ignoring the events (see (c) to (g)). The proposed methods suppress this problem. A comparison of B and BIVT reveals that the BIVT can generate sentences that are grounded with events (e.g., “pork” is accurately verbalized in (a) in BIVT, whereas “fat” is generated in (a) in B).



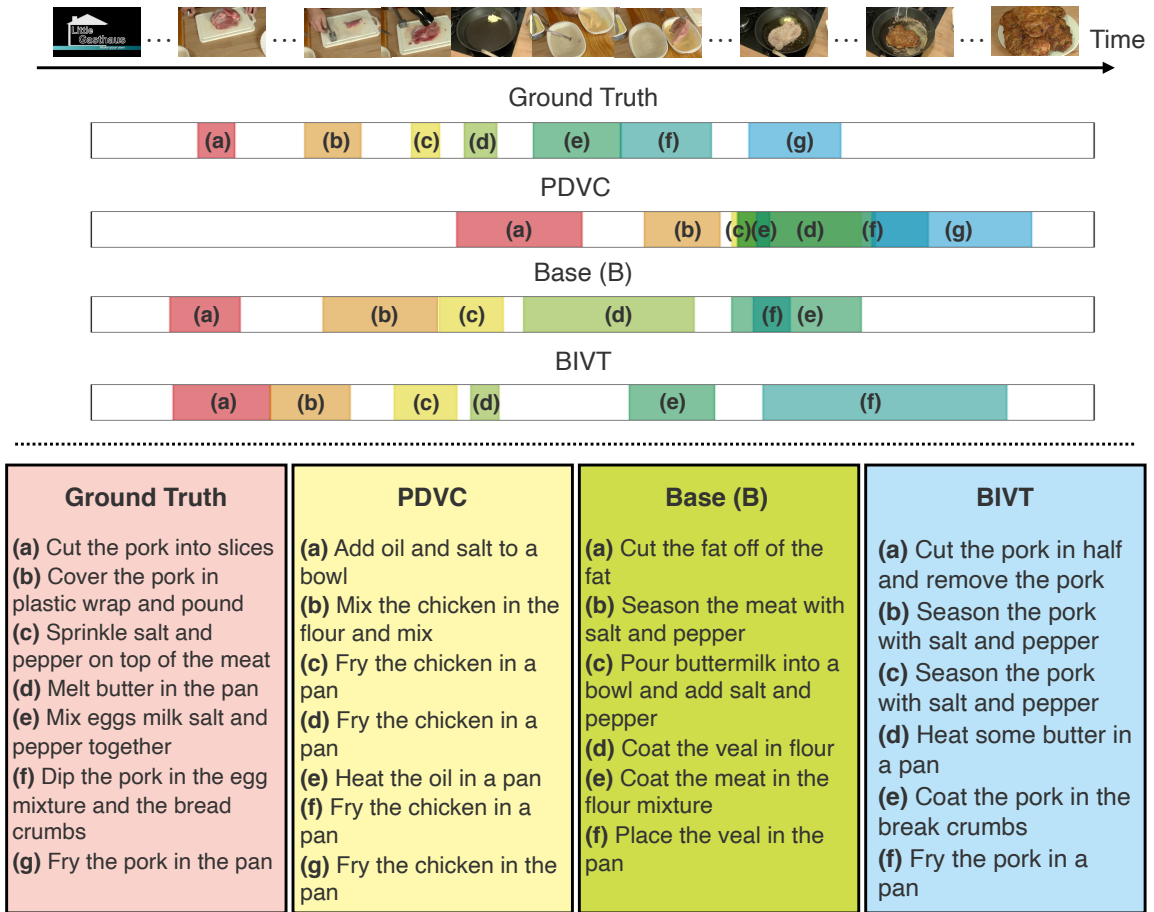


Fig. 4.10 Examples of the generated procedural texts. We compare three models: PDVC, B and BIVT with the ground truth.

#### 4.5.4 Discussion on the detailed model settings

Here, we discuss the detailed model settings from five perspectives: (1) loss ablation studies, (2) memory update strategies, (3) input modalities, (4) event encoders and (5) parameter sensitivity and the event candidates  $N$ . The results demonstrate that these parameters are important to succeed in our task.

##### Loss ablation studies

Table 4.4 shows ablation studies on the loss function for the BIVT model. The experiment yields two insights. First, all the losses are essential for training the model. This is confirmed by removing all of them from the full model (compare (a) and (g)). Second, material-based losses are more necessary than action-based ones. Removing the material selection loss has a more significant influence on the performance, compared to the action selection loss

Table 4.4 Loss ablation studies. MS, AS, MA, and AA represent material selection, action selection, material attention, and action attention losses, respectively.

	MS	AS	MA	AA	dvc_eval			SODA		
					BLEU4	METEOR	CIDEr-D	METEOR	CIDEr-D	tIoU
(a)					1.11	6.37	26.54	5.69	26.54	33.15
(b)	✓				1.28	7.31	31.59	6.46	33.10	33.78
(c)		✓			1.44	7.32	32.19	6.35	30.86	33.35
(d)	✓	✓			1.57	7.45	33.56	6.76	35.45	34.56
(e)	✓	✓	✓		1.76	7.79	36.84	7.12	36.96	34.91
(f)	✓	✓		✓	1.57	7.70	34.58	6.93	37.42	34.53
(g)	✓	✓	✓	✓	<b>1.92</b>	<b>8.04</b>	<b>37.24</b>	<b>7.29</b>	<b>38.93</b>	<b>35.06</b>

Table 4.5 Comparison of memory update strategies: separate vs joint.

	dvc_eval			SODA		
	BLEU4	METEOR	CIDEr-D	METEOR	CIDEr-D	tIoU
<b>B</b>						
Separate	0.92	5.67	24.19	5.19	23.12	<b>33.96</b>
Joint	<b>1.04</b>	<b>6.03</b>	<b>24.98</b>	<b>5.45</b>	<b>25.08</b>	33.26
<b>BIV</b>						
Separate	1.39	7.24	<b>33.02</b>	<b>6.54</b>	<b>33.00</b>	33.81
Joint	<b>1.40</b>	<b>7.27</b>	32.67	6.46	32.95	<b>34.13</b>
<b>BIVT</b>						
Separate	1.78	7.85	36.65	6.92	36.84	33.85
Joint	<b>1.92</b>	<b>8.04</b>	<b>37.24</b>	<b>7.29</b>	<b>38.93</b>	<b>35.06</b>

(compare (b) with (d) and (c) with (d)). This tendency is the same as the relationship between the material and action attention losses.

### Memory update strategies: separate or joint?

Table 4.5 shows a comparison of the memory update strategies. While the separate memory update does not mix the memories in the event and sentence transformers, the joint approach proposed in Chapter 4.3 fuses these memories. The result demonstrates that the joint approach outperforms the separate approach, indicating the effectiveness of the joint memory update strategy.

### Input modalities: video only or multimodal?

MIL-NCE has two branches of encoders: video and text encoders. In our experiments, we use only the video branch but can employ a text encoder by inputting the predicted events with generated captions of PDVC. We compare video only version and the multimodal version;

Table 4.6 Comparison of input modalities: video only and multimodal versions.

	dvc_eval			SODA		
	BLEU4	METEOR	CIDEr-D	METEOR	CIDEr-D	tIoU
<b>B</b>						
Video only	<b>1.04</b>	<b>6.03</b>	<b>24.98</b>	<b>5.45</b>	<b>25.08</b>	<b>33.26</b>
Multimodal	0.44	3.91	15.49	3.51	13.21	27.33
<b>BIV</b>						
Video only	<b>1.40</b>	<b>7.27</b>	<b>32.67</b>	<b>6.46</b>	<b>32.95</b>	<b>34.13</b>
Multimodal	0.60	4.68	19.02	4.03	14.17	27.32
<b>BIVT</b>						
Video only	<b>1.92</b>	<b>8.04</b>	<b>37.24</b>	<b>7.29</b>	<b>38.93</b>	<b>35.06</b>
Multimodal	0.78	5.18	21.00	4.66	19.51	28.94

Table 4.7 Comparison of the model’s performance when varying the event encoders: TSN and MIL-NCE. Note that unlike TSN, which is pre-trained on only vision resources, the MIL-NCE is pre-trained on instructional vision-and-language resource, Howto100M.

	dvc_eval			SODA		
	BLEU4	METEOR	CIDEr-D	METEOR	CIDEr-D	tIoU
<b>B</b>						
TSN	0.36	4.24	15.55	3.79	14.98	31.71
MIL-NCE	<b>1.04</b>	<b>6.03</b>	<b>24.98</b>	<b>5.45</b>	<b>25.08</b>	<b>33.26</b>
<b>BIV</b>						
TSN	0.52	4.93	18.98	4.51	18.32	31.77
MIL-NCE	<b>1.40</b>	<b>7.27</b>	<b>32.67</b>	<b>6.46</b>	<b>32.95</b>	<b>34.13</b>
<b>BIVT</b>						
TSN	0.99	5.87	23.60	5.26	22.84	32.36
MIL-NCE	<b>1.92</b>	<b>8.04</b>	<b>37.24</b>	<b>7.29</b>	<b>38.93</b>	<b>35.06</b>

in the multimodal version, we convert event timestamps and sentences into vectors using MIL-NCE branches, simply concatenate them and input them into the models.

Table 4.6 shows a comparison of the input modalities, indicating that the video-only inputs achieve much better than the multimodal inputs. This occurs because the sentences generated by PDVC are semantically overlapped (see Fig. 4.10) and do not contribute to our recipe generation task.

### Event encoders

Table 4.7 shows the performance difference when changing the event encoders, indicating that the MIL-NCE proved significantly superior to the TSN in all of the settings. This occurs because the MIL-NCE is pre-trained on the vision-and-language resource, Howto100M,

Table 4.8 Comparison of the model’s performance when varying the number of the event candidates  $N$ .

	dvc_eval			SODA		
	BLEU4	METEOR	CIDEr-D	METEOR	CIDEr-D	tIoU
<b>B</b>						
N=25	<b>1.27</b>	<b>6.49</b>	<b>27.84</b>	<b>6.19</b>	<b>29.34</b>	<b>35.26</b>
N=50	0.98	6.42	27.12	5.89	26.34	33.79
N=100	1.04	6.03	24.98	5.45	25.08	33.26
N=200	0.93	6.16	25.93	5.52	26.13	33.93
<b>BIV</b>						
N=25	<b>1.71</b>	7.50	<b>34.18</b>	<b>7.02</b>	<b>36.39</b>	<b>36.13</b>
N=50	1.51	<b>7.52</b>	33.37	6.86	34.24	35.60
N=100	1.40	7.27	32.67	6.46	32.95	34.13
N=200	1.23	6.81	29.51	5.95	28.86	33.15
<b>BIVT</b>						
N=25	1.87	7.95	36.12	<b>7.51</b>	<b>39.06</b>	<b>36.74</b>
N=50	1.81	7.99	36.48	7.36	38.49	36.38
N=100	<b>1.92</b>	<b>8.04</b>	<b>37.24</b>	7.29	38.93	35.06
N=200	1.78	7.66	35.21	6.77	35.64	33.48

which captures the fine-grained event-level semantics of cooking procedures. We conclude that pre-training on an appropriate resource is essential to effective performance on our task.

### Parameter sensitivity and the event candidates $N$

The PDVC allows users to select the number of candidates  $N$  when training the model. In the original study on the PDVC,  $N$  is set to be 100 on the YouCook2 dataset. In this experiment, we change the parameter  $N$  to be 25, 50, 100 and 200 to investigate the parameter sensitivity of the model. Table 4.8 shows the performance change of the proposed method in different model settings: B, BIV and BIVT. The results demonstrate that the proposed method consistently performs well on  $N = 25$ . We observe that increasing  $N$  degrades the model performance. Although a higher  $N$  makes the maximum tIoU larger (shown in Chapter 4.2), the ratio of events that are not oracle but highly overlapped with the ground-truth increases. This prevents the model from selecting oracle events precisely and causes it to overfit the training set.

## 4.6 Conclusion

In this paper, we tackled procedural text generation from unsegmented instructional videos, a task that requires agents to (1) extract key events that are essential to dish completion and

(2) generate sentences for the extracted events. We first analyzed the state-of-the-art DVC models and set our goal to obtain correct procedural text by selecting oracle events from the output events of the DVC model and re-generating sentences for them.

To achieve this, we proposed a transformer-based multimodal recurrent learning model, which consists of the event selector and sentence generator. The event selector selects oracle events from the event candidate in the correct order, and the sentence generator generates procedural text grounded in the events. Both modules have memory vectors to remember the history of previous predictions to estimate the next step. The proposed memory mixing approach efficiently combines them, effectively sharing the previous predictions between the event selector and sentence generator. To generate more accurate procedural text, we also proposed an extended model by introducing two additional modules: dot-product visual simulator and textual attention module.

In the experiments, we tested the methods in the cooking domain and confirm that the base model outperforms the state-of-the-art DVC models and the extended model boosts the model's performance. In addition, we showed that the proposed models can select the correct number of events, as with the ground-truth events. The qualitative evaluation revealed that the proposed approaches can select events in the correct order and generate procedural text grounded in the video content. Finally, we discussed the detailed experimental settings for optimal procedural text generation.



# Chapter 5

## BioVL2: Egocentric Biochemical Video-and-Language Dataset

### 5.1 Introduction

In this chapter, we aim to extend our research focus from everyday domains to important domains in terms of practical applications. We selected the biochemical domain because of the need to generate protocols in terms of the reproducibility of experiments. As reported in [7], the wet-lab research, such as biochemistry and life science, faces a reproducibility crisis because 75% to 80% researchers have failed to reproduce another scientist's experiments in this field.

One of the reasons for this issue is the inadequacy of the protocols. In wet-lab research, the protocols (Fig. 5.1) play an important role in ensuring reproducibility. They should describe not only actions and objects but also other essential information for reproduction, such as the number of objects, time and object states, if necessary. For example, in Fig. 5.1, “Thoroughly resuspend pellet with 250  $\mu$ L of Cell Resuspension Solution” contains an action “resuspend” with two target objects (“pellet” and “cell resuspension solution”) with their quantity “250 $\mu$ L” and adverb “thoroughly.” Although the ideal protocols enable the researchers to reproduce experiments completely, the real ones are sometimes insufficient due to missing manipulations and unclear descriptions that prevent the researchers from following the protocols precisely.

The integrated processing of vision and language can be the solutions to this problem. For example, the video-and-text alignment can automatically create multimedia protocols, encouraging the researchers to understand the content of the experiments intuitively. The video captioning technology reduces the burden of writing the protocols by providing the

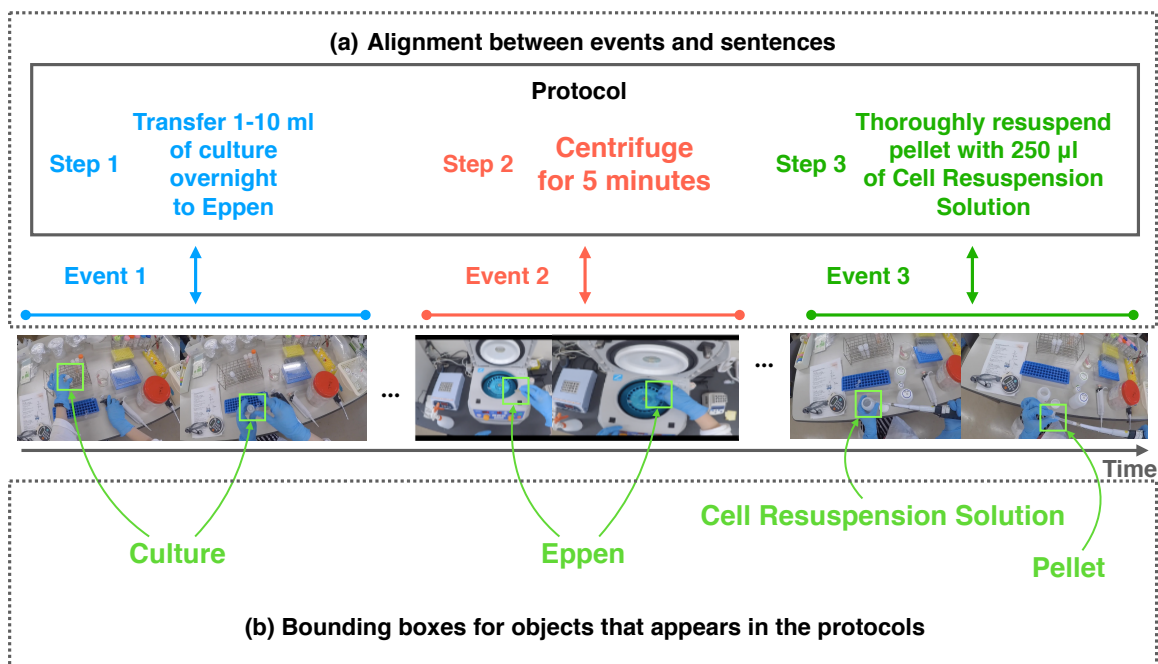


Fig. 5.1 An overview of the BioVL2 dataset, which consists of two types of video-and-language annotations: (a) alignment between events and sentences and (b) bounding boxes for objects that appear in the protocols.

researchers with a draft, leading to a decrease in the protocol writing problems mentioned above.

Despite such potential promising applications, few researchers have tackled video-and-language research in the wet-lab domain [81, 82]. One of the difficulties is constructing and releasing the benchmark dataset online. To address this issue, we first propose an egocentric biochemical video-and-language dataset, called BioVL2 (Fig. 5.1), which consists of experiment videos, protocols and two types of annotations to connect videos with text protocols:

1. **Event-and-sentence alignment.** We divide protocols into sentences by actions and align them with events in the video. We follow the annotation formats to the existing video-and-language research [147, 57] and assume that this annotation can be used for video captioning [136, 84] and video-text alignment [81, 82].
2. **Bounding boxes for objects in the protocols.** We attach bounding boxes with objects that appear in the protocols, which enables researchers to analyze spatial information of objects, such as object classes and states. This annotation enables the researchers to try tracking spatiotemporal object changes [146] from the videos.





Fig. 5.2 “まほろ,” a robot conducting biomedical experiments developed by National Institute of Advanced Industrial Science and Technology. The proto is quoted from [https://www.aist.go.jp/sst/ja/exhibition/innovation\\_zone/zone11/index.html](https://www.aist.go.jp/sst/ja/exhibition/innovation_zone/zone11/index.html) (accessed 5/21, 2023).

Not limited to the above research directions, this project has a wide range of potential applications in the future. For example, one can use the dataset for challenging interdisciplinary research between CV, NLP and robotics, such as training robots to conduct experiments from protocols and videos. “まほろ,” a robot conducting biomedical experiments, is a good example for applications of our resources (Fig. 5.2). We believe that the BioVL2 dataset is the first step toward the goals.

As discussed in Chapter 2.3.3, constructing an egocentric dataset poses scalability challenges. To overcome this issue, we adopted a user-oriented design for data collection. Our approach involved envisioning biochemical researchers capturing videos of their experiments and sharing them with other researchers for future replication. However, expecting researchers to configure camera settings and equipment for recording experiments would be a burden, and such an approach would not scale up the dataset. To address this challenge, we leveraged unedited first-person videos as the primary visual source, inspired by [27]. Unlike third-person cameras that require researchers to set up multiple cameras and synchronize them, first-person cameras are easy to use and cover a wide range of the experiment space. Consequently, we collected 32 videos from four protocols, with a total length of 2.5 hours.

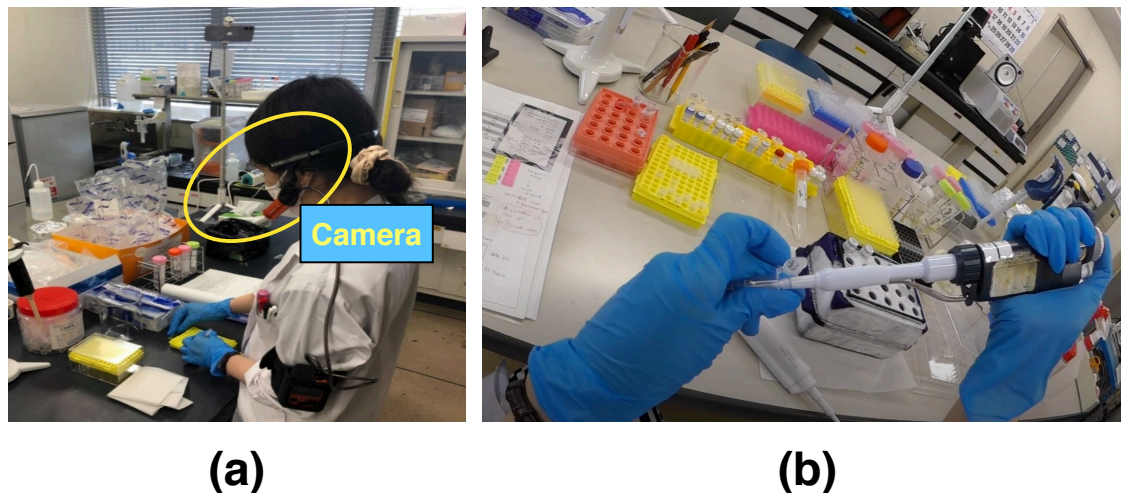


Fig. 5.3 (a) and (b) shows the recording studio of experiments and the view from the equipped first-person camera, respectively.

Based on the constructed BioVL2 dataset, we tackle the task of generating protocols from experiment videos. Because the dataset size is quite smaller than the general benchmarks on video captioning [57, 147, 136], it is infeasible to train a deep model in an end-to-end manner. Therefore, we employ the model proposed by [119], which is designed to be applicable to small datasets by utilizing external resources on the web. Our experimental evaluation demonstrates that the model can generate more accurate protocols than a weak baseline, which outputs objects appearing in the videos. In addition, we analyze the model’s behavior and limitations in detail and discuss the future directions of the BioVL project.

## 5.2 BioVL2 dataset

In this section, we describe the detail of the BioVL2 dataset. We first explain how to construct the dataset and then report the statistics and annotation agreement.

### 5.2.1 Dataset construction

#### Video recording

**Participant.** We asked one researcher (1 female) for our video collection. During experiments, the researcher put on a headset that fixes a wearable first-person camera (Fig. 5.3).

Table 5.1 An example of the annotation for PCR. The values in the table represent seconds. Note that @ is written in the table, but not included in the real dataset.

Sentence	Start	End
add sterile distilled water	30	45
add primer1	64	99
add primer2	106	130
add template	149	173
add primeSTAR@Max Premix	190	238
set in DNA engine	260	266

Note that the headset is light enough for researchers to concentrate on their experiments <sup>1</sup>. We asked her to conduct experiments as usual.

**Experiment target.** We choose the basic well-known four experiments that have well-established protocols in the biochemical domain: miniprep, PCR, DNA extraction and agarose gel creation. We took eight videos per experiment, collecting 32 videos in total <sup>2</sup>. Only DNA extraction has two different methods: Phenol-chloroform extraction and Ethanol precipitation. We took videos four times for each method.

**Data preprocessing.** In a few experiments, the researcher needed to wait while executing specific instructions (e.g., centrifuge samples). During the waiting time, the researcher put off the headset, leaving the camera on. We manually trimmed such waiting times because they are not related to any instructions.

### Event-and-sentence alignment annotation

Based on the collected videos, we annotated the alignment annotation between events and sentences. The protocols are graphically described using figures or illustrations and are not well-organized as text documents. Thus, we asked the researcher to explain the content of the experiments and transcribed it as text protocols. Then, we divide the protocol into sentences by actions and obtain a sequence of instructions. For example, “Invert 4 times to mix and add 10  $\mu$ l of Alkaline Protease Solution.” can be divided into “Invert 4 times to mix” and “Add 10  $\mu$ l of Alkaline Protease Solution.” Here, we remove “and” from the text. Finally, the annotator watched the videos and annotated events in the start and end timestamp format for each sentence. In this process, to remove the ambiguity of event timestamps, we annotated the period from the starting time when the researcher touches the target objects to the ending time when she releases them after her manipulation. Table 5.1 shows the annotation example of PCR.

<sup>1</sup>We use Panasonic HX-A500 for recording experiments.

<sup>2</sup>We took multiple videos for the same experiment because of the large variety of actions.

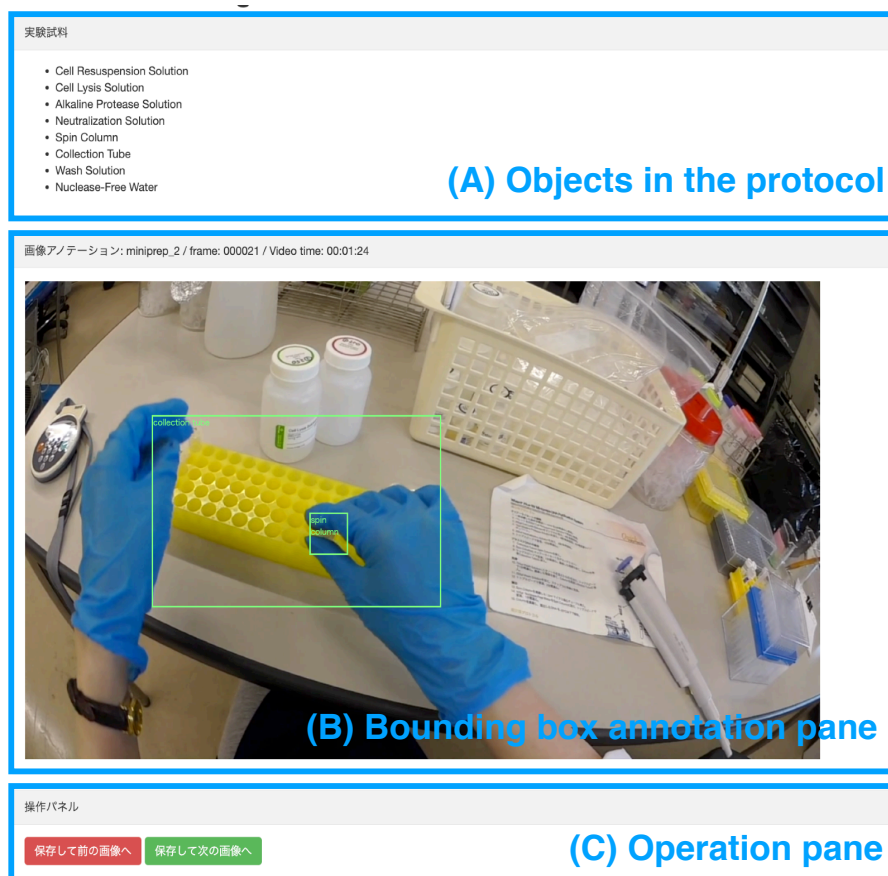


Fig. 5.4 A screen of the bounding box web annotation tool. (A), (B) and (C) shows the objects in the protocols, annotation bounding box pane and operation pane, respectively.

### Bounding boxes for objects in the protocols

In addition to the alignment annotations, we annotate bounding boxes for objects in the protocols. We extracted frames per 4 seconds (4 fps) and annotate bounding boxes with objects if (1) her hands touch them and (2) they appear in the protocols (Fig. 5.4). Note that objects in the protocols are categorized into object-based bio-NEs (b-NEs) defined in the WLP dataset [60] and they are manually extracted from the protocols. We also emphasize that our annotation targets are all of the video frames, rather than only events. This is because annotating all video frames is helpful for a wide range of applications, including protocol generation from experiment videos.

Table 5.2 Statistics of text annotations. The average and standard deviation (std) are written in the table. Note that in miniprep and agarose gel creation, several steps are skipped depending on the situation. Thus, note that their standard deviations are not equal to 0.

		#steps	#words/#steps	#objects	#verbs
DNA extraction	Phenol chloroform	4.0 ( $\pm 0.0$ )	6.0 ( $\pm 1.9$ )	2.0 ( $\pm 0.0$ )	4.0 ( $\pm 0.0$ )
	Ethanol	9.0 ( $\pm 0.0$ )	4.9 ( $\pm 2.9$ )	4.5 ( $\pm 1.7$ )	4.3 ( $\pm 0.5$ )
PCR		6.0 ( $\pm 0.0$ )	3.0 ( $\pm 1.0$ )	6.0 ( $\pm 0.0$ )	2.0 ( $\pm 0.0$ )
Agarose gel creation		10.3 ( $\pm 0.4$ )	4.7 ( $\pm 2.4$ )	5.5 ( $\pm 0.5$ )	7.0 ( $\pm 0.0$ )
Miniprep		28.2 ( $\pm 0.4$ )	6.4 ( $\pm 2.5$ )	7.9 ( $\pm 1.4$ )	8.1 ( $\pm 1.2$ )

Table 5.3 The number of unique objects and verbs, which do not appear in the other kinds of protocols.

		#Unique objects	#Unique verbs
DNA extraction	Phenol chloroform	2.0 ( $\pm 0.0$ )	0.0 ( $\pm 0.0$ )
	Ethanol	4.5 ( $\pm 1.7$ )	0.3 ( $\pm 0.5$ )
PCR		6.0 ( $\pm 0.0$ )	0.0 ( $\pm 0.0$ )
Agarose gel creation		5.5 ( $\pm 0.5$ )	4.0 ( $\pm 0.0$ )
Miniprep		7.9 ( $\pm 1.4$ )	3.0 ( $\pm 1.1$ )

## 5.2.2 Statistics

Next, we describe the statistics on the BioVL2 dataset from the perspective of textual and visual sides. Our analysis demonstrates that the BioVL2 dataset contains a large variety of experiments from both sides.

### Protocol

Table 5.2 shows the textual statistics, indicating that the number of words attached to sentences has a big gap between experiments. This reveals the variety of instructions in the BioVL2 dataset; the largest one is miniprep and the smallest is phenol-chloroform. The number of kinds of objects has the same tendency as this, but the smallest kinds of actions are PCR. To verify the diversity of objects and actions, we investigate the number of unique actions and objects that do not appear in other kinds of protocols (Table 5.3). A comparison of the right two columns in Table 5.2 and Table 5.3 shows that the objects' kinds are equal but the action kinds are substantially small. This reveals that the verbs are similar, but the objects are unique in the dataset, indicating the diversity of the objects.

Table 5.4 Statistics of video length. The table values show mean and standard deviations.

		Video length(second)
DNA extraction	Phenol chloroform	269.4 ( $\pm 58.6$ )
	Ethanol	399.4 ( $\pm 19.2$ )
PCR		254.6 ( $\pm 18.1$ )
Agarose gel creation		312.6 ( $\pm 64.5$ )
Miniprep		382.1 ( $\pm 69.9$ )

Table 5.5 Statistics of frames that has bounding box annotations.

		#Annotated frames	#All frames	Ratio
DNA extraction	Phenol chloroform	195	263	74.1
	Ethanol	265	394	67.3
PCR		322	498	64.7
Agarose gel creation		282	614	45.9
Miniprep		245	752	32.6

## Video

Table 5.4 and Fig. 5.5 show the statistics on video and event length, respectively. The results indicate that the BioVL2 dataset has a large variety of video lengths; the maximum video length is Ethanol precipitation (399 seconds) and the shortest is PCR (254 seconds). When focusing on the event-wise length, the miniprep consists of 72.4% of short events that have less than 10 seconds, while the phenol-chloroform consists of 50.0% of long events that have more than 30 seconds.

## Bounding box annotation

Table 5.5 and Table 5.7 describe statistics on the number of frames that have at least one bounding box. Table 5.6 shows statistics on unique objects associated with bounding boxes after classifying them into b-NE tag categories. Note that “Seal” does not appear although other b-NEs, such as “Reagent”, “Location” and “Device” appear in the BioVL2 dataset. To this end, we count the number of Reagents, Locations and Devices in the table. This result indicates that while the most frequent object is a Reagent, the least frequent one is a Device. From the result of Table 5.5, the number of annotated frames is different between experiments. The highest ratio of annotated frames is Phenol chloroform and the lowest one is miniprep.

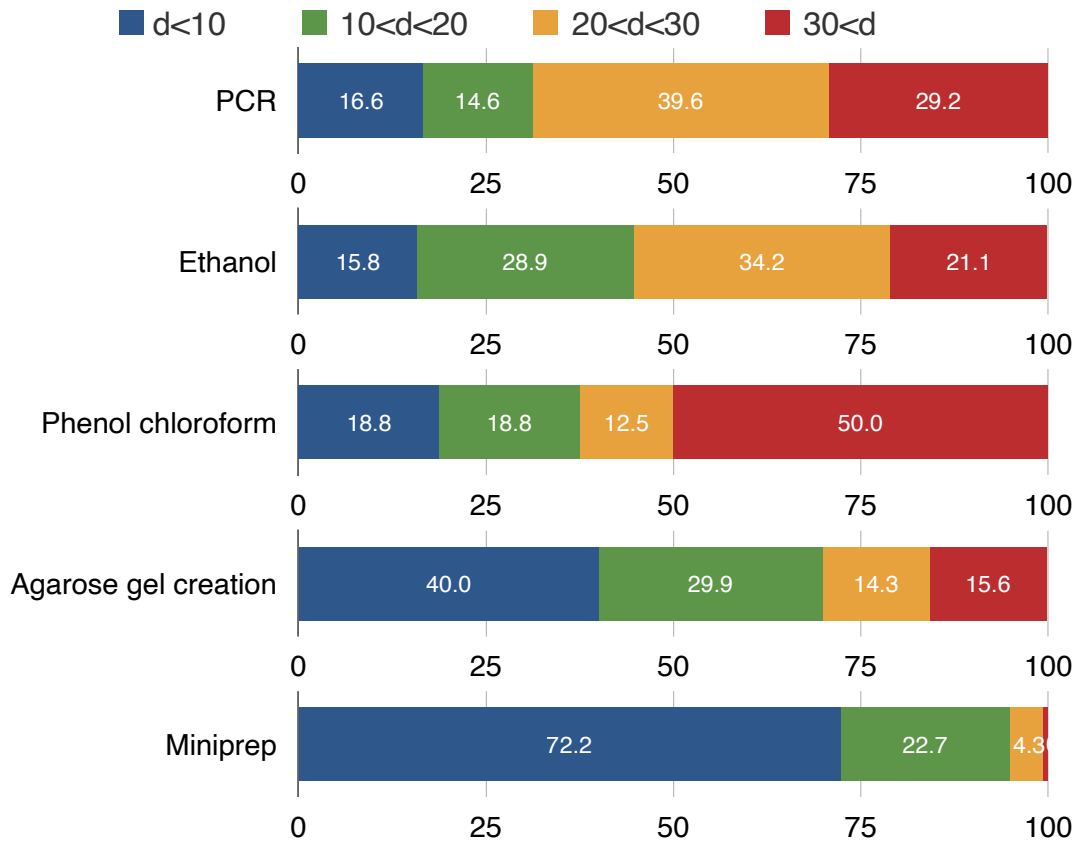


Fig. 5.5 Distribution of event length  $d$  ( $d$  represents seconds).

### 5.2.3 Annotation agreement

To verify the quality of the annotations, we asked other annotators to attach alignment and bounding boxes and computed the annotation agreement. All instructions to the annotators are the same as those described in Chapter 5.2.1. Because annotating all of the videos is costly, we randomly choose one video per experiment for the re-annotation process. The following subsequent sections report the annotation agreement.

#### Alignment annotation between events and sentences

We asked another researcher to annotate alignments between events and sentences and computed agreement scores by comparing them with the results in Chapter 5.2.1. Here, we use tIoU as agreement metrics for computing the overlap of the event annotations. Table 5.8 shows an agreement ratio, indicating that tIoU scores are over 75% on average in all experiments. In existing text-based event retrieval tasks [66], recall is computed by regarding

Table 5.6 Statistics on unique objects associated with bounding box annotations after classifying them into b-NE tag category types. Although among b-NE tags, object-based NEs are Reagent, Location, Device, and Seal, Seal does not appear in the BioVL2 dataset. Thus, we count Reagent, Location, Device in the table.

		#Reagent	#Location	#Device
DNA extraction	Phenol chloroform	1	1	0
	Ethanol	6	0	0
PCR		5	0	1
Agarose gel creation		2	3	1
Miniprep		11	4	0
Total		25	8	2

Table 5.7 Statistics on bounding box annotations.

		Total	Mean	Max
DNA extraction	Phenol chloroform	68	1.0	1
	Ethanol	130	1.0	2
PCR		176	1.0	1
Agarose gel creation		372	1.1	2
Miniprep		627	1.2	3

the events as correct if tIoU is over 70%, thus 75% tIoU scores are enough high end ensure the annotation quality.

### Bounding box annotations for objects that appear in the protocols

We also compute the agreement ratio of bounding boxes by comparing the results in Chapter 5.2.1 with the newly-annotated ones. As the evaluation metrics, we use mean Average Precision (mAP) that is commonly used in object detection tasks [98]. We regard the newly-annotated ones as prediction and the annotation results in Chapter 5.2.1 as the ground-truth labels and computed the mAP scores<sup>3</sup>.

Fig. 5.6 shows the mAP scores and the distribution of true positives and false positives. mAP is 72.73%, ensuring the high quality of annotations by considering that mAP scores of the state-of-the-art practical object detection models range from 0.6 to 0.7. On the other hand, we observe that there exist objects with low AP, such as the lowest one of new\_tube. The main reason for the low AP is that the new annotator attaches annotations on all of the tubes in the videos while the original annotator annotates tubes that appear after an intermediate instruction. We need to consider such instance-level annotations, which is the future work of this study.

<sup>3</sup>We use <https://github.com/Cartucho/mAP> (accessed 2022/11/10)



Table 5.8 Agreement of event-and-sentence alignments. The mean of tIoU is shown in the table.

		tIoU
DNA extraction	Phenol chloroform	88.9
	Ethanol	91.7
PCR		99.4
Agarose gel creation		82.2
Miniprep		76.0

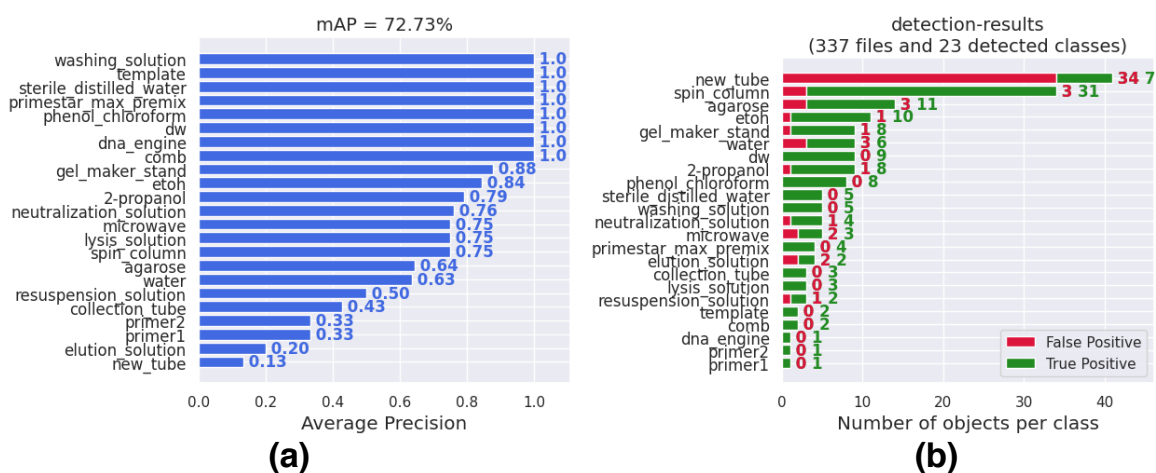


Fig. 5.6 (a) and (b) shows AP and mean of APs (mAP) and the number of true positives and false positives, respectively.

### 5.3 Protocol generation from experiment videos

Based on the constructed BioVL2 dataset, we tackle protocol generation from experiment videos. Because of the limited dataset size, it is infeasible to train end-to-end video captioning models proposed in the literature [57, 148]. To this end, we use the model proposed by [119], which focuses on generating procedural text from a small number of instructional videos by utilizing external resources. Specifically, it targets the cooking domain and generates recipes by following the five steps: (1) detecting objects using the Faster-RCNN [98] for each frame, (2) extracting recipe-Named entities (r-NEs) from the detected objects, (3) constructing r-NE sequence by concatenating the subsequent frame information, (4) generating candidate sentences by using the language model pre-trained on r-NE and sentence pairs and (5) outputting a recipe by exploring the generated candidate sentences based on the Viterbi algorithm [125]. The characteristic of this approach is unnecessary for large-scale video-text pairs because it does not use other visual information except for the objects. The pairs of

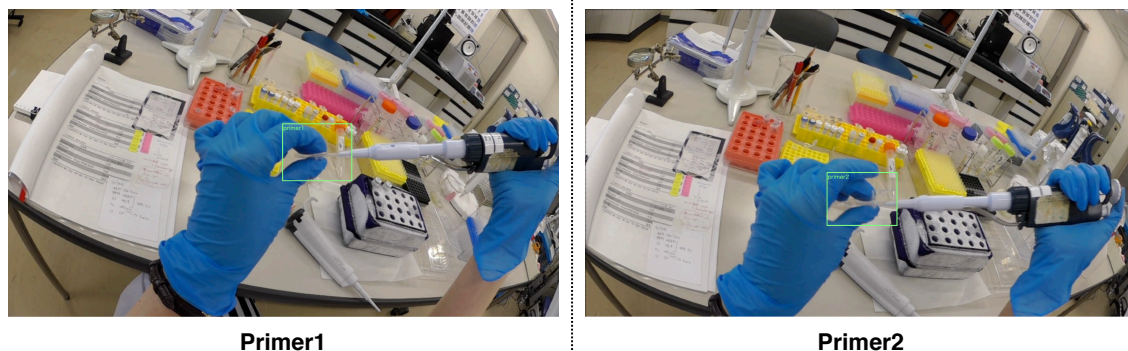


Fig. 5.7 A comparison of frames between manipulating Primer1 and Primer2.

r-NEs and sentences can be acquired by crawling recipes on the web, enabling to pre-train a language model, which leads to generating recipes without video-text pairs.

To apply this approach to the BioVL2 dataset, we modify the model on the following two points:

- Instead of (1) and (2), we directly use the annotated bounding boxes in the downstream processes. Although we can train the object detection models in the cooking domain [46], it is hard to train them in the biochemical domain because the different objects have a similar appearance. We take an example in Fig. 5.7, where Primer1 and Primer2 appear in (a) and (b) frames respectively. Even humans cannot distinguish them from only single frames though they can recognize them properly after watching the whole video. Therefore, we tackle the task of generating a protocol based on the annotated bounding boxes. This task assumes that objects can be acquired automatically by using QR codes in the future.
- We use the pre-trained language model as a transformer, rather than LSTM. The transformer achieves better performance on machine translation, document summarization [69] and image captioning [24] than LSTM, and is expected to be effective for our task. In addition, we incorporate the copy mechanism [103] into the model, which encourages the decoder to generate the input object information. To pre-train the transformer, we use the WLP dataset.

Fig. 5.8 shows an overview of this approach with the above two modifications. Let frames that have at least one bounding boxes be  $F = (f_1, f_2, \dots, f_n, \dots, f_{|F|})$ . Based on the annotated objects in the subsequent frames, we construct the b-NE sequences, such as DNA engine and (DNA engine, Template) in Fig. 5.8. Based on the constructed b-NE sequence, the pre-trained language generation model generates protocol candidate sentences for each of

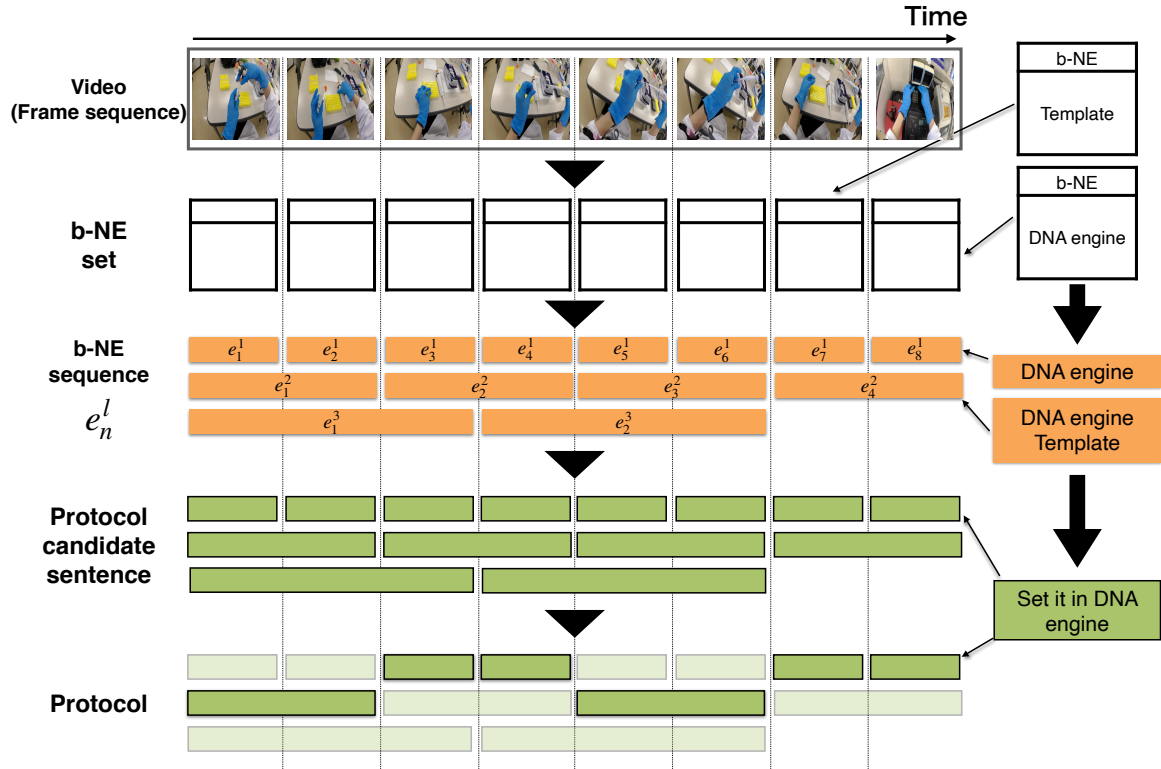


Fig. 5.8 An overview of the protocol generation from experiment videos.

b-NE (“Set it in DNA engine” in Fig. 5.8). Finally, the system outputs protocols by exploring the most plausible combination from the protocol candidate sentences.

### 5.3.1 b-NE sequence construction

Based on the subsequent frames  $f_n^{n+(l-1)}$  ( $l$  is a subsequent length), we follow process (3) to acquire the b-NE sequence. Let the  $n$ -th frame  $f_n$  has the b-NE set  $\mathcal{E}_n$ . Here, we formulate the b-NE set  $\mathcal{E}_{n+(l-1)}$  of the subsequent frames  $f_n^{n+(l-1)}$  as  $\mathbf{e}_n^l \in \mathcal{E}_n \times \mathcal{E}_{n+1} \times \dots \times \mathcal{E}_{n+(l-1)}$  ( $\times$  is the Cartesian product). For example, in Fig. 5.8, the  $l = 1$  b-NE set contains only the DNA engine and the  $l = 2$  b-NE set consists of (DNA engine, Template). We construct the b-NE sequence for  $l = 1, 2$ , and 3 as with the original paper.

### 5.3.2 Generating protocol candidate sentence from b-NE sequence

#### Pre-training the language model

Based on the b-NE sequence  $\mathbf{e}$ , the model generates protocol candidate sentences. Fig. 5.9 shows an overview of the transformer-based language model pre-trained on the WLP

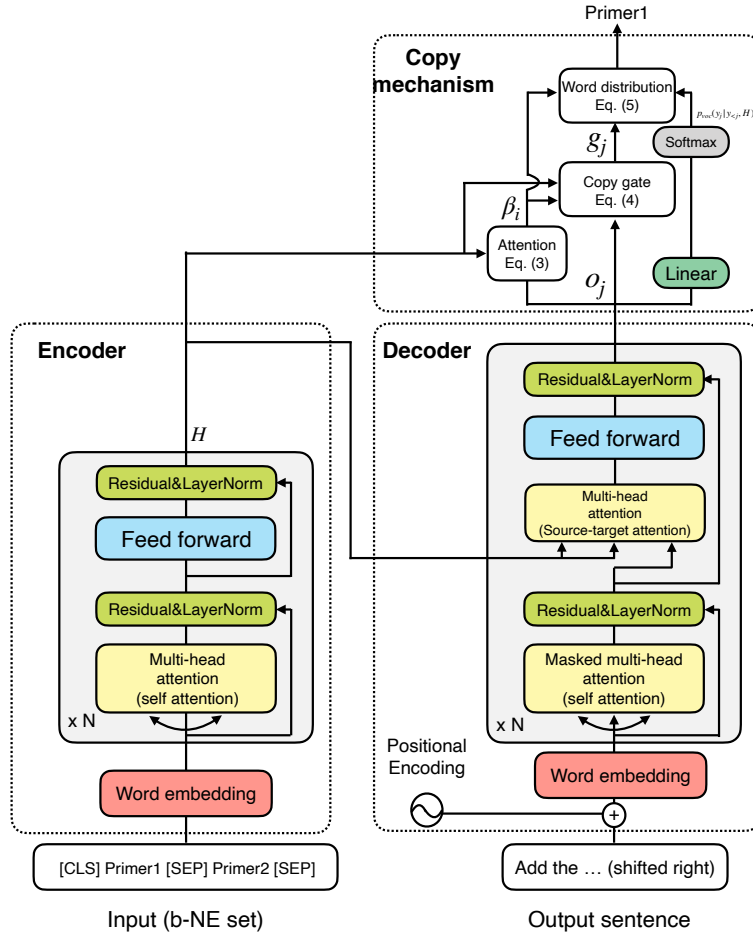


Fig. 5.9 Transformer-based sentence generation model. Given objects, the model is trained to generate corresponding sentences.

dataset. The model is designed as the auto-regressive encoder-decoder architecture with the copy mechanism. Given the input of the b-NE sequence  $X = (x_1, x_2, \dots, x_i, \dots, x_{|X|})$  ( $x_i$  is  $i$ -th word) that has the special head and separate token of [CLS] and [SEP], the model generates a sentence  $Y = (y_1, y_2, \dots, y_j, \dots, y_{|Y|})$  ( $y_j$  is  $j$ -th word). we formulate this task mathematically. Let  $E(\cdot)$  be the transformer encoder and  $D(\cdot)$  be the transformer decoder. Given the output representations  $H = (h_1, h_2, \dots, h_i, \dots, h_{|X|})$  of the encoder, the output vector  $o_j$  corresponding to  $j$ -th word is computed as:

$$H = E(X), \quad (5.1)$$

$$o_j = D(Y_{<j}, H), \quad (5.2)$$

where  $Y_{<j}$  represents the subsequent words of less than  $j$ . As shown in Fig. 5.9, the encoder outputs are injected into the source-target attention layer in the decoder. Note that the word embedding layer in the encoder and decoder is optimized without sharing the parameters and positional encoding is added only to the decoder side, not to the encoder side because the inputs are the set representations.

Because the input objects must be included in the output sentences in our task setting, modeling this explicitly encourages the decoder to generate accurate sentences. To achieve this, we introduce the copy mechanism into the model. Based on the encoder and decoder output  $H$  and  $o_j$ , the attention probability is computed as  $\beta_j^i$ :

$$\beta_j^i = \frac{\exp\{(o_j)^\top W_c h_i\}}{\sum_k \exp\{(o_j)^\top W_c h_k\}}, \quad (5.3)$$

where  $W_c$  represents a linear layer. Then, the copy gate  $g_j$  ( $0 \leq g_j \leq 1$ ) that selects words from the vocabulary or b-NE set is computed as:

$$g_j = \sigma(W_g[o_j; \sum_m \beta_j^m h_m] + b_g), \quad (5.4)$$

where  $[\cdot]$ ,  $\sigma(\cdot)$ ,  $W_g$ ,  $b_g$  represent the concatenation function, sigmoid function, linear weights and bias. Based on the  $g_j$ , the word probability distribution  $p(y_j|y_{<j}, H)$  is computed as:

$$p(y_j|y_{<j}, H) = (1 - g_j)p_{voc}(y_j|y_{<j}, H) + g_j \sum_{i:x_i=y_j} \beta_j^i, \quad (5.5)$$

where  $p_{voc}(y_j|y_{<j}, H)$  represents the probability for the word  $y_j$  in the vocabulary. Given the input/output pairs  $(X, Y) \in \mathcal{D}$  ( $\mathcal{D}$  represents the training dataset), the following negative-log likelihood is minimized to train the model:

$$\mathcal{L}(\theta) = - \sum_{(X,Y) \in \mathcal{D}} \log p(Y|X; \theta), \quad (5.6)$$

where  $\theta$  represents the trainable parameters in the model.

### Score computation

Based on the language model pre-trained on the WLP dataset, we acquire the protocol candidate sentences based on the b-NE sequence described in Chapter 5.3.1. In this process, we compute the likelihood scores for each of them. Specifically, given the b-NE subsequence

$\mathbf{e}_n^l$ , the scores are computed as:

$$\text{Score}(\mathbf{e}_n^l) = \prod_{i=1}^N p(d_i | d_1, d_2, \dots, d_{i-1}; \mathbf{e}_n^l), \quad (5.7)$$

where  $d_i$  represents  $i$ -th word in the output sentence and  $N$  represents the length of the word sequence.

### 5.3.3 Protocol generation

Based on the protocol candidate sentences, we acquire the sequence that has the maximum plausible scores by exploring the computed scores using the Viterbi algorithm. Here we introduce two types of heuristics. Since it is almost impossible for one researcher to perform two operations in parallel, the corresponding partial frame sequences of the protocol candidate sentences must not be overlapped. In addition, to prevent the same protocol sentences from appearing more than once, the score of the protocol candidate sentences which has appeared once in the protocol is set to 0. Under this condition, the score of a protocol candidate sentence can change. Although it should be searched for score maximization, we use the Viterbi algorithm for the calculation, because the change of the score is limited at the time of generation of the same sentence and it is considered that it does not occur so much. By calculating the path of the protocol candidate sentence sequence for increasing the score, the generated protocol sequence is output as a protocol. The higher the score, the more protocol-like the sentences are.

## 5.4 Experiments

In this section, we evaluate the model’s performance on protocol generation from experiment videos. First, we describe the detailed experimental settings, datasets, and evaluation metrics. Then, we show the quantitative and qualitative results and confirm that the model can generate protocols more accurately than the baseline approach. Finally, we analyze the model’s behavior and limitations and discuss the future development of the project.

### 5.4.1 Experimental settings

#### Dataset

A large-scale pair of b-NE sets and sentences is necessary to train the language model. We use the WLP dataset, which consists of b-NE annotations with 18 tag categories. From them,

Table 5.9 Statistics on the WLP dataset, including three filtering settings in the table. The first one is using all of the b-NE tags, the second is using only the object-based NEs, and the third is using only the Reagent and Location tags in the WLP dataset. For each case, we also show the statistics, where more than 20 words are filtered. We use the original dataset split to the original WLP dataset because BioVL2 is used for the test set.

	All b-NE tags			object-based NEs			Reagent and Location		
	#sentences	Avg. #words	Avg. #b-NEs	#sentences	Avg. #words	Avg. #b-NEs	#sentences	Avg. #words	Avg. #b-NEs
w/o filtering									
Training	8,005	15.6	5.8	6,992	16.6	2.3	6,729	16.8	2.2
Validation	2,709	15.5	5.8	2,353	16.6	2.4	2,261	16.7	2.2
w/ filtering									
Training	5,984	11.5	4.6	4,996	12.1	1.9	4,761	12.2	1.8
Validation	2,025	11.3	4.6	1,678	12.0	1.9	1,594	12.0	1.8

we use Reagent, Location and Device tags because the annotated objects in the BioVL2 dataset belong to one of them. In addition, based on the fact that the average number of words in the BioVL2 dataset is less than 10 words, we filter the dataset (1) if the sentence does not contain any of the three tags or (2) if the sentence has more than 20 words. Furthermore, the number of Device tags is much smaller than other tags, thus we modify filtering condition (1) to the case whether the sentence does not contain any of the Reagent and Location. The influence on this filtering is discussed in Chapter 5.4.2 for details. Table 5.9 shows the statistics on the original and filtered dataset. Note that this dataset is used only for training the model because BioVL2 is used for evaluation.

### Data preprocessing and hyperparameters

We replace the words that have at most 5 times frequency with the unknown words to train the model, resulting in a 1,827 vocabulary size. We set the hidden size of the Transformer to 768, the number of layers to 2 and the number of heads to 12 in the multi-head attention layers. We use the positional encoding as with [121]. For train the transformers, we use Adam optimization [56] with  $\alpha = 0.0001$ ,  $\beta_1 = 0.9$  and  $\beta_2 = 0.999$  and L2 delay to 0.01. The first 5 epochs are used for warmup and the maximum epoch is set to be 50. The batch size is set to be 16. For evaluation, we use the trained model that has the minimum loss on the validation set.

### Evaluation metrics

We use BLEU [90], METEOR [8] and ROUGE-L [67] as the word-overlap metrics, which are commonly used in text generation tasks. We report the BLEU scores by varying  $N$  to be  $N = 1, 2, 3, 4$ <sup>4</sup>. Because the number of steps in the generated protocols is not always equal to

<sup>4</sup>For evaluation, we use <https://github.com/tylin/coco-caption> (accessed on 2022/11/10)

Table 5.10 Word-overlap evaluation. The bold scores represent the best among the methods.

Model	BLEU1	BLEU2	BLEU3	BLEU4	METEOR	ROUGE-L
Baseline	3.0	2.0	1.1	0.5	6.2	6.1
All b-NE tags (w/o #word filtering)						
Transformer	15.0	8.7	4.2	2.0	11.1	22.3
Transformer + Copy mechanism	13.0	8.5	5.2	3.3	12.4	28.1
All b-NE tags (w/ #word filtering)						
Transformer	15.2	9.7	6.0	4.0	10.8	19.9
Transformer + Copy mechanism	13.3	7.9	3.7	1.8	12.2	27.6
Object-based b-NEs (w/o #word filtering)						
Transformer	43.8	29.3	19.5	13.4	20.0	29.4
Transformer + Copy mechanism	36.8	26.9	20.0	14.8	18.5	<b>36.7</b>
Object-based b-NEs (w/ #word filtering)						
Transformer	39.9	26.9	18.1	12.6	18.5	27.5
Transformer + Copy mechanism	42.5	30.6	21.3	15.5	20.7	32.7
Reagent and Location (w/o #word filtering)						
Transformer	36.8	25.8	18.2	13.3	19.5	24.9
Transformer + Copy mechanism	44.0	30.9	<b>21.8</b>	<b>16.4</b>	<b>21.7</b>	33.1
Reagent and Location (w/ #word filtering)						
Transformer	38.1	27.2	20.0	15.5	18.8	26.2
Transformer + Copy mechanism	<b>44.7</b>	<b>31.7</b>	<b>21.8</b>	15.2	21.1	32.2

those of the ground truth, it is impossible to conduct the sentence-level evaluation. Instead, we concatenate the sentences and conduct the document-level evaluation.

## 5.4.2 Quantitative evaluation

For the language model, we compare the baseline (outputting b-NEs in the inputs), transformer only, with the transformer+copy mechanism. Table 5.10 shows the word-overlap metrics, revealing the following four insights.

First, it is necessary to train the language model for accurate protocol generation. We can confirm this by comparing the weak baseline with the learning-based approaches of the transformer and transformer+copy mechanism. Second, the WLP dataset can be used for training the language model for the BioVL2 dataset. The WLP dataset collects the protocols in the biological domain, and the research fields are a bit different from the BioVL2 dataset. However, 83% of the vocabulary appears in the WLP dataset and has no big difference between them. Third, the copy mechanism is quite effective for training the model because it outperforms the Transformer only model. It encourages the decoder to generate the input b-NEs correctly, leading the model to generate accurate protocols. Finally, the kinds of input b-NEs for training the model have a big impact on the model’s performance. This can be confirmed by comparing the model that uses all of the b-NE tags as inputs with other models. In addition, the model trained on only Reagent and Location achieves higher performance

















PCR		
<p><b>Step 1</b></p>  <p>sterile distilled water</p> <p><b>Step 2</b></p>  <p>Primer1</p> <p><b>Step 3</b></p>  <p>PrimeSTAR® Max Premix</p>	<p><b>Step 1</b></p>  <p>sterile distilled water</p> <p><b>Step 2</b></p>  <p>Primer1</p> <p><b>Step 3</b></p>  <p>Primer2</p> <p><b>Step 4</b></p>  <p>PrimeSTAR® Max Premix</p> <p><b>Step 5</b></p>  <p>PrimeSTAR® Max Premix, PrimeSTAR® Max Premix</p>	<p><b>Step 1</b></p>  <p>sterile distilled water</p> <p><b>Step 2</b></p>  <p>Primer1</p> <p><b>Step 3</b></p>  <p>Primer2</p> <p><b>Step 4</b></p>  <p>PrimeSTAR® Max Premix</p> <p><b>Step 5</b></p>  <p>Template</p> <p><b>Step 6</b></p>  <p>DNA engine</p>
<b>Transformer only</b>	<b>Transformer + copy mechanism</b>	<b>Ground-truth protocol</b>

Fig. 5.10 Examples of the generated protocols and the ground-truth on PCR. The model in this figure is “Reagent, Location and Device (w/ #word filtering).” We also show the selected frames with b-NE annotation information. Note that the registered trademark symbol ® is shown in the figure, but the generated protocols do not contain them.

than that trained on Reagent, Location and Device. This is because of the Reagent and Location amount for a large part of objects in the BioVL2 dataset. On the other hand, word count filtering does not have a large influence on the model’s performance.

### 5.4.3 Qualitative evaluation

Fig. 5.10 shows the generated protocols from the transformer only and transformer+copy mechanism, in comparison to that of the ground-truth.

#### Insights

As described in Chapter 5.1, the generated protocols should include the actions with the target objects and correct adverb, quantity and time information, if necessary. Although the transformer-only model generally fails to generate objects, the transformer+copy model can generate accurate objects (e.g., “Primer1,” “Primer2,” and “PrimeSTAR” in steps 1, 2 and 4), indicating that the copy mechanism encourages the decoder to verbalize the objects effectively.

Table 5.11 The most frequent top-10 verbs and objects which has the highest PMIs for the verbs on the training set of the WLP dataset. The PMI scores overestimate the low frequency words, thus we report the objects which appear more than 10 times in the dataset.

Verb	Frequency	The top-10 objects with the highest PMI
add	949	dna stripping solution (6.74), precipitation solution (6.74), chloroform (6.65), stand (6.60), genomic lysis buffer (6.49)
incubate	496	dark (6.84), assays (6.80), sections (6.67), nanobeads (6.35), primary antibody (6.29)
mix	374	dna stripping solution (8.08), precipitation solution (7.94), components (7.93), choloform (7.93), contents (6.99)
remove	346	upper phase (8.05), forceps (7.45), petri dish (7.32), xylenes (7.19), upper reservoir (6.81)
centrifuge	339	centrifuge tube (8.22), centrifuge (8.20), upper phase (8.08), cell strainer (7.57), cell pellet (7.36)
place	293	magnetic rack (7.72), petri dish (7.56), top (7.40), magnetic stand (7.30), magnet (7.22)
wash	283	ethanol wash (8.45), wash buffer (8.45), intracellular staining perm wash buffer (8.45), dna pellet (8.21), each well (7.94)
transfer	223	per tube (7.74), microfuge tube (7.69), isopropanol (7.55), cuvette (7.50), aqueous dna (7.09)
discard	220	flow-through (8.40), cell pellet (8.07), supernatants (7.67), spin cartridge (7.57), supernatant (7.52)
resuspend	187	cell strainer (8.43), mojosort (8.28), intracellular staining perm wash buffer (8.20), te (7.81), cell staining buffer (7.78)

## Limitations

Even though the copy mechanism enhances the model to generate objects correctly, this model has clear three limitations. First, it fails to generate verbs accurately. In Fig. 5.10, instead of “prepare” generated by the model, “add” should be verbalized. Our approach does not incorporate visual information except for objects into the inputs. It depends on the language model to generate correct verbs. This is one of the insufficient points to generate reliable protocols. Second, it does not always cover all of the objects in the generated protocols. For example, “DNA engine” is generated in Fig. 5.10, but not included in the generated protocol. Verbalizing all of the objects in the protocols is essential to accurate protocol generation. The third limitation is generating the correct quantity and time. The current outputs, such as “1 volume,” “1 minutes,” and “300l” in Fig. 5.10 are incorrect. As with the action generation, this point also depends on the acquired language model and is insufficient to generate correct protocols.

## 5.4.4 Discussion

Then we move to the analysis of the model’s behavior based on the above insights and limitations. Finally, we discuss the future development of this project.

## 5.4.5 PMI-based analysis on the language model between objects and actions

In steps 2 and 3 in Fig. 5.10, although we input only the object names to the transformer+copy mechanism, the model can verbalize “add.” Our research question arises: why does the model generate correct actions Are there any relationships between the objects and actions? To answer this, we use Point-wise mutual information (PMI), which computes the co-occurrence

Table 5.12 The most frequent top-10 objects and verbs have the highest PMIs for the verbs on the training set of the WLP dataset. The PMI scores overestimate the low frequent words, thus we report the verbs which appear more than 10 times in the dataset.

Object	Frequency	The top-10 verbs with the highest PMI
tube	435	flicking (7.71), placing (7.48), disturbing (7.35), inverting (7.31), insert (7.12)
cells	406	scale up (7.82), seperating (7.72), lyse (7.67), working (7.58), flicking (7.44)
sample	309	load (6.88), insert (6.62), process (6.62), determine (6.47), cool (6.47)
supernatant	298	pour off (7.70), dissociate (7.56), discard (7.43), save (6.95), decent (6.85)
ethanol	171	disturbing (8.69), pipette off (8.53), invert (7.42), immerse (7.30), removed (7.21)
plate	170	plate (9.22), dissociate (8.95), spread (8.37), read (8.32), seal (8.07)
dna	162	precipitate (8.72), invert (8.20), elute (7.91), running (7.55), store (7.50)
ice	162	chill (8.55), thaw (8.55), kept (8.28), keep (8.00), incubating (7.87)
tubes	157	disturbing (8.59), precipitate (7.71), aliquot (7.67), label (7.48), clamp (6.87)
beads	130	disturbing (9.28), vortexing (7.92), seperate (7.60), spin (7.38), pipet (7.32)

of objects and actions and is commonly used to analyze the language model. If the PMI scores are high, objects and actions frequently co-occur in the sentences, otherwise do not. Specifically, PMI is computed as follows:

$$\text{PMI}(\text{object}, \text{action}) = \log_2 \frac{P(\text{object}, \text{action})}{P(\text{object})P(\text{action})} = \log_2 \frac{C(\text{object}, \text{action})N}{C(\text{object})C(\text{action})}, \quad (5.8)$$

where  $C(\cdot)$  computes the number of objects, actions and co-occurrence of them and  $N$  represents the number of all words in the training set.

Table 5.11 and 5.12 show the computed PMI results, indicating that the top-5 values are competitive thus verb/action are not uniquely determined for each object/action. Therefore, why then can the model generate “add” in the protocols correctly? One of the acceptable hypotheses is that the model tends to generate frequent actions. We found that “Primer1” and “Primer2” is processed as unknown words, but the copy mechanism can generate protocols including them by copying them into the output sentences. The action “add” is the most frequent action in this experiment, thus the model generates “add” for them.

#### 5.4.6 Relationship between the number of steps and word-overlap evaluation

Chapter 5.2 reveals the large variety of the number of steps for each experiment, but it is unclear whether the number of steps has a relationship to the word-overlap evaluation. To clarify this, we report the experiment-wise word-overlap evaluation with the number of generated and ground-truth steps in the protocols.

Table 5.13 shows the relationship between the number of steps in the protocols and word-overlap evaluation. The minimum gap between the steps of generated and ground-truth

Table 5.13 Relationship between the number of steps in the protocols and the word-overlap evaluation results. The model we use in this table is “Reagent, Location, and Device (w/ #word filtering).”

		Generated protocols	Ground truth	BLEU1	BLEU2	BLEU3	BLEU4	METEOR	ROUGE-L
DNA extraction	Phenol chloroform	3.0 ( $\pm 0.0$ )	4.0 ( $\pm 0.0$ )	34.5	23.5	16.0	11.2	18.2	39.1
	Ethanol	6.3 ( $\pm 1.5$ )	9.0 ( $\pm 0.0$ )	28.7	15.5	6.7	0.0	15.9	30.0
PCR		5.0 ( $\pm 0.0$ )	6.0 ( $\pm 0.0$ )	23.5	15.0	8.5	0.0	22.3	33.5
Agarose gel creation		5.0 ( $\pm 1.2$ )	10.3 ( $\pm 0.4$ )	33.2	21.4	10.0	0.0	13.8	24.7
Miniprep		10.9 ( $\pm 2.7$ )	28.2 ( $\pm 0.4$ )	49.4	36.6	26.8	19.8	24.5	36.1

protocols is PCR and Phenol chloroform, and the maximum one is miniprep. However, we can observe that the word-overlap metrics achieve the best performance on miniprep despite the gap. The main reason for this is increasing the number of steps causes an increase in the number of words, resulting in the increased agreement of the number of n-grams and higher performance on the word-overlap metrics. In future work, we need to consider the number of steps for evaluation in addition to the word-overlap metrics.

### 5.4.7 Future development of the BioVL project

Based on the insights we learned from this research, we enumerate two future directions of the project as follows:

- **Use QR codes to detect objects.** Although we annotated bounding boxes manually in this research, it is costly and time-consuming. We believe that QR codes are suitable for this project because most objects are solid and do not change their form<sup>5</sup>, unlike the cooking domain, where the object form changes drastically. Inspired by this idea, we start to collect the videos by attaching QR codes to the objects (Fig. 5.11).
- **Create a useful app.** Collaboration with biochemical researchers is essential to collect the wet-lab experiment videos. Although we asked them to take videos in this study, it is necessary to build a framework for researchers to take videos spontaneously in the future. Creating a useful app is one of the promising approaches. Constructing such win-win relationships is essential for future development.

## 5.5 Conclusion

In this chapter, we aim to extend the single target domain to multiple domains. Especially, this study focused on the biochemical domain because of its need of generating protocols from experiment videos in terms of the reproducibility of experiments. The main problem

<sup>5</sup>Liquid is mainly used in the container.

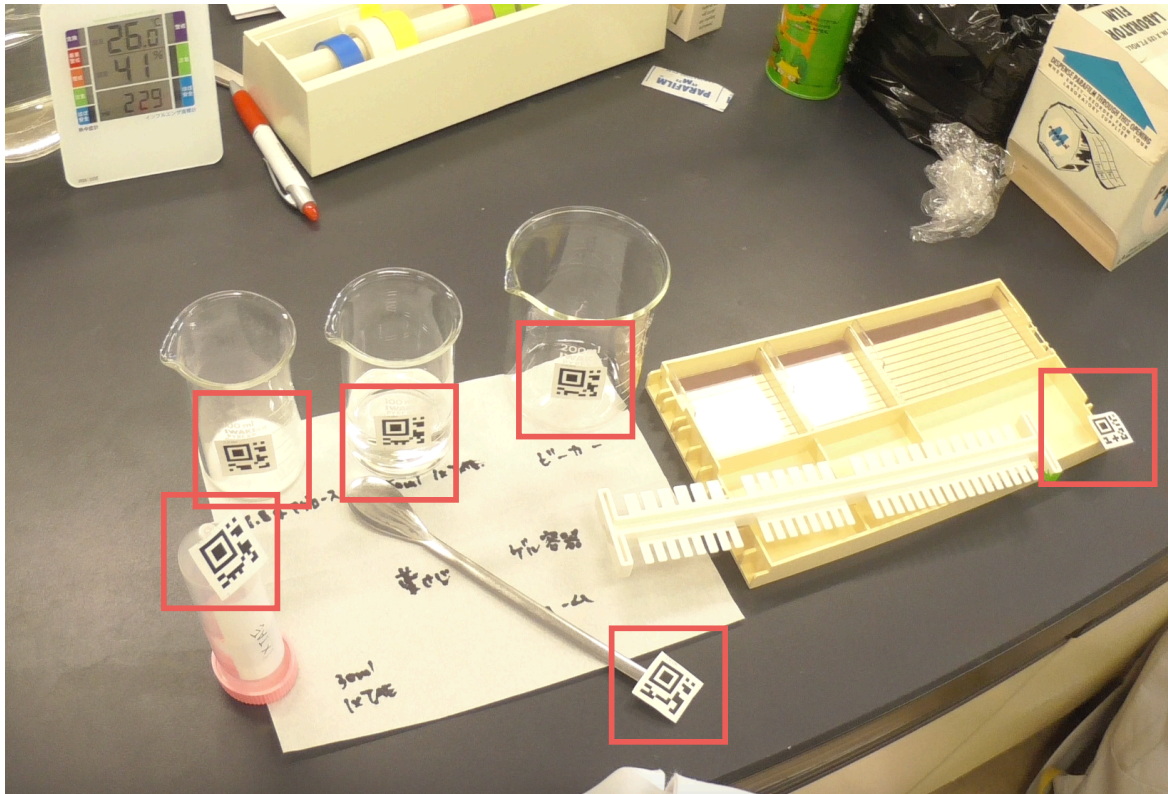


Fig. 5.11 The experiment environments, where QR codes @are attached to objects used in the protocol.

is, there are no public video-and-language datasets online. To address this issue, we first constructed the BioVL2 dataset, which is an egocentric biochemical video-and-language dataset. For each of the four experiments, the BioVL2 dataset has eight videos, summing up to 32 videos. All of the videos are attached with two kinds of video-and-language annotations: (1) event-and-sentence alignment and (2) bounding boxes for objects appearing in the protocols. Based on the constructed BioVL2 dataset, we tackled protocol generation from experiment videos. As a result, the model can generate accurate protocols to a certain extent, compared with the weak baseline approach. We analyze the model's behavior and limitations in detail and discuss the future development of the project. We hope that this dataset enhances video-and-language research, resulting in innovative products to enhance reproducibility in the biochemical domain.



# Chapter 6

## Conclusion

### 6.1 Summary

One of the overarching research topics in multimedia is to construct intelligent multimedia archives by linking multi-modal information for high-level video processing, such as retrieving a video from a video database or searching specific scenes, objects and actions from the videos. Video captioning contributes to this goal because it enables computers to connect multiple modalities by converting video content into textual symbols directly. The motivation of this thesis is to extend this technology to generate procedural text from instructional videos for broad benefits in both academic and industrial communities.

We formulate procedural text generation task as a two-stage prediction task: (1) extracting key events to achieve the goal and (2) generating sentences that reflect on the extracted events. The key challenges of this task are three-fold: (1) developing models capable of generating accurate procedural text based on video content, (2) learning both the event extractor and sentence generator in a story-aware manner and (3) extending our research focus from everyday to important domains that are in high demand for video verbalization.

Challenge (1) has two sub-tasks: (a) recognizing materials in the videos and (b) generating sentences by considering the dependency of actions and materials. In this study, we addressed (b) in the setting of generating procedural text from key events in the video and materials. The essential ability is to be robust to recognize material state changes, which are often accompanied by drastic appearance transformations. To achieve this, we proposed the state-aware approach, which modified the existing NLU reasoning-based simulator into the visual simulator and introduced it into the transformer-based encoder-decoder architecture. The visual simulator effectively represents the state changes as the transition of material vectors in the latent feature space. Without any manual annotations of material state changes, the proposed method achieved better performance than the state-of-the-art video captioning

models, indicating the effectiveness of the visual simulator. In addition, our qualitative evaluation revealed that these models could generate accurate procedural text by considering the state changes of materials effectively. Finally, we discussed the full prediction settings of materials and found that the proposed method worked well with the ground-truth materials, rather than the predicted ones.

Challenge (2) is similar to DVC, which aims at detecting events thoroughly and generating sentences describing them. However, the traditional DVC approaches focused on detecting events in parallel, leading to redundant output without capturing the story awareness. To address this issue, we proposed the multimodal recurrent learning approach of the event selector and sentence generator. Our approach is to predict the next steps of events and sentences by memorizing and mixing the history of the previous prediction, encouraging the model to predict the accurate number of events in the correct order. Our experimental results showed that the proposed methods achieved better performance than the traditional DVC approaches. In addition, we also confirmed that they could generate accurate procedural text by selecting an appropriate number of events in the correct order.

To address Challenge (3), we selected the biochemical domain as the target domain because of its high demand to generate protocols in terms of reproducibility. The main problem is no public biochemical video-and-language datasets available on the web, thus we constructed BioVL2, which consists of 32 egocentric biochemical video-and-language datasets, covering the basic biochemical experiments with diverse characteristics from both the video and language sides. Based on it, we applied the existing procedural text generation model. The experimental results showed that this approach could generate more accurate protocols than the weak baseline that outputs objects appearing in the input frames. We also analyzed the model's behavior and limitations and discussed the future development of the BioVL project.

## 6.2 Limitation

To implement the proposed system for practical use, our research has the following three limitations.

- **It is needed for accurate prediction to input all of the materials.** In Chapters 3 and 4, we indicated the significance of inputting ground-truth materials for precise procedural text generation. However, these constraints render the current system inconvenient for users as they are forced to remember all of the materials and input them in addition to the videos. To make the system more practical and user-friendly, we need to develop a



method to reduce this burden. One idea to address this issue is to predict the materials from the videos beforehand, suggest them to users and allow for corrections if needed.

- **Performance issue for practical use.** From Chapters 3 to 5, we developed procedural text generation models from instructional videos. Although the proposed methods performed better than the baseline methods, they still did not achieve enough performance for practical use. Improving the performance is essential to launch the system to production environments in the future.
- **Limited kinds of data in the biochemical domain.** In Chapter 5, we constructed the BioVL2 dataset, which covers four basic experiments in the biochemical domain. The kinds of experiments we covered are limited, basic and different from the experiments that the wet-lab researchers daily conduct. For practical use, we need to increase the variety of experiments in the biochemical domain. To achieve this, it would be essential to construct a good relationship between informatics and biochemistry researchers.

## 6.3 Future work

Finally, we itemize the future work of this thesis.

### 6.3.1 Build accurate models for untrimmed egocentric videos

The proposed approaches in Chapters 3 and 4 in this thesis were verified on third-person instructional videos collected from the web. Recently, egocentric videos have been paid attention to by researchers because they have rich information about our procedural activities to achieve a task, and video-and-language egocentric datasets have appeared. Building accurate models on egocentric videos is a promising research direction to extend our research from web to real-world data.

### 6.3.2 Build domain-agnostic models to generate procedural text from instructional videos

In this thesis, we focused on training the models in the specific domain: the cooking (Chapters 3, 4) and biochemical domains (Chapter 5). Although the domain-specific supervised approaches worked well, it is necessary to prepare large-scale datasets for each domain. This is costly, thus training domain-agnostic models that do not require domain-specific annotations is desirable for practical use. Recently, with the rapid growth of foundation

models [94, 15], such as GPT4 [87], researchers started to try to generate sentences from images without image-text pairs [118, 143]. Although their performance is currently limited compared with the supervised approaches, the researchers reported that the foundation models can generate accurate text only with a few examples (This is called *in-context learning* [15]) without paired annotations. Utilizing these models is an interesting research direction to achieve domain-agnostic models.

### **6.3.3 Transfer the learned representations into other tasks**

It is known to be effective to transfer the learned representations to other tasks in NLP [26, 38] and CV [94, 20, 74]. The learned representations in this thesis are vision-and-language, thus it is an interesting direction to transfer them into real-world applications. For example, in Chapter 3, we tried to imitate the human-like understanding by modeling the state transition of materials and showed that the material embedding captures the semantic vector arithmetic of actions and materials. If we can inject such knowledge into robots, they might be more intelligent to read procedural text by interpreting the state transitions of materials with visual observations. Investigating the capability of the learned embedding in real-world tasks is a promising research idea that involves CV, NLP and robotics.

# References

- [1] Akbik, A., Blythe, D., and Vollgraf, R. (2018). Contextual string embeddings for sequence labeling. In *Proc. COLING*, pages 1638–1649.
- [2] Alayrac, J.-B., Bojanowski, P., Agrawal, N., Sivic, J., Laptev, I., and Lacoste-Julien, S. (2016). Unsupervised learning from narrated instruction videos. In *Proc. CVPR*, pages 4575–4583.
- [3] Alayrac, J.-B., Sivic, J., Laptev, I., and Lacoste-Julien, S. (2017). Joint discovery of object states and manipulation actions. In *Proc. ICCV*, pages 2127–2136.
- [4] Amac, M. S., Yagcioglu, S., Erdem, A., and Erdem, E. (2019). Procedural reasoning networks for understanding multimodal procedures. In *Proc. CoNLL*, pages 441–451.
- [5] Aradhye, H., Toderici, G., and Yagnik, J. (2009). Video2text: Learning to annotate video content. In *Proc. ICDM Workshop*.
- [6] Bain, M., Nagrani, A., Varol, G., and Zisserman, A. (2021). Frozen in time: A joint video and image encoder for end-to-end retrieval. In *Proc. ICCV*, pages 2183–2192.
- [7] Baker, M. (2016). 1,500 scientists lift the lid on reproducibility. *Nature*, 533:452–454.
- [8] Banerjee, S. and Lavie, A. (2005). METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proc. ACL Workshop IEEMMTS*, pages 65–72.
- [9] Bansal, S., Arora, C., and Jawahar, C. (2022). My view is the best view: Procedure learning from egocentric videos. In *Proc. ECCV*.
- [10] Bojanowski, P., Lajugie, R., Grave, E., Bach, F., Laptev, I., Ponce, J., and Schmid, C. (2015). Weakly-supervised alignment of video with text. In *Proc. ICCV*, pages 4462–4470.
- [11] Bosselut, A., Celikyilmaz, A., He, X., Gao, J., Huang, P.-S., and Choi, Y. (2018a). Discourse-aware neural rewards for coherent text generation. In *Proc. NAACL-HLT*, pages 173–184.
- [12] Bosselut, A., Levy, O., Holtzman, A., Ennis, C., Fox, D., and Choi, Y. (2018b). Simulating action dynamics with neural process networks. In *Proc. ICLR*.
- [13] Brohan, A., Brown, N., Carbajal, J., Chebotar, Y., Dabis, J., Finn, C., Gopalakrishnan, K., Hausman, K., Herzog, A., Hsu, J., Ibarz, J., Ichter, B., Irpan, A., Jackson, T., Jesmonth, S., Joshi, N. J., Julian, R., Kalashnikov, D., Kuang, Y., Leal, I., Lee, K.-H., Levine, S., Lu,

- Y., Malla, U., Manjunath, D., Mordatch, I., Nachum, O., Parada, C., Peralta, J., Perez, E., Pertsch, K., Quiambao, J., Rao, K., Ryoo, M., Salazar, G., Sanketi, P., Sayed, K., Singh, J., Sontakke, S., Stone, A., Tan, C., Tran, H., Vanhoucke, V., Vega, S., Vuong, Q., Xia, F., Xiao, T., Xu, P., Xu, S., Yu, T., and Zitkovich, B. (2022). Rt-1: Robotics transformer for real-world control at scale.
- [14] Brown, P. F., Pietra, S. A. D., Pietra, V. J. D., and Mercer, R. L. (1993). The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19:263–311.
- [15] Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. (2020). Language models are few-shot learners. In *Proc. NeurIPS*, pages 1877–1901.
- [16] Chandu, K., Nyberg, E., and Black, A. W. (2019). Storyboarding of recipes: Grounded contextual generation. In *Proc. ACL*, pages 6040–6046.
- [17] Chang, C.-Y., Huang, D.-A., Sui, Y., Fei-Fei, L., and Niebles, J. C. (2021). D3tw: Discriminative differentiable dynamic time warping for weakly supervised action alignment and segmentation. In *Proc. CVPR*, pages 3546–3555.
- [18] Chang, C.-Y., Huang, D.-A., Xu, D., Adeli, E., Fei-Fei, L., and Niebles, J. C. (2020). Procedure planning in instructional videos. In *Proc. ECCV*, pages 334–350.
- [19] Chen, J. and wah Ngo, C. (2016). Deep-based ingredient recognition for cooking recipe retrieval. In *Proc. ACM MM*, pages 32–41.
- [20] Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. (2020). A simple framework for contrastive learning of visual representations. In *Proc. ICML*, pages 1597–1607.
- [21] Cheng, F. and Bertasius, G. (2022). Tallformer: Temporal action localization with a long-memory transformer. In *Proc. ECCV*, pages 502–521.
- [22] Christel, M., Kanade, T., Mauldin, M., Reddy, R., Sirbu, M., Stevens, S., and Wactlar, H. (1995). Informedia digital video library. *Communications of the ACM*, pages 57 – 58.
- [23] Christel, M., Smith, M., Taylor, C. R., and Winkler, D. B. (1998). Evolving video skims into useful multimedia abstractions. In *Proc. CHI*, pages 171 – 178.
- [24] Cornia, M., Stefanini, M., Baraldi, L., and Cucchiara, R. (2020). Meshed-memory transformer for image captioning. In *Proc. CVPR*, pages 10578–10587.
- [25] Dai, Z., Yang, Z., Yang, Y., Carbonell, J., Le, Q., and Salakhutdinov, R. (2019). Transformer-xl: attentive language models beyond a fixed-length context. In *Proc. ACL*, pages 2978–2988.
- [26] Dalvi, B., Huang, L., Tandon, N., tau Yih, W., and Clark, P. (2018). Tracking state changes in procedural text: a challenge dataset and models for process paragraph comprehension. In *Proc. NAACL*, pages 1595–1604.

- [27] Damen, D., Doughty, H., Farinella, G. M., Fidler, S., Furnari, A., Kazakos, E., Moltisanti, D., Munro, J., Perrett, T., Price, W., and Wray, M. (2018). Scaling egocentric vision: The EPIC-KITCHENS dataset. In *Proc. ECCV*, pages 753–771.
- [28] Das, P., Xu, C., Doell, R. F., and Corso, J. J. (2013). A thousand frames in just a few words: Lingual description of videos through latent topics and sparse object stitching. In *Proc. CVPR*, pages 2634–2641.
- [29] de la Torre, F., Hodgins, J. K., Montano, J., and Valcarcel, S. (2009). Detailed human data acquisition of kitchen activities: the cmu-multimodal activity database (cmu-mmac). In *Proc CHI Workshop. Developing Shared Home Behavior Datasets to Advance HCI and Ubiquitous Computing Research*.
- [30] Deng, C., Chen, S., Chen, D., He, Y., and Wu, Q. (2021). Sketch, ground, and refine: Top-down dense video captioning. In *Proc. CVPR*, pages 234–243.
- [31] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). ImageNet: A large-scale hierarchical image database. In *Proc. CVPR*, pages 248–255.
- [32] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proc. NAACL-HLT*, pages 4171–4186.
- [33] Doman, K., Kuai, C. Y., Takahashi, T., Ide, I., and Murase, H. (2012). Smart video-cooking: a multimedia cooking recipe browsing application on portable devices. In *Proc. ACM MM*, pages 1267–1268.
- [34] Donahue, J., Hendricks, L. A., Rohrbach, M., Venugopalan, S., Guadarrama, S., Saenko, K., and Darrell, T. (2015). Long-term recurrent convolutional networks for visual recognition and description. In *Proc. CVPR*, pages 2625–2634.
- [35] Driess, D., Xia, F., Sajjadi, M. S. M., Lynch, C., Chowdhery, A., Ichter, B., Wahid, A., Tompson, J., Vuong, Q., Yu, T., Huang, W., Chebotar, Y., Sermanet, P., Duckworth, D., Levine, S., Vanhoucke, V., Hausman, K., Toussaint, M., Greff, K., Zeng, A., Mordatch, I., and Florence, P. (2023). Palm-e: An embodied multimodal language model. In *arXiv preprint arXiv:2303.03378*.
- [36] Farhadi, A., Hejrati, M., Sadeghi, M. A., Young, P., Rashtchian, C., Hockenmaier, J., and Forsyth, D. (2010). Every picture tells a story: Generating sentences from images. In *Proc. ECCV*, pages 15–29.
- [37] Fujita, S., Hirao, T., Kamigaito, H., Okumura, M., and Nagata, M. (2020). Soda: Story oriented dense video captioning evaluation framework. In *Proc. ECCV*, pages 517–531.
- [38] Gao, T., Yao, X., and Chen, D. (2021). Simcse: Simple contrastive learning of sentence embeddings. In *Proc. EMNLP*, pages 6884–6910.
- [39] Guadarrama, S., Krishnamoorthy, N., Malkarnenkar, G., Venugopalan, S., Mooney, R., Darrell, T., and Saenko, K. (2013). Youtube2text: Recognizing and describing arbitrary activities using semantic hierarchies and zero-shot recognition. In *Proc. ICCV*, pages 2712–2719.

- [40] Gupta, A. and Durrett, G. (2019). Tracking discrete and continuous entity state for process understanding. In *Proc. NAACL Workshop SPNLP*, pages 7–12.
- [41] Hamada, R., Ide, I., and Sakai, S. (2000a). Associating cooking video with related textbook. In *Proc. ACMMM*, pages 237–241.
- [42] Hamada, R., Ide, I., Sakai, S., and Tanaka, H. (2000b). Structural analysis of cooking preparation steps in japanese. In *Proc. IRAL*, pages 157–164.
- [43] Hamada, R., Okabe, J., Ide, I., Satoh, S., Sakai, S., and Tanaka, H. (2005). Cooking navi: assistant for daily cooking in kitchen. In *Proc. ACMMM*, pages 371–374.
- [44] Harashima, J., Someya, Y., and Kikuta, Y. (2017). Cookpad image dataset: An image collection as infrastructure for food research. In *Proc. ACM SIGIR*, pages 1229–1232.
- [45] Hashimoto, A., Mori, N., Funatomi, T., Yamakata, Y., Kakusho, K., and Minoh, M. (2008). Smart kitchen : A user centric cooking support system. In *Proc. IPMU*, pages 848–854.
- [46] Hashimoto, A., Sasada, T., Yamakata, Y., Mori, S., and Minoh, M. (2014). Kusk dataset: Toward a direct understanding of recipe text and human cooking activity. In *Proc. Ubicomp*, pages 583–588.
- [47] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proc. CVPR*, pages 770–778.
- [48] Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9:1735–1780.
- [49] Huang, D.-A., Lim, J. J., Fei-Fei, L., and Niebles, J. C. (2017). Unsupervised visual-linguistic reference resolution in instructional videos. In *Proc. CVPR*, pages 2183–2192.
- [50] Huang, D.-A., Nair, S., Xu, D., Zhu, Y., Garg, A., Fei-Fei, L., Savarese, S., and Niebles, J. C. (2019). Neural task graphs: Generalizing to unseen tasks from a single video demonstration. In *Proc. CVPR*, pages 8565–8574.
- [51] Ioffe, S. and Szegedy, C. (2015). Batch normalization: accelerating deep network training by reducing internal covariate shift. In *Proc. ICML*, pages 448–456.
- [52] Jang, E., Gu, S., and Poole, B. (2017). Categorical reparametrization with gumble-softmax. In *Proc. ICLR*.
- [53] Jermurawong, J. and Habash, N. (2015). Predicting the structure of cooking recipes. In *Proc. EMNLP*, pages 781–786.
- [54] Kiddon, C., Ponnuraj, G. T., Zettlemoyer, L., and Choi, Y. (2015). Mise en Place: Unsupervised interpretation of instructional recipes. In *Proc. EMNLP*, pages 982–992.
- [55] Kiddon, C., Zettlemoyer, L., and Choi, Y. (2016). Globally coherent text generation with neural checklist models. In *Proc. EMNLP*, pages 329–339.
- [56] Kingma, D. P. and Ba, J. (2015). Adam: A method for stochastic optimization. In *Proc. ICLR*.

- [57] Krishna, R., Hata, K., Ren, F., Fei-Fei, L., and Niebles, J. C. (2017). Dense-captioning events in videos. In *Proc. ICCV*, pages 706–715.
- [58] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Proc. NeurIPS*, pages 1097–1105.
- [59] Kuehne, H., Arslan, A., and Serre, T. (2014). The language of actions: Recovering the syntax and semantics of goal-directed human activities. In *Proc. CVPR*, pages 780–787.
- [60] Kulkarni, C., Xu, W., Ritter, A., and Machiraju, R. (2018). An annotated corpus for machine reading of instructions in wet lab protocols. In *Proc. NAACL-HLT*, pages 97–106.
- [61] Kulkarni, G., Premraj, V., Dhar, S., Li, S., Choi, Y., Berg, A. C., and Berg, T. L. (2011). Baby talk: Understanding and generating simple image descriptions. In *Proc. CVPR*, pages 1601–1608.
- [62] Lafferty, J., McCallum, A., and Pereira, F. C. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. AAAI*, pages 282–289.
- [63] Laroche, R., Dziekan, J., Roussarie, L., and Baczyk, P. (2013). Cooking coach spoken/multimodal dialogue systems. In *Proc. CwC*, pages 34–35.
- [64] LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature*, 521:436–444.
- [65] Lei, J., Wang, L., Shen, Y., Yu, D., Berg, T., and Bansal, M. (2020a). Mart: memory-augmented recurrent transformer for coherent video paragraph captioning. In *Proc. ACL*, pages 2603–2614.
- [66] Lei, J., Yu, L., Berg, T. L., and Bansal, M. (2020b). Tvr: A large-scale dataset for video-subtitle moment retrieval. In *Proc. ECCV*, pages 447–463.
- [67] Lin, C.-Y. and Och, F. J. (2004). Automatic evaluation of machine translation quality using longest common subsequence and skip-bigram statistics. In *Proc. ACL*, pages 605–612.
- [68] Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. (2014). Microsoft coco: Common objects in context. In *Proc. ECCV*, pages 740–755.
- [69] Liu, Y. and Lapata, M. (2019). Hierarchical transformers for multi-document summarization. In *Proc. ACL*, pages 5070–5081.
- [70] Lu, J., Batra, D., Parikh, D., and Lee, S. (2019). Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks. In *Proc. NeurIPS*.
- [71] Maeta, H., Sasada, T., and Mori, S. (2015). A framework for procedural text understanding. In *Proc. IWPT*, pages 50–60.
- [72] Malmaud, J., Huang, J., Rathod, V., Johnston, N., Rabinovich, A., and Murphy, K. (2015). What’s cookin’? interpreting cooking videos using text, speech and vision. In *Proc. NAACL*, pages 143–152.

- [73] Mao, J., Huang, J., Toshev, A., Camburu, O., Yuille, A. L., and Murphy, K. (2016). Generation and comprehension of unambiguous object descriptions. In *Proc. CVPR*, pages 11–20.
- [74] Miech, A., Alayrac, J.-B., Smaira, L., Laptev, I., Sivic, J., and Zisserman, A. (2020). End-to-end learning of visual representations from uncurated instructional videos. In *Proc. CVPR*, pages 9879–9889.
- [75] Miech, A., Zhukov, D., Alayrac, J.-B., Tapaswi, M., Laptev, I., and Sivic, J. (2019). HowTo100M: Learning a text-video embedding by watching hundred million narrated video clips. In *Proc. ICCV*, pages 2630–2640.
- [76] Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *NeurIPS*, pages 3111–3119.
- [77] Mintz, M., Bills, S., Snow, R., and Jurafsky, D. (2009). Distant supervision for relation extraction without labeled data. In *Proc. ACL-IJCNLP*, pages 1003–1011.
- [78] Miura, K., Hamada, R., Ide, I., Sakai, S., and Tanaka, H. (2002). Motion based automatic abstraction of cooking videos. In *Proc. ACMMM Workshop on Multimedia Information Retrieval*, pages 29–32.
- [79] Momouchi, Y. (1980). Control structures for actions in procedural texts and pt-chart. In *Proc. COLING*, pages 108–114.
- [80] Mori, S., Maeta, H., Yamakata, Y., and Sasada, T. (2014). Flow graph corpus from recipe texts. In *Proc. LREC*, pages 2370–2377.
- [81] Naim, I., Song, Y., Liu, Q., Kautz, H., Luo, J., and Gildea, D. (2014). Unsupervised alignment of natural language instructions with video segments. In *Proc. AAAI*, pages 1558–1564.
- [82] Naim, I., Song, Y. C., Liu, Q., Huang, L., Kautz, H., Luo, J., and Gildea, D. (2015). Discriminative unsupervised alignment of natural language instructions with corresponding video segments. In *Proc. NAACL*, pages 164–174.
- [83] Nishimura, T., Hashimoto, A., and Mori, S. (2019). Procedural text generation from a photo sequence. In *Proc. INLG*, pages 409–414.
- [84] Nishimura, T., Hashimoto, A., Ushiku, Y., Kameko, H., and Mori, S. (2021). State-aware video procedural captioning. In *Proc. ACMMM*, pages 1766–1774.
- [85] Nishimura, T., Hashimoto, A., Ushiku, Y., Kameko, H., Yamakata, Y., and Mori, S. (2020). Structure-aware procedural text generation from an image sequence. *IEEE Access*, 9:2125–2141.
- [86] Nouri, E., Sim, R., Fourney, A., and White, R. W. (2020). Proactive suggestion generation: Data and methods for stepwise task assistance. In *Proc. SIGIR*, pages 1585–1588.
- [87] OpenAI (2023). Gpt-4 technical report.



- [88] over Vision to Describe Images, C. L. (2021). Ankush gupta and yashaswi verma and c. jawahar. In *Proc. AAAI*, pages 606–612.
- [89] Pan, L., Chen, J., Wu, J., Liu, S., Ngo, C.-W., Kan, M.-Y., Jiang, Y.-G., and Chua, T.-S. (2020). Multi-modal cooking workflow construction for food recipes. In *Proc. ACMMM*, pages 1132–1141.
- [90] Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). BLEU: A method for automatic evaluation of machine translation. In *Proc. ACL*, pages 311–318.
- [91] Park, J. S., Rohrbach, M., Darrell, T., and Rohrbach, A. (2019). Adversarial inference for multi-sentence video description. In *Proc. CVPR*, pages 6598–6608.
- [92] Pennington, J., Socher, R., and Manning, C. (2014). Glove: global vectors for word representation. In *Proc. EMNLP*, pages 1532–1543.
- [93] Plummer, B. A., Wang, L., Cervantes, C. M., Caicedo, J. C., Hockenmaier, J., and Lazebnik, S. (2017). Flickr30k entities: Collecting region-to-phrase correspondences for richer image-to-sentence models. *IJCV*, 123:74–93.
- [94] Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., Krueger, G., and Sutskever, I. (2021). Learning transferable visual models from natural language supervision. In *Proc. ICML*, pages 8748–8763.
- [95] Radford, A., Metz, L., and Chintala, S. (2015). Unsupervised representation learning with deep convolutional generative adversarial networks. arXiv.
- [96] Rajpurkar, P., Jia, R., and Liang, P. (2018). Know what you don’ t know: Unanswerable questions for squad. In *Proc. ACL*, pages 784–789.
- [97] Regneri, M., Rohrbach, M., Wetzell, D., Thater, S., Schiele, B., and Pinkal, M. (2013). Grounding action descriptions in videos. *TACL*, 1:25–36.
- [98] Ren, S., He, K., Girshick, R., and Sun, J. (2016). Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39:1137–1149.
- [99] Rohrbach, M., Rohrbach, A., Regneri, M., Amin, S., Andriluka, M., Pinkal, M., and Schiele, B. (2015). Recognizing fine-grained and composite activities using hand-centric features and script data. *IJCV*, 119:1–28.
- [100] Salvador, A., Drozdal, M., Giro-i-Nieto, X., and Romero, A. (2019). Inverse Cooking: recipe generation from food images. In *Proc. CVPR*, pages 10453–10462.
- [101] Salvador, A., Hynes, N., Aytar, Y., Marin, J., Ofli, F., Weber, I., and Torralba, A. (2017). Learning cross-modal embeddings for cooking recipes and food images. In *Proc. CVPR*, pages 3020–3028.
- [102] Santoro, A., Faulkner, R., Raposo, D., Rae, J., Chrzanowski, M., Weber, T., Wierstra, D., Vinyals, O., Pascanu, R., and Lillicrap, T. (2019). Relational recurrent neural networks. In *Proc. NeurIPS*, pages 7299–7310.

- [103] See, A., Liu, P. J., and Manning, C. D. (2017). Get to the point: Summarization with pointer-generator networks. In *Proc. ACL*, pages 1073–1083.
- [104] Sener, F., Chatterjee, D., Shelepov, D., He, K., Singhanian, D., Wang, R., and Yao, A. (2022). Assembly101: A large-scale multi-view video dataset for understanding procedural activities. In *Proc. CVPR*.
- [105] Shen, Y., Wang, L., and Elhamifar, E. (2021). Learning to segment actions from visual and language instructions via differentiable weak sequence alignment. In *Proc. CVPR*, pages 10156–10165.
- [106] Shi, B., Ji, L., Liang, Y., Duan, N., Chen, P., Niu, Z., and Zhou, M. (2019). Dense procedure captioning in narrated instructional videos. In *Proc. ACL*, pages 6382–6391.
- [107] Shi, B., Ji, L., Niu, Z., Duan, N., Zhou, M., and Chen, X. (2020). Learning semantic concepts and temporal alignment for narrated video procedural captioning. In *Proc. ACM MM*, pages 4355–4363.
- [108] Shibata, T. (2007). Structural understanding of instruction videos by integrating linguistic and visual information.
- [109] Shibata, T. and Kurohashi, S. (2006). Unsupervised topic identification by integrating linguistic and visual information based on hidden markov models. In *Proc. ACL-COLING*, pages 755–762.
- [110] Shou, Z., Wang, D., and Chang, S.-F. (2016). Temporal action localization in untrimmed videos via multi-stage cnns. In *CVPR*, pages 1049–1058.
- [111] Smith, M. and Kanade, T. (1997). Video skimming and characterization through the combination of image and language understanding techniques. In *Proc. CVPR*, pages 775–781.
- [112] Stein, S. and McKenna, S. J. (2013). Combining embedded accelerometers with computer vision for recognizing food preparation activities. In *Proc. UbiComp*, pages 729–738.
- [113] Sun, C., Myers, A., Vondrick, C., Murphy, K., and Schmid, C. (2019). Videobert: a joint model for video and language representation learning. In *Proc. ICCV*, pages 7464–7473.
- [114] Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Proc. NeurIPS*, pages 3104–3112.
- [115] Tan, G., Liu, D., Wang, M., and Zha, Z.-J. (2020). Learning to discretely compose reasoning module networks for video captioning. In *Proc. IJCAI*, pages 745–752.
- [116] Tang, Y., Ding, D., Rao, Y., Zheng, Y., Zhang, D., Zhao, L., Lu, J., and Zhou, J. (2019). Coin: A large-scale dataset for comprehensive instructional video analysis. In *Proc. CVPR*, pages 1207–1216.
- [117] Tenorth, M., Bandouch, J., and Beetz, M. (2009). The tum kitchen data set of everyday manipulation activities for motion tracking and action recognition. In *Proc. ICCV Workshop. THEMIS*, pages 1089–1096.

- [118] Tewel, Y., Shaley, Y., Schwartz, I., and Wolf, L. (2022). Zerocap: Zero-shot image-to-text generation for visual-semantic arithmetic. In *Proc. CVPR*, pages 17918–17928.
- [119] Ushiku, A., Hashimoto, H., Hashimoto, A., and Mori, S. (2017). Procedural text generation from an execution video. In *Proc. IJCNLP*, pages 326–335.
- [120] van der Maaten, L. and Hinton, G. (2008). Visualizing data using t-sne. *Journal of Machine Learning Research*, 9:2579–2605.
- [121] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. In *Proc. NeurIPS*, pages 5998–6008.
- [122] Vedantam, R., Zitnick, C. L., and Parikh, D. (2015). CIDEr: consensus-based image description evaluation. In *Proc. CVPR*, pages 4566–4575.
- [123] Venugopalan, S., Rohrbach, M., Donahue, J., Mooney, R., Darrell, T., and Saenko, K. (2015). Sequence to sequence-video to text. In *Proc. ICCV*, pages 4534–4542.
- [124] Vinyals, O., Toshev, A., Bengio, S., and Erhan, D. (2015). Show and tell: A neural image caption generator. In *Proc. CVPR*, pages 3156–3164.
- [125] Viterbi, A. (1967). Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, 13:260–269.
- [126] Wactlar, H. D., Kanade, T., Smith, M. A., and Stevens, S. (1996). Intelligent access to digital video: Informedia project. *Computer*, 29:46–52.
- [127] Wang, H., Lin, G., Hoi, S. C. H., and Miao, C. (2020). Structure-aware generation network for recipe generation from images. In *Proc. ECCV*, pages 359–374.
- [128] Wang, L., Xiong, Y., Wang, Z., Qiao, Y., Lin, D., Tang, X., and Gool, L. V. (2019a). Temporal segment networks for action recognition in videos. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, pages 2740–2755.
- [129] Wang, T., Zhang, R., Lu, Z., Zheng, F., Cheng, R., and Luo, P. (2021a). End-to-end dense video captioning with parallel decoding. In *Proc. ICCV*, pages 6847–6857.
- [130] Wang, T., Zheng, H., Yu, M., Tian, Q., and Hu, H. (2021b). Event-centric hierarchical representation for dense video captioning. *IEEE Trans. on Circuits and Systems for Video Technology*, pages 1890–1900.
- [131] Wang, W., Wang, Y., Chen, S., and Jin, Q. (2019b). YouMakeup: A large-scale domain-specific multimodal dataset for fine-grained semantic comprehension. In *Proc. EMNLP-IJCNLP*, pages 5133–5143.
- [132] Williams, R. J. and Zipser, D. (1989). A learning algorithm for continually running fully recurrent neural networks. *Neural Computation*, 1:270–280.
- [133] Wu, J., Pan, L., Chen, J., and Jiang, Y.-G. (2022). Ingredient-enriched recipe generation from cooking videos. In *Proc. ICMR*, pages 249–257.
- [134] Xiong, Y., Dai, B., and Lin, D. (2018). Move forward and tell: a progressive generator of video descriptions. In *Proc. ECCV*, pages 489–505.

- [135] Xu, H., Ghosh, G., Huang, P.-Y., Okhonko, D., Aghajanyan, A., Metze, F., Zettlemoyer, L., and Feichtenhofer, C. (2021). Videoclip: Contrastive pre-training for zero-shot video-text understanding. In *Proc. EMNLP*, pages 6787–6800.
- [136] Xu, J., Mei, T., Yao, T., and Rui, Y. (2016). MSR-VTT: A large video description dataset for bridging video and language. In *Proc. CVPR*, pages 5288–5296.
- [137] Yagcioglu, S., Erdem, A., Erdem, E., and Ikizler-Cinbis, N. (2018). RecipeQA: A challenge dataset for multimodal comprehension of cooking recipes. In *Proc. EMNLP*, pages 1358–1368.
- [138] Yamakata, Y., Mori, S., and Carroll, J. (2020). English recipe flow graph corpus. In *Proc. LREC*, pages 5187–5194.
- [139] Yamakata, Y., Kakusho, K., and Minoh, M. (2010). Object recognition based on object’s identity for cooking recognition task. In *Proc. ISM*, pages 278–283.
- [140] Young, P., Lai, A., Hodosh, M., and Hockenmaier, J. (2014). From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions. *TACL*, 2:67–78.
- [141] Yu, L., Poirson, P., Yang, S., Berg, A. C., and Berg, T. L. (2016). Modeling context in referring expressions. In *Proc. ECCV*, pages 69–85.
- [142] Zamir, N., Noy, A., Friedman, I., Protter, M., and Zelnik-Manor, L. (2020). Asymmetric loss for multi-label classification. *arXiv*.
- [143] Zeng, A., Attarian, M., Ichter, B., Choromanski, K., Wong, A., Welker, S., Tombari, F., Purohit, A., Ryoo, M., Sindhwani, V., Lee, J., Vanhoucke, V., and Florence, P. (2022). Socratic models: Composing zero-shot multimodal reasoning with language. *arXiv*.
- [144] Zhao, H., Hadji, I., Dvornik, N., Derpanis, K. G., Wildes, R. P., and Jepson, A. D. (2022). P3iv: Probabilistic procedure planning from instructional videos with weak supervision. In *Proc. ECCV*, pages 2938–2948.
- [145] Zhou, L., Kalantidis, Y., Chen, X., Corso, J. J., and Rohrbach, M. (2019a). Grounded video description. In *Proc. CVPR*, pages 6578–6587.
- [146] Zhou, L., Louis, N., and Corso, J. J. (2019b). Weakly-supervised video object grounding from text by loss weighting and object interaction. In *Proc. BMVC*.
- [147] Zhou, L., Xu, C., and Corso, J. J. (2018a). Towards automatic learning of procedures from web instructional videos. In *Proc. AAAI*, pages 7590–7598.
- [148] Zhou, L., Zhou, Y., Corso, J. J., Socher, R., and Xiong, C. (2018b). End-to-end dense video captioning with masked transformer. In *Proc. CVPR*, pages 8739–8748.
- [149] Zhukov, D., Alayrac, J.-B., Cinbis, R. G., Fouhey, D., Laptev, I., and Sivic, J. (2019). Cross-task weakly supervised learning from instructional videos. In *Proc. CVPR*, pages 3537–3545.
- [150] 任天堂 (2006). 任天堂：しゃべる！dsお料理ナビ.

# List of Publications

## Journal (Peer Reviewed)

1. Taichi Nishimura, Atsushi Hashimoto, Yoshitaka Ushiku, Hirotaka Kameko, and Shinsuke Mori, “Recipe Generation from Unsegmented Videos,” ACM Transactions on Multimedia Computing, Communications, and Applications (Major revision), 2023 → **Chapter 4**.
2. Taichi Nishimura, Atsushi Hashimoto, Yoshitaka Ushiku, Hirotaka Kameko, and Shinsuke Mori, “State-aware Video Procedural Captioning,” Multimedia Tools and Applications, 2023 → **Chapter 3**.
3. 西村太一, 迫田航次郎, 牛久敦, 橋本敦史, 奥田奈津子, 小野富三人, 亀甲博貴, 森信介, “BioVL2 データセット: 生化学分野における一人称視点の実験映像への言語アノテーション”, 自然言語処理, 第29巻4号, 2022 (優秀論文賞) → **Chapter 5**.
4. Taichi Nishimura, Atsushi Hashimoto, Yoshitaka Ushiku, Hirotaka Kameko, Yoko Yamakata, and Shinsuke Mori, “Structure-Aware Procedural Text Generation from an Image Sequence,” IEEE Access, Vol 9, 2021.
5. 西村太一, 橋本敦史, 森信介, “料理における重要語に着目した写真列からのレシピの自動生成”, 自然言語処理, 第27巻2号, 2020.

## Journal (Peer Reviewed, Co-author)

1. 白井圭佑, 橋本敦史, 西村太一, 亀甲博貴, 栗田修平, 森信介, “調理動作後の物体の視覚的状态予測を目指した Visual Recipe Flow データセットの構築と評価”, 自然言語処理, 2023 (採録決定).

### International Conference (Peer Reviewed)

1. Taichi Nishimura, Katsuhiko Ishiguro, Keita Higuchi, and Masaaki Kotera, “Multimodal Dish Pairing: Predicting Side Dishes to Serve with a Main Dish,” In Proceedings of the 1st International Workshop on Multimedia for Cooking, Eating, and related Applications in conjunction with ACMMM, 2022 (**Best Paper Award**).
2. Taichi Nishimura, Atsushi Hashimoto, Yoshitaka Ushiku, Hirotaka Kameko, and Shinsuke Mori, “State-aware Video Procedural Captioning,” In Proceedings of the 29th ACM International Conference on Multimedia, 2021.
3. Taichi Nishimura, Kojiro Sakoda, Atsushi Hashimoto, Yoshitaka Ushiku, Natsuko Tanaka, Fumihito Ono, Hirotaka Kameko, and Shinsuke Mori, “Egocentric Biochemical Video-and-Language Dataset,” In Proceedings of the 4th Workshop on Closing the Loop Between Vision and Language in conjunction with ICCV, 2021.
4. Taichi Nishimura, Suzushi Tomori, Hayato Hashimoto, Atsushi Hashimoto, Yoko Yamakata, Jun Harashima, Yoshitaka Ushiku, and Shinsuke Mori, “Visual Grounding Annotation of Recipe Flow Graph,” In Proceedings of the 12th International Conference on Language Resources and Evaluation, 2020.
5. Taichi Nishimura, Atsushi Hashimoto, and Shinsuke Mori, “Procedural Text Generation from a Photo Sequence,” In Proceedings of the 12th International Conference on Natural Language Generation, 2019.
6. Taichi Nishimura, Atsushi Hashimoto, Yoko Yamakata and Shinsuke Mori, “Frame Selection for Producing Recipe with Pictures from an Execution Video of a Recipe,” In Proceedings of the 11th Workshop on Multimedia for Cooking and Eating Activities in conjunction with ICMR, 2019 (**Best Paper Award**).

### International Conference (Peer Reviewed, Co-author)

1. Keisuke Shirai, Atsushi Hashimoto, Taichi Nishimura, Hirotaka Kameko, Shuhei Kurita, Yoshitaka Ushiku and Shinsuke Mori. “Visual Recipe Flow: A Dataset for Learning Visual State Changes of Objects with Recipe Flows,” In Proceedings of the 29th International Conference on Computational Linguistics, 2022.
2. Sara Ozeki, Masaaki Kotera, Katsuhiko Ishiguro, Taichi Nishimura, Keita Higuchi, “Recipe Recommendation for Balancing Ingredient Preference and Daily Nutrients,” In

- Proceedings of The 1st International Workshop on Multimedia for Cooking, Eating, and related APplications 2022 in conjunction with ACMMM2022, 2022.
3. Kento Tanaka, Taichi Nishimura, Hiroaki Nanjo, Keisuke Shirai, Hirotaka Kameko, and Masatake Dantsuji, “Image Description Dataset for Language Learners,” In Proceedings of the 13th International Conference on Language Resources and Evaluation, 2022.
  4. Jieyong Zhu, Taichi Nishimura, Makoto Goto, and Shinsuke Mori, “Multimedia Retrieval of Historical Materials,” In Digital Humanities, 2022.
  5. Atsushi Hashimoto, Taichi Nishimura, Yoshitaka Ushiku, Hirotaka Kameko, and Shinsuke Mori, “Cross-modal Representation Learning for Understanding Manufacturing Procedure,” In Proceedings of the 23th International Conference on Human-Computer Interaction, 2022.

### Domestic Conference (Non Peer-Reviewed)

1. 西村太一, “「BioVL2データセット: 生化学分野における一人称視点の実験映像への言語アノテーション」の研究経緯”, 自然言語処理.
2. 大野けやき, 西村太一, 亀甲博貴, 森信介, “テキスト中の場所表現認識と係り受けに基づく緯度経度推定ツールの開発”, 言語処理学会第29回年次大会 (NLP2023).
3. 木下聖, 西村太一, 亀甲博貴, 森信介, “株式投資家の関心を考慮したニュース記事抽出によるストーリー生成”, 言語処理学会第29回年次大会 (NLP2023).
4. 山本航輝, 西村太一, 亀甲博貴, 森信介, “VideoCLIPを用いた実験動画からのプロトコル生成”, 言語処理学会第29回年次大会 (NLP2023).
5. 吉田智哉, 西村太一, 亀甲博貴, 森信介, “単語の階層関係に基づくデータ拡張を利用した画像キャプション生成の検討”, 言語処理学会第29回年次大会 (NLP2023).
6. 森田康介, 西村太一, 亀甲博貴, 森信介, “テキストアナリティクスツールのログからの実験設定の説明文生成”, 言語処理学会第29回年次大会 (NLP2023).

7. 八木拓真, 西村太一, 清丸寛一, 唐井希, “大規模言語モデルからの知識抽出に基づく画像からのスクリプト予測の検討”, 言語処理学会第29回年次大会 (NLP2023).
8. 尾関沙羅, 小寺正明, 石黒勝彦, 西村太一, 樋口啓太, “ユーザ嗜好と栄養摂取基準に基づくレシピ推薦手法の開発”, 第30回インタラクティブシステムとソフトウェアに関するワークショップ (WISS2022).
9. 森田康介, 西村太一, 亀甲博貴, 森信介, “テキストマイニングツールのログからの実験設定の説明文生成”, 第253回自然言語処理研究会 (NL253).
10. 西村太一, 橋本敦史, 牛久祥孝, 森信介, 映像からのストーリー生成: イベント選択器と文生成器の同時学習, 言語処理学会第28回年次大会 (NLP2022).
11. 迫田航次郎, 西村太一, 森信介, 小野富三人, 田中奈津子, 生化学分野におけるVideo&Languageデータセットの構築, 言語処理学会第28回年次大会 (NLP2022).
12. 星島洸明, 西村太一, 亀甲博貴, 森信介, “市民科学でのアノテーション作業支援と作業者の能力向上支援”, 言語処理学会第28回年次大会 (NLP2022).
13. 田中健斗, 西村太一, 南條浩輝, 白井圭佑, 亀甲博貴, “画像描写問題における学習者作文の訂正文生成”, 言語処理学会第28回年次大会 (NLP2022).
14. Jieyong Zhu, Taichi Nishimura, Makoto Goto, Shinsuke Mori, “Cross-modal Retrieval of Historical Materials,” 言語処理学会第28回年次大会 (NLP2022).
15. 田中健斗, 西村太一, 白井圭佑, 亀甲博貴, 森信介, “写真描画問題における自動採点手法の検討”, 人工知能学会全国大会 (JSAI2021).
16. 西村太一, 橋本敦史, 牛久祥孝, 森信介, “手順構造を考慮した作業映像からの手順書生成”, 言語処理学会第27回年次大会 (NLP2021).
17. 迫田航次郎, 西村太一, 森信介, 順構造を考慮した手順書からの作業画像検索, 言語処理学会第27回年次大会 (NLP2021).
18. 星島洸明, 西村太一, 亀甲博貴, 森信介, “複数作業者を想定したアノテーションツールの作成と機能の検討”, 言語処理学会第27回年次大会 (NLP2021).



19. 西村太一, 橋本敦史, 牛久祥孝, 森信介, “写真列と構造要素からの手順構造と手順書の同時学習”, 言語処理学会第26回年次大会 (NLP2020).
20. 西村太一, 友利涼, 橋本隼人, 橋本敦史, 山肩洋子, 原島純, 牛久祥孝, 森信介, “レシピフローグラフへのVisual Groundingアノテーション”, 言語処理学会第26回年次大会 (NLP2020).
21. 西村太一, 橋本敦史, 森信介, “作業写真列からの手順書の自動生成”, ヒューマンコミュニケーション基礎研究会 (HCS2019).
22. 西村太一, 橋本敦史, 原島純, 山肩洋子, 森信介, Bounding Boxを付与したフローグラフコーパスの提案, 自然言語処理若手の会 (yans2019).
23. 西村太一, 橋本敦史, 山肩洋子, 森信介, 写真付き手順書生成のための実施映像からのフレーム選択, 第11回データ工学と情報マネジメントに関するフォーラム (DEIM2019).
24. 島崎郁花, 西村太一, 上岡玲子, テキスタイルセンサを用いた腹巻型笑いログシステムによる笑い検出の検討, 第22回日本バーチャルリアリティ学会 (VRSJ2017).

