Interpreting Instructional Texts Towards Robot Execution

Keisuke Shirai

Abstract

Procedural text provides concrete steps to reach a specific goal. Research on procedural texts has long been addressed in natural language processing, multimedia, and robotics. We are interested in building intelligent robot systems that interpret and act on procedural texts. Our ultimate goal is to build a robot system that works on procedural texts in the real world. We consider that such robots must be equipped with various capabilities: interpreting the workflow of procedural texts, predicting the post-action states of objects, making plans interpretable by robots, and executing actions based on them. This thesis consists of research works contributing the realization of the first three capabilities with a primary focus on the cooking domain.

Chapter 2 focuses on the problem of predicting the post-action visual states of objects in the cooking domain. The prediction is performed based on the current state of the object, and this requires interpreting the workflow of the cooking recipe. In addition to this, capturing the visual state changes of objects requires annotations of pre-action and post-action visual states of objects. In this direction of research, no dataset with the above features has been proposed. To this end, we create the Visual Recipe Flow (VRF) dataset, a new multimodal cooking dataset. We describe the data collection process, annotation procedure, statistics, and the quality of the annotations in order. Finally, a multimodal retrieval experiment assesses the importance of textual and visual annotations for predicting the post-action states of objects.

Chapter 3 focuses on the problem of converting linguistic instructions into plans interpretable by robots. Recently, an approach of using large language models (LLMs) to convert the instructions into plans directly has attracted attention. However, there are two issues with this approach. One is that the conversion is performed end-to-end,

and it lacks interpretability, which is an essential factor for the continued development of a method. The other is that the obtained plans are not necessarily correct because LLMs do not consider the feasibility of the plans. To tackle this problem, we develop a Vision-Language Interpreter that generates problem descriptions and drives a symbolic planner to find valid plans based on the descriptions. The problem descriptions provide task specifications written in formal language, which are interpretable by humans. In addition to this, the plans obtained through the symbolic planner are guaranteed to be logically correct. To evaluate the performance of ViLaIn, we constructed the Problem Description Generation (ProDG) dataset. Experimental results show that ViLaIn can generate logically correct problem descriptions and find valid plans with high accuracy.

Chapter 4 focuses on the problem of interpreting the workflow of procedural texts. In the cooking domain, previous work proposed a recipe flow graph representation (r-FG) that represents cooking recipes as flow graphs. The r-FG captures the dependencies of actions in the text, which is vital information for robots that work on procedural texts. Extending the r-FG to handle procedural texts in non-cooking domains has various benefits. Thus, we propose an extended representation of the r-FG. Concretely, we focus on wikiHow articles and propose a wikiHow flow graph (w-FG) representation. To investigate flow graph prediction performance in non-cooking domains, we create the w-FG corpus, a new flow graph corpus from wikiHow articles. In experiments, we assume a low-resource setting given the high annotation cost of flow graphs. We use the domain adaptation from the existing r-FG corpus to the w-FG corpus to address this problem. Experimental results show that the domain adaptation model significantly improves flow graph prediction performance in the target domains compared to baseline models.

Acknowledgments

First of all, I would like to express my gratitude to my supervisor, Professor Shinsuke Mori. This thesis could never have been completed without his support. In my graduate school life, I was very fortunate to have had the opportunity to learn about his research philosophy and his approach to solving problems.

I would like to thank Professor Hisashi Kashima and Professor Yuichi Nakamura for joining my PhD committee and checking my thesis. Their thoughtful advice helped me to make my thesis more coherent.

I wish to thank Assistant Professor Hirotaka Kameko, who has been a great help to me throughout my graduate student life. I have discussed with him a great deal of research topics, but I have always been amazed by his quick thinking. I am thankful to Taichi Nishimura for providing me with many insightful and helpful comments on research. I am grateful to Secretary Asuka Kimura. I am indebted to her for her help with paperwork related to my research. I would like to additionally thank all the other laboratory members.

I would like to express my sincere gratitude to Atsushi Hashimoto at OMRON SINIC X Corporation (OSX). During my PhD, he has been supported me with extensive guidance as practically the second supervisor. I am grateful to Yoshitaka Ushiku at OSX for his many insightful comments on research. I am also thankful to Cristian C. Beltran-Hernandez, Masashi Hamaya, Shohei Tanaka, and Kazutoshi Tanaka at OSX and Kento Kawaharazuka at the University of Tokyo for having fruitful discussions on my joint research project. I have really enjoyed my research internship at OSX thanks to many people, and I extend my thanks to them.

I would like to express my deepest gratitude to Professor Takashi Ninomiya at

Ehime University, my undergraduate supervisor. When I was an undergraduate student, I was fortunate to have the opportunity to learn how to think in academia from him. I would like to thank Kazuma Hashimoto at Google Research and Akiko Eriguchi at Microsoft Research. When I was a master's student, I had the opportunity to collaborate with them on machine translation research, but I was always impressed by their thoughtfulness in research. I would like to thank Shuhei Kurita at the RIKEN Center for Advanced Intelligence Project for his appropriate comments on research. I would additionally thank Zhao Tianyu and Kei Sawada at rinna Co., Ltd for their help during my internship at rinna.

Finally, I want to thank my family for supporting me throughout my graduate school life.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Natural language is a communication tool for humans. We use natural language texts to record and convey information such as knowledge and skills to future generations. The natural language texts include news articles, encyclopedia pages, and academic papers. Among them, procedural text, such as cooking recipes and furniture assembly instructions, is unique in that it consists of multiple steps to achieve a specific goal. The steps are linguistic instructions that describe concrete actions to be taken. Research on instructional texts, including procedural texts and linguistic instructions, has long been conducted in natural language processing, multimedia, and robotics.

We are interested in building intelligent robot systems that interpret and act on procedural texts by integrating these lines of research. Our ultimate goal is to build a robot system that actually works based on procedural texts in the real world. We consider that such a robot must be equipped with various capabilities: (i) interpreting the workflow of procedural texts (Mori et al., 2014; Maeta et al., 2015; Kiddon et al., 2015), (ii) predicting the post-action visual states of objects (Souček et al., 2022, 2023), (iii) making plans interpretable by robots (MacMahon et al., 2006; Matuszek et al., 2010; Lin et al., 2023), and (iv) executing actions based on them. As illustrated in Figure 1.1, our system integrates these capabilities to interpret and execute procedural texts. Realizing such systems is of great significance in that it leads to the realization of robotic assistants for human activities (Hamada et al., 2005; Hashimoto et al., 2008) and robots executing complicated procedures instead of humans (Bollini

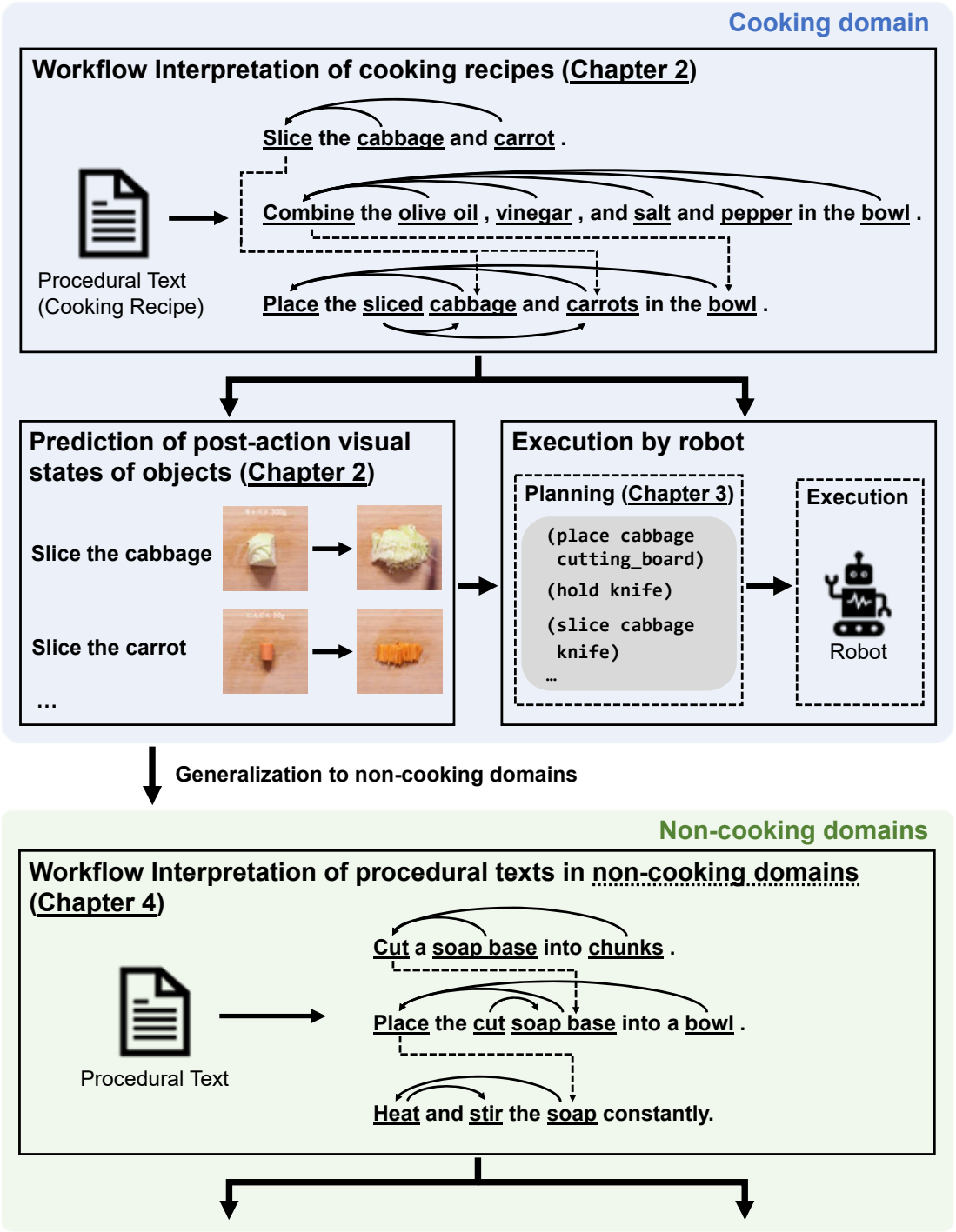

Figure 1.1: Illustration of our ultimate goal and position of each chapter in this thesis.

et al., 2013; Yoshikawa et al., 2023).

Towards the realization of this grand goal, this thesis summarizes three research works contributing to the realization of (i), (ii), and (iii) in the above capabilities, with a primary focus on the cooking domain. The first work tackles (i) and (ii) in the cooking domain, while the second work focuses on (iii) in the cooking domain. The third work addresses (i) to generalize the target domain from the cooking to non-cooking domains.

1.1 Background

This section briefly overviews three research topics that are particularly relevant to this thesis: workflow interpretation of procedural texts, tracking and recognition of object state changes, and conversion of linguistic instructions into actionable representations by robots.

1.1.1 Workflow interpretation of procedural texts

Procedural texts contain actions, objects, and other information about procedures. Each action is performed on an object and produces an intermediate product that will be used in later actions. This flow of actions forms a complex workflow throughout the text, making machine interpretation of the workflow of procedural texts challenging. In natural language processing (NLP), various approaches have been proposed to represent the workflow of procedural texts by particularly focusing on the cooking domain.

The early work by Momouchi (1980) proposed the PT-chart that represents the control structure of Japanese cooking recipes. Hamada et al. (2000) and Hamada et al. (2005) proposed to make flow graphs from cooking recipes through structural analysis using domain-specific constraints and knowledge. Tasse and Smith (2008) presented the SOUR CREAM system that aims to perform semantic parsing on cooking recipes. Mori et al. (2012) introduced the first machine learning approach for structural analysis of cooking recipes that extracts predicate-argument structures from a cooking

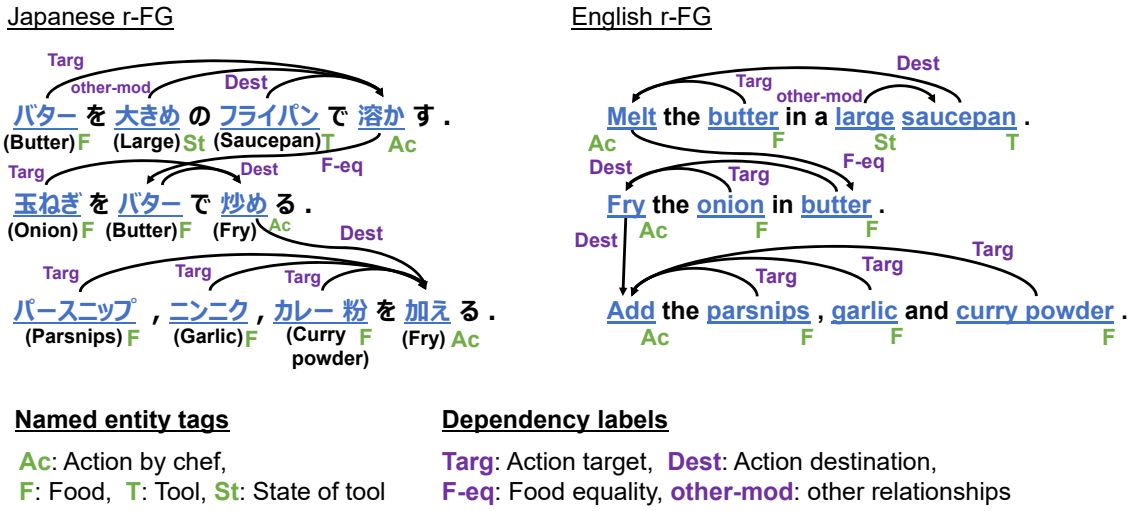

Figure 1.2: Example of a Japanese r-FG representation (Mori et al., 2014) (left) and an English r-FG representation (Yamakata et al., 2020) (right). Both recipes provide semantically the same steps.

step. After that, Mori et al. (2014) proposed a recipe flow graph (r-FG) representation for Japanese cooking recipes. On the other hand, Kiddon et al. (2015) built an unsupervised algorithm to interpret recipes as flow graphs, and Jermsurawong and Habash (2015) presented an approach to represent recipes with food ingredients as a dependency tree data structure.

Recipe flow graph (r-FG) Among these approaches, the r-FG (Mori et al., 2014) is particularly important to this thesis. As shown in the left of Figure 1.2, the r-FG by Mori et al. (2014) represents a Japanese cooking recipe as a directed acyclic graph (DAG). Here, nodes represent expressions relevant to procedures using tags, and edges represent the dependencies between the nodes using labels. Maeta et al. (2015) built an automatic prediction framework for the r-FG, in which an r-FG is predicted in two steps: node prediction and edge prediction. Yamakata et al. (2017, 2020) presented an r-FG representation for English recipes, which we refer to as an English r-FG, as shown in the right of Figure 1.2. Mori et al. (2014) provided an r-FG corpus of 266 Japanese recipes, while Yamakata et al. (2020) provided a corpus of 300 English recipes.

1.1.2 Tracking and recognizing the state changes of objects

Actions in instructional texts are performed on target objects, and they change the states of the objects to varying degrees. Recognizing the state changes and predicting the post-action state of an object are both essential abilities for intelligent agents. Realizing these abilities requires knowledge of the world and a reasoning capability about the action and object based on the current state.

In NLP, there exist works focusing on procedural texts. Early work by Henaff et al. (2017) and Ji et al. (2017) developed model architectures that maintain and update the state of entities. Henaff et al. (2017) evaluated their model by predicting the locations of agents in a grid world environment, while Ji et al. (2017) used the InScript corpus (Modi et al., 2017) for evaluation. Inspired by these works, Bosselut et al. (2018) proposed Neural Process Networks (NPNs) that predict states of ingredients after executing cooking steps. Nishimura et al. (2021) applied this idea to procedural text generation from videos. Dalvi et al. (2018) created the ProPara dataset for tracking state changes focusing on scientific processes (e.g., photosynthesis and erosion). Different from the works by Bosselut et al. (2018) and Dalvi et al. (2018), which considered a closed set of state changes, Tandon et al. (2020) constructed the OPENPI dataset for tracking an unrestricted set of state changes.

In computer vision, this direction of research goes back to early works on object attribute recognition (Ferrari and Zisserman, 2007; Farhadi et al., 2009; Parikh and Grauman, 2011). Fathi and Rehg (2013) modeled actions based on environment state changes and improved action recognition performance. Isola et al. (2015) focused on the state changes of objects for the first time and proposed to learn the changes from an image collection. Alayrac et al. (2017) proposed to discover object states from carefully curated videos based on actions on them. More recent works introduced two large-scale datasets, the Ego4D dataset (Grauman et al., 2022) and the ChangeIt dataset (Souček et al., 2022) dataset, which contain annotations of object state changes and state-modifying actions. Based on the ChangeIt dataset, Souček et al. (2023) proposed GenHowTo that takes images of an initial state and textual prompt and generates images of the final state after the action. On the other hand,

Saini et al. (2022) focused on the compositions of objects and proposed to learn disentangled features for object attributes. Saini et al. (2023) proposed the ChopNLearn dataset to recognize unseen compositions of objects focusing on the task of cutting vegetables.

1.1.3 Converting linguistic instructions into actionable representations

Research on converting linguistic instructions into representations interpretable and executable by robots has been tackled in NLP and robotics (Tellex et al., 2020).

The early work by MacMahon et al. (2006) considered a virtual indoor environment and proposed to convert route instructions into simple actions such as `Turn` and `Travel` via syntactic parsing on the instructions. Branavan et al. (2009) used reinforcement learning to realize mapping instructions to system commands in Windows troubleshooting guides. Chen and Mooney (2011) introduced a framework that learns to map navigation instructions into machine-interpretable plans. Matuszek et al. (2010) proposed to use a statistical machine translation system to learn to translate linguistic instructions into machine-interpretable ones. More recent works (Singh et al., 2023; Lin et al., 2023) used large language models (Brown et al., 2020; OpenAI, 2023) to convert linguistic instructions into plans executable by robots without learning.

In works targeted cooking recipes, Beetz et al. (2011) proposed to interpret pancake recipes on the web through reasoning about cooking actions. Bollini et al. (2013) presented the BakeBot that converts cooking recipes for baking cookies into plans of symbolic actions such as `(pour butter mbowl)` and `(mix mbowl)`.

1.2 Problems of Interest

1.2.1 Prediction of post-action visual states of objects in cooking recipes

The state of objects appearing in procedural texts dynamically changes by executing actions. Predicting the post-action states of objects based on the current states would help the robot execute the actions. For example, the predicted states can be used as the goal states of the actions. The prediction can take the form of text and images. Here, we consider prediction by image because such visual information could provide qualitative information such as color and shape, which are not described in text. In support of this, approaches that execute actions based on predicted visual states are gaining attention in the field of robotics (Nair et al., 2018; Kapelyukh et al., 2023). We tackle this problem in the cooking domain.

Realizing a model that predicts the post-action visual states of objects based on the current states requires visual annotations of pre-action and post-action states of objects. On the other hand, cooking is a sequential process, and objects would be subject to multiple actions throughout the procedure and integrated into the final dish. Thus, to capture this sequential nature, an annotation of the cooking workflow (e.g., by recipe flow graph (Mori et al., 2014)) is also required. Although previous work has proposed datasets with cooking workflow representation and visual annotation (Nishimura et al., 2020; Pan et al., 2020), a dataset with all of the above annotations has never been proposed. Therefore, we start by constructing a dataset to address this problem.

1.2.2 Converting steps to plans interpretable by robots

Executing a procedural text requires converting steps into plans interpretable by robots. The plans usually consist of sequences of symbolic actions such as `pick(a)` (pick a) or `place(a, b)` (place a on b). A single step can be converted into multiple symbolic actions. For example, a step of "slice the carrot and place them in a bowl." can be converted into a sequence of symbolic actions: `pick(carrot, left_hand)`

(pick up the carrot with a left hand), `place(carrot, cutting_board)` (place the carrot on the cutting board), `hold(knife, right_hand)` (hold a knife with a right hand), `slice(carrot, right_hand)` (slice the carrot with the knife), etc.

In recent years, works using large language models (LLMs) to convert linguistic instructions into plans directly have actively been addressed (Huang et al., 2022; Singh et al., 2023; Lin et al., 2023). These approaches are attractive in that they can utilize the linguistic knowledge of the LLMs and use the LLMs in a zero-shot manner by using a few-shot prompting technique. However, there are two issues with the approach. One is that they perform the conversion end-to-end, which lacks interpretability. Interpretability is a crucial factor for the continued development of a method. The other is that the obtained plans are not necessarily correct because LLMs do not have a mechanism to check the correctness of the plans. These issues would likely be more severe when considering executing the entire procedure. On the other hand, in the field of classical planning, a symbolic planner (Helmert, 2006) is used to find plans. The planner requires a problem description, which is a task specification file written in formal language, as input, and the obtained plans are guaranteed to be logically correct. The above two types of approaches have their own attractive strengths. Here, we attempt to develop a framework that combines the best of both words.

1.2.3 Workflow interpretation of procedural texts beyond the cooking domain

A recipe flow graph (r-FG) (Mori et al., 2014) is a representation that represents cooking recipes as flow graphs. The r-FG captures the dependencies of actions throughout the text, which is an essential feature for realizing robots that work on procedural texts. An automatic r-FG prediction framework by Maeta et al. (2015) predicts an r-FG in two steps: node prediction and edge prediction. Previous work has proposed r-FG corpora of Japanese recipes (Mori et al., 2014) and English recipes (Yamakata et al., 2017, 2020). More recently, Nishimura et al. (2020) introduced a visual annotation of r-FG.

While these developments have been made, the r-FG has only been applied to cooking recipes. Developing a generalized flow graph representation that handles non-cooking domains would have various benefits. For example, it enables the sharing of knowledge about procedures between domains. It is also expected to contribute to the realization of robust robot systems that execute procedural texts regardless of the domain. To this end, we attempt to extend the r-FG to represent procedural texts in non-cooking domains as flow graphs.

1.3 Contributions

The contributions of this thesis are summarized as follows. Figure 1.1 illustrates the relationship between these contributions.

1.3.1 Constructing the Visual Recipe Flow dataset

We have constructed the Visual Recipe Flow (VRF) dataset, a new multimodal dataset that aims to predict post-action visual states of objects. The VRF dataset consists of two types of annotations: the workflow representation of cooking recipes by r-FG and visual annotations of pre-action and post-action visual states of objects. The visual annotations are connected to cooking actions in the r-FG, which form a cross-modal relationship. We have measured inter-annotator agreements and confirmed that the VRF dataset consists of high-quality annotations. Through multimodal retrieval experiments, we have confirmed that both textual and visual annotations play an important role in predicting post-action visual states of objects.

1.3.2 Proposing Vision-Language Interpreter

We have proposed Vision-Language Interpreter (ViLaIn), a novel framework that converts linguistic instructions and scene observations into problem descriptions. ViLaIn drives a symbolic planner and finds valid plans based on the predicted descriptions. ViLaIn uses a few-shot prompting and predicts the descriptions from a few input/output examples without additional training. We have also proposed the

problem description generation (ProDG) dataset, a new dataset to evaluate the performance of ViLaIn with new evaluation metrics. We have experimentally confirmed that ViLaIn can generate syntactically correct problem descriptions and find valid plans with high accuracy.

1.3.3 Proposing a generalized flow graph representation

We have focused on wikiHow articles and proposed a wikiHow flow graph (w-FG) representation, an extension of the r-FG to represent procedural texts in non-cooking domains. We have created the w-FG corpus that covers 4 wikiHow top categories and contains 30 annotated articles for each domain. In experiments, we assumed a low-resource setting where the amount of available training data is small in the target domain and proposed using the existing r-FG corpus as additional training data. We have experimentally shown that domain adaptation from the r-FG to w-FG corpus significantly improved performance on both node prediction and edge prediction.

1.4 Thesis Outline

This thesis consists of five chapters as follows:

- In **Chapter 2**, we present the VRF dataset. This chapter first overviews the characteristics of the dataset. Then, the annotation standards, data collection, and annotation procedures are described in order. The annotation results, including the statistics and inter-annotator agreements, are introduced. After that, the multimodal retrieval experiments and their results are explained. Finally, possible applications of the VRF dataset are mentioned.

  In **Chapter 3**, we present ViLaIn and the ProDG dataset. This chapter first explains a problem statement and then introduces ViLaIn with the three modules that comprise it. Then, the ProDG dataset and evaluation metrics are then introduced. Finally, experimental results on the ProDG dataset are discussed.

  In **Chapter 4**, we present a w-FG representation and the w-FG corpus. This chapter first briefly reviews the r-FG and then introduces the w-FG. After that,

the w-FG corpus is then introduced with its data collection process, annotation procedure, and statistics. In experiments, results on node prediction and edge prediction are discussed. Finally, related work to this work is described.

- In **Chapter 5**, we conclude this thesis and discuss possible future directions of research.

Chapter 2

Visual Recipe Flow: A Dataset for Learning Visual State Changes of Objects with Recipe Flows

2.1 Introduction

Prediction is an innate ability of humans. In our daily lives, we make various predictions and act based on them. In cooking, for example, we first predict the state of foods after cooking actions and then act on the predictions. This requires (i) specific knowledge about actions and target objects and (ii) an interpretation of the workflow of the entire procedure. Therefore, building robots that follow cooking recipes requires realizing these two capabilities, as illustrated in Figure 2.1. In this example, the robot is predicting an observation after the second action while identifying the required food for the action.

Previous studies in this direction have annotated cooking procedures with visual annotations (Nishimura et al., 2020; Pan et al., 2020). (Nishimura et al., 2020) targeted pairs of images and cooking instructions and annotated the images with bounding boxes corresponding to expressions of actions and foods in the instructions. (Pan et al., 2020) annotated the cooking procedures with a frame sequence extracted from the attached cooking videos. Their datasets are meaningful in that they provide

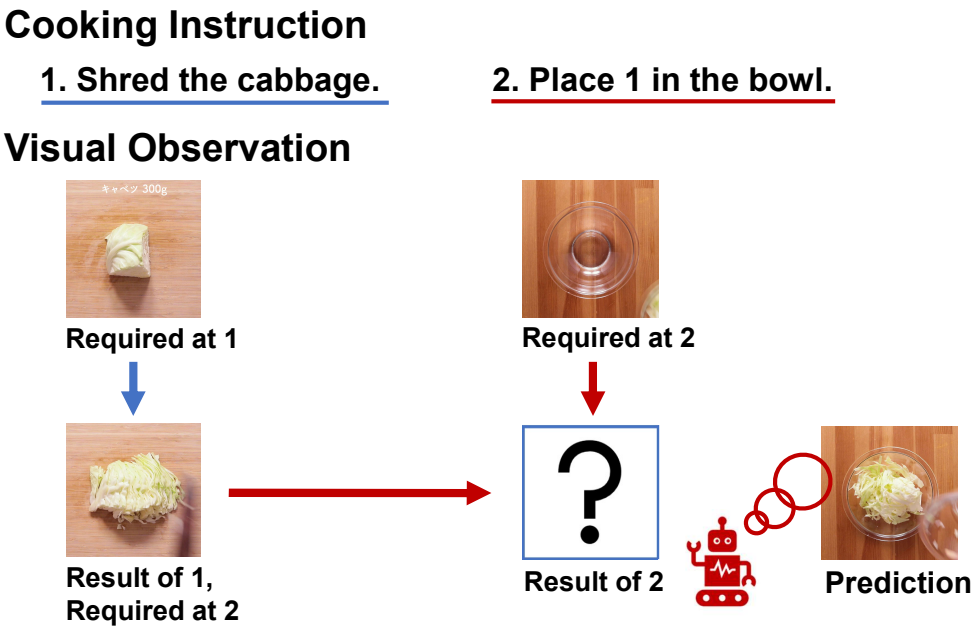

Figure 2.1: Example of predicting cooking action results by a cooking robot. The agent predicts the result of the second action "place."

cross-modal annotations by connecting visual information to cooking instructions. However, more fine-grained annotations at the action level rather than the instruction level are required for our goal. For example, an instruction "slice tomatoes and place them in a bowl." contains two cooking actions, "cutting" and "putting," but the annotations in the previous studies cannot handle them separately. Therefore, in order to predict the results of cooking actions, annotations at the cooking action level are necessary.

We propose the Visual Recipe Flow (VRF) dataset, a new dataset for predicting the post-action visual states of objects[1] in cooking recipes. As shown in Figure 2.2, the VRF dataset consists of (i) visual state changes of objects by cooking actions and (ii) representations of the entire recipe workflow. The visual state changes are represented as pairs of visual observations before and after cooking actions. The recipe workflow is represented as a recipe flow graph (r-FG) (Mori et al., 2014). Here, the observation pairs are associated with the cooking actions in r-FG, which allows

[1]In this chapter, an object refers to either a food or a tool.

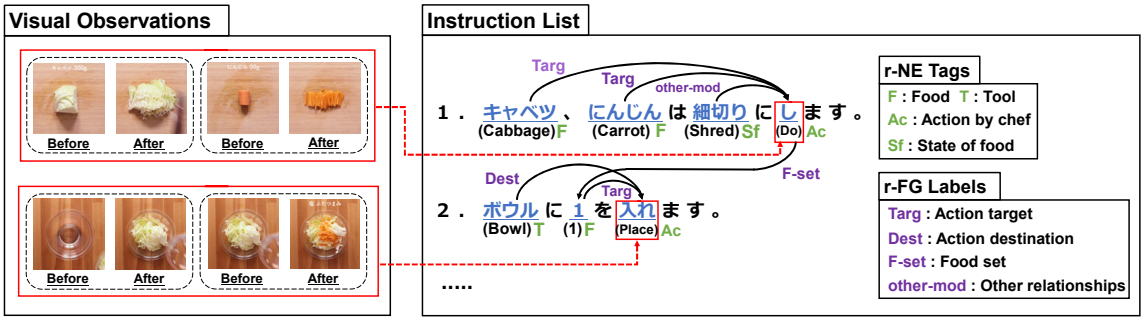

Figure 2.2: Example of annotation. The image pairs represent pre-action and post-action observations of objects. These image pairs are connected to cooking actions in the instruction list. In the instruction list, the dependencies of cooking actions are represented by r-FG.

us to handle cross-modal relationships between the real world and text. We provide our annotation and a script to construct the VRF dataset by downloading cooking recipes and videos from the website.[2]

Previous studies related to ours include tracking state changes of objects on texts (Dalvi et al., 2018; Bosselut et al., 2018; Tandon et al., 2020). These studies aim to understand procedural texts by predicting procedures' effects on objects as symbols in the real world. A recent work extended the framework (Bosselut et al., 2018) to handle visual information and applied it to video captioning (Nishimura et al., 2021). Our work follows this trend of research, and a key difference from these works is that ours focuses on predicting objects' states as images. Images are expected to provide richer information about the real world than symbols because they can handle information about objects' appearances (Isola et al., 2015; Zhang et al., 2021) that are difficult to describe in detail in text. In addition, the sequential nature of cooking would evaluate the document comprehension capability of large language models (Srivastava et al., 2022) and the learning capability of vision-language models in few-shot learning settings (Alayrac et al., 2022).

[2] https://github.com/kskshr/Visual-Recipe-Flow

2.2 Visual Recipe Flow Dataset

The Visual Recipe Flow (VRF) dataset represents the pre-action and post-action states of objects in a recipe using visual annotations. In the VRF dataset, expressions such as objects and cooking actions are identified using the recipe named entity (r-NE) (Mori et al., 2014). The recipe workflow is represented as dependencies between these r-NEs using a recipe flow graph (r-FG) (Mori et al., 2014). This section overviews the characteristics of the r-FG and visual annotation in the VRF dataset.

2.2.1 Recipe flow graph (r-FG)

The r-FG is defined as an acyclic directed graph (DAG) as shown in Figure 2.2. In r-FG, a node refers to an expression identified by r-NE, and an edge represents a dependency between two r-NEs with a label. One of the features of r-FG is that it can identify objects that are required for the actions. For example, in Figure 2.2, the object "1" targeted by the action "place" is the product of the first instruction, which is shredded cabbage and carrot, and this relationship is identified by tracing the edge of r-FG. In addition, the VRF dataset provides annotations of dependencies from the ingredient list to the cooking instructions, unlike the previous study (Mori et al., 2014; Yamakata et al., 2020). This feature enables us to consider the workflow of cooking recipes associating them with ingredient lists, similar to previous work by Jermsurawong and Habash (2015).

2.2.2 Visual annotation

The visual annotation represents the visual state changes of objects by cooking actions. The state changes are represented as pairs of images corresponding to the pre-action and post-action states. Here, each pair of images is associated with a cooking action expression in the r-FG, which forms cross-modal annotation. In the case that a cooking action targets multiple objects, a separate pair of images is provided for each object, as in Figure 2.2. This allows us to capture the state changes of objects even when a single action targets multiple objects.

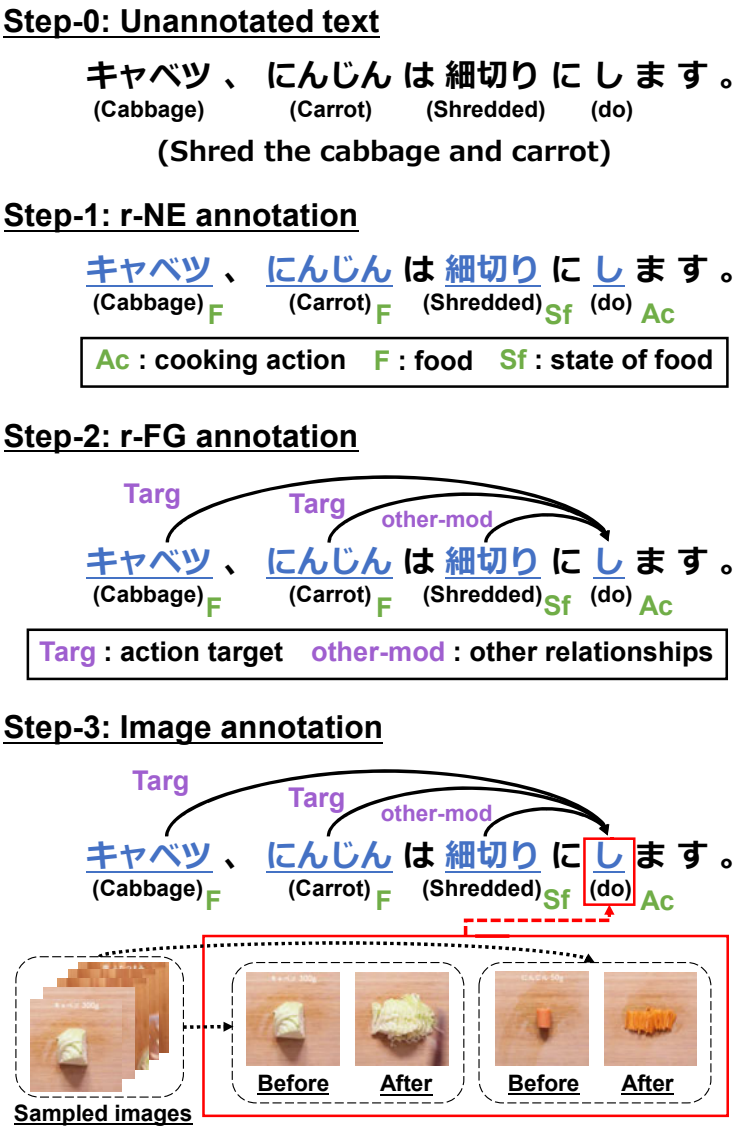

Figure 2.3: Illustration of annotation steps. Each step is performed based on the annotation result of the previous step.

2.3 Annotation standards

The VRF dataset consists of textual and visual annotations. The textual annotation is also divided into recipe named entity (r-NE) and recipe flow graph (r-FG). This section describes the annotation standards for these three types of annotations. Each recipe is assumed to be written in Japanese, to have an ingredient list and instruction

Tag	Meaning	Annotation examples	# of annotations
F	Foods	にんじん (carrot), 根本 (root), １ (1)	5,098
T	Tools	ボウル (bowl), ラップ (wrap)	758
D	Duration	5分程 (about 5 minutes), １０分 (10 minutes)	129
Q	Quantity	残り (the rest), 大さじ１ (1 tablespoon)	1,778
Ac	Action by chef	切 (cut), 入れ (put), 切り落と (cut off)	2,532
Af	Action by food	馴染 (blend), 取れ (remove), 沸騰 (boil)	353
Sf	State of food	一口大 (bite-size pieces), 薄切り (slices), 半分 (half)	971
St	State of tool	６００W (600W), 中火 (medium heat)	67
Total	—	—	11,686

Table 2.1: r-NE tags with their meaning and the number of annotations in the VRF dataset.

list, and to have an attached cooking video. Figure 2.3 illustrates an annotation example.

2.3.1 r-NE annotation

The r-NE annotation assigns r-NE tags to word sequences in the ingredient lists and instruction lists. Sentences are assumed to be segmented into words by using KyTea (Neubig et al., 2011), a Japanese word segmentation tool. A word is assigned with one tag at most. There are nine r-NE tags and their definitions are based on the previous work (Mori et al., 2014). Table 2.1 shows the list of r-NE tags. In the tags, Ac (action by chef), F (foods; F), and T (tools; T) are especially important in our study because they are directly related to cooking actions. For cooking actions that appeared in the ingredient lists, such as "boiled" and "sliced," we annotated with Sf (the state of foods) instead of Ac, assuming that such actions are already completed before the beginning of the cooking.

2.3.2 r-FG annotation

The r-FG annotation creates a DAG by treating the r-NEs as nodes and by annotating labeled directed edges between the nodes. There are 13 types of labels, and their definitions are based on the previous work (Maeta et al., 2015). Table 2.2 shows

Label	Meaning	Example (Starting node → Ending node)	# of annotations
Agent	Action agent	盛り付け (serve) → 完成 (complete), 味 (flavor) → 馴染(ませる) (season)	330
Targ	Action target	キャベツ (cabbage) → 切(る) (cut), 1 (1) → 入れ(る) (place)	2,961
Dest	Action destination	器 (plate) → 盛り付け(る) (serve), 耐熱ボウル (heatproof bowl) → 入れ(る) (place)	1,025
T-comp	Tool	電子レンジ (microwave) → 加熱(する) (heat), フォーク (fork) → 潰(す) (mash)	157
F-comp	Food	水 (water) → さら(す) (let), 塩コショウ (salt and pepper) → 調え(る) (season)	20
F-eq	Food equality	にんじん (carrot) → にんじん (carrot), 切(る) (cut) → 2 (2)	2,397
F-part-of	Food part-of	にんじん (carrot) → 皮 (skin), ミニトマト (mini tomato) → ヘタ (stem)	330
F-set	Food set	酢 (vinegar) → A (A), ドレッシング (dressing) → 材料 (ingredients)	987
T-eq	Tool equality	加熱 (head) → 耐熱ボウル (heat-resistant bowl), フライパン (pan) → フライパン (pan), ···	4
T-part-of	Tool part-of	—	0
A-eq	Action equality	な(り) (do) → 乳化(する) (emulsify)	1
V-tm	Head of clause for timing	馴染(んだら) (season) → 盛り付け(る) (serve), しんなり (soften) → 加熱(する) (heat), ···	112
other-mod	Other relationships	薄切り (slice) → し (do) 半分 (half) → 切(る) (cut)	2,967
Total	—	—	11,291

Table 2.2: r-FG labels with their meaning and the number of annotations in the VRF dataset.

the list of labels. In the labels, Targ (action target) and Dest (action direction) are especially important in this work because they describe the relationships among actions (Ac), foods (F), and tools (T). Figure 2.4 illustrates an annotation example of r-NE and r-FG for an ingredient list and instruction list.

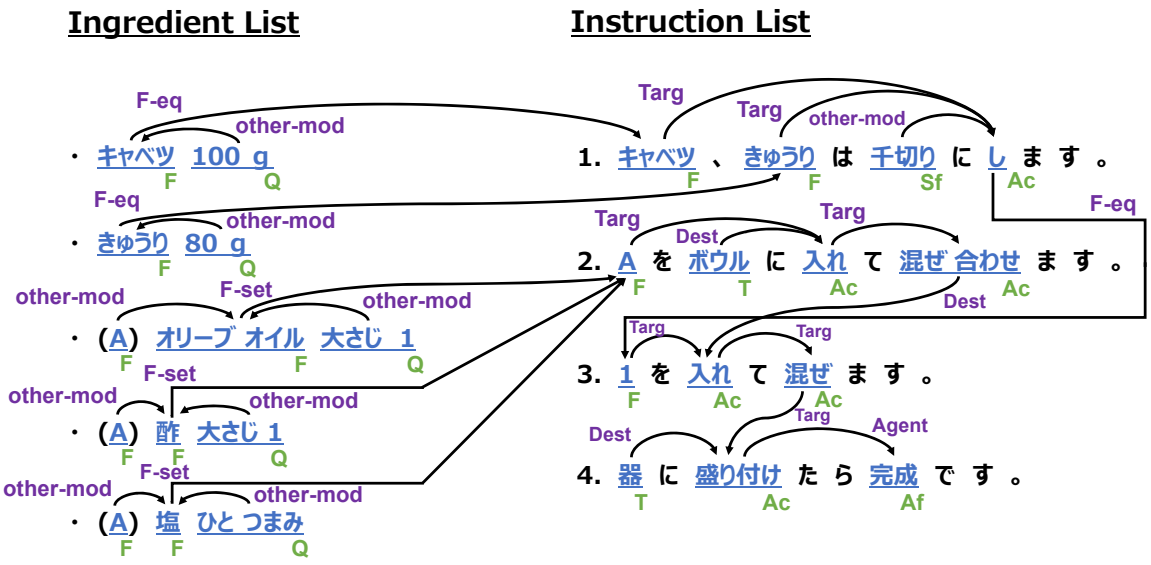

Figure 2.4: Example of r-NE and r-FG annotation for an ingredient and instruction list.

2.3.3 Visual annotation

The visual annotation assigns visual state changes of objects to cooking actions. The state changes are represented as pairs of images corresponding to pre-action and post-action states, and the images are extracted from attached cooking videos at 3 FPS. If a single cooking action targets multiple objects, each object is separately annotated with an image pair. If there are multiple image candidates for annotation, the one that best captures the target object is selected. In cases where the cooking action is not recorded in the video, or most of the objects in the images are covered by hands or tools, no annotation is performed, and the missing values are recorded instead. Therefore, it should be noted that not all state changes are annotated with two images.

2.4 Annotation results

This section first describes the data collection and annotation procedure. Then, the statistics of the dataset and its inter-annotator agreements are introduced. Finally, we

conduct multimodal retrieval experiments to evaluate the annotations of the dataset and discuss the results.

2.4.1 Annotation procedure

The dataset construction started by collecting recipes from scratch because recipes in the existing flow graph corpus (Mori et al., 2014) were not attached to cooking videos. We collected 200 recipes written in Japanese with cooking videos performed based on them from Kurashiru.[3] The cooking videos are taken by a fixed camera; thus, it is possible to annotate visual state changes of objects from a fixed viewpoint. As described in Section 2.1, the goal of this research is to realize robots that interpret cooking recipes as input and execute actions in the real world. Towards this goal, we targeted the category of salads and collected salad recipes in order to start with recipes, including relatively simple processes.

The annotation for the VRF dataset requires an annotator with experience in natural language processing annotation and cooking. Hence, we requested one such annotator to annotate all the collected recipes. The annotator training was performed by annotating 20 recipes randomly collected from the r-FG corpus (Mori et al., 2014). The training continued until the agreement with the ground-truth annotations of r-NE and r-FG exceeds 80%. The training for the visual annotation was conducted by using 50 recipes out of the total 200 recipes to be annotated, checking the results after every 10, 20, and 50 annotations, and giving detailed instructions when the annotation result differs from an annotation specification. During the annotator training, we revised the annotation specification as necessary.

In the previous study (Mori et al., 2014), the annotation was done by filling a spreadsheet. A potential problem with this way of annotation is that it is costly and may cause unexpected annotation errors due to manual work. Therefore, we developed an annotation tool that covers all three annotation steps required for our dataset. The implementation of the tool is already available on the web.[4] The annotation with this tool for all of the 200 recipes took 120 hours in total.

[3] `https://www.kurashiru.com`, accessed on December 14, 2021.
[4] `https://github.com/kskshr/Flow-Graph-Annotation-Tool`

Annotated image		# of visual state changes of objects
Pre-action state	Post-action state	
		597
✓		72
	✓	485
✓	✓	2,551
Total		3,705

Table 2.3: Statistics of the visual annotation.

Annotation	Precision	Recall	F1
r-NE	97.93	98.88	98.40
r-FG	86.18	86.04	86.11
Image	75.13	70.60	72.80

Table 2.4: Inter-annotatotor agreements.

2.4.2 Statistics

There were 1,701 ingredients and 1,077 instructions in the collected data. In addition, they contained 89 types of cooking action verbs and 275 types of food ingredient expressions. The r-NE annotation results in 11,686 r-NE tags, as shown in Table 2.1. The table also shows that Ac, F, and T appears frequently in the dataset, accounting for 81% of the total. The r-FG annotation results in 11,291 r-FG labels, as shown in Table 2.2. The table also shows that Targ, Dest, F-eq, and other-mod are frequently appeared in the dataset, accounting for 83% of the total.

The visual annotation results are shown in Table 2.3. This annotation step results in 3,705 visual state changes in total. The table shows that 2,551 state changes were annotated with both pre-action and post-action states, 485 state changes were annotated with post-action states only, and 72 state changes were annotated with pre-action states only. The remaining 597 state changes could not be annotated with any image. The annotation used 5,659 images in total (3,824 without duplicates). The reason for the duplicates is that performing multiple actions on the same object in succession can cause a case where the same image is used for the state after the action and the state before the next action.

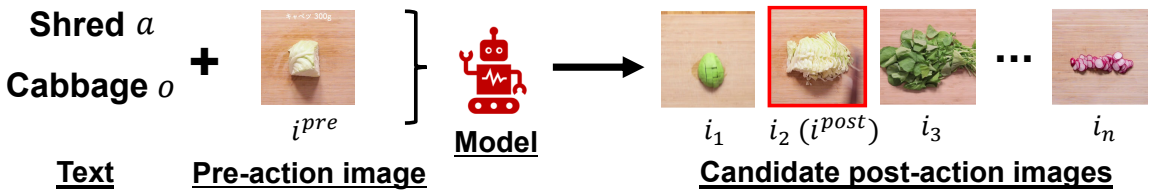

Figure 2.5: Illustration of the retrieval task. The model retrieves the correct post-action image i^{post} from the action verb a, object o, and pre-action image i^{pre}. In this example, the image surrounded by the red square is the correct image.

2.4.3 Annotation quality

To investigate the quality of the annotations, we requested another annotator with experience in natural language processing annotation and cooking to annotate 10 recipes (5% of the total) randomly collected from the dataset.[5] The annotators were trained using the same procedure as in Section 2.4.1 for all three annotation steps.

Table 2.4 shows the inter-annotator agreements between the annotations obtained in Section 2.4.1 and this subsection. Here, we consider the annotations in Section 2.4.1 as the ground-truth data and calculated the accuracy, recall, and F1 value, following (Yamakata et al., 2020). The r-NE and r-FG annotations obtained very high agreement rates of 98.40% and 86.11%, respectively. As for the image annotations, the agreement rate was 72.80%, which is lower than those of the previous annotation steps. However, given the influence of the annotation errors from the previous stages and the fact that multiple images can be strong candidates during the visual annotation, we consider that this score would be reasonable.

2.4.4 Experiments

Task definition. We evaluated the annotations of the VRF dataset by considering a new multimodal retrieval task and by conducting experiments. Given the textual expression of a verb a and an object o and a pre-action image i^{pre} that corresponds to the text, this task aims to select the corresponding post-action image i^{post} from a set of candidate images $i_1, i_2, \cdots, i_n$. Figure 2.5 shows an example.

[5]The extracted 10 recipes contained 623 r-NE tags, 616 r-FG labels, and 199 visual state changes.

Model architecture. We solve this task by retrieval via a shared embedding space (Miech et al., 2019). Our method learns two neural networks; one for encoding the post-action image candidates into vectors on the embedding space and the other for estimating a vector corresponding to i^{post} from a, o, and i^{pre}. The estimated vector is used to retrieve the vector for i^{post}. The following describes these two neural networks.

The first network estimates the vector corresponding to i^{post} from a, o, and i^{pre}. a and o are encoded into d_v-dimensional vectors h_a and h_o using Bidirectional Long Short-Term Memory (BiLSTM) (Lample et al., 2016). h_a and h_o are obtained by encoding the whole recipe by BiLSTM and by selecting the vectors corresponding to a and o. When a or o consists of multiple tokens, the vector is obtained as an average of those vectors. i^{pre} is encoded into a d_t-dimensional vector h_i^{pre} using a pre-trained convolutional neural network (CNN) as follows:

$$h_i^{\text{pre}} = W_1^{\text{pre}}\text{CNN}(i^{\text{pre}}) + b_1^{\text{pre}},$$

where $W_1^{\text{pre}} \in \mathbb{R}^{d_t \times d_i}$ and $b_1^{\text{pre}} \in \mathbb{R}^{d_t}$ are learnable parameters. Based on h_a, h_o, and h_i^{pre}, we compute $\tilde{h}^{\text{pre}}$ as follows:

$$\tilde{h}^{\text{pre}} = W_3^{\text{pre}}(\text{ReLU}(W_2^{\text{pre}}[h_a; h_o; h_i^{\text{pre}}] + b_2^{\text{pre}})) + b_3^{\text{pre}},$$

where ; represents to the concatenation, $W_2^{\text{pre}} \in \mathbb{R}^{3d_t \times 3d_t}$, $W_3^{\text{pre}} \in \mathbb{R}^{d_t \times 3d_t}$, $b_2^{\text{pre}} \in \mathbb{R}^{3d_t}$, $b_3^{\text{pre}} \in \mathbb{R}^{d_t}$ are all learnable. Finally, we obtain a vector $\hat{h}^{\text{pre}}$ that corresponds to the correct post-action image on the shared embedding space as follows:

$$\hat{h}^{\text{pre}} = \frac{f(\tilde{h}^{\text{pre}})}{||f(\tilde{h}^{\text{pre}})||_2},$$

where

$$f(h) = (W_4^{\text{pre}}h + b_4^{\text{pre}}) \circ \sigma(W_5^{\text{pre}}(W_4^{\text{pre}}h + b_4^{\text{pre}}) + b_5^{\text{pre}}),$$

following the previous work (Miech et al., 2018). Here, σ and $\circ$ refers to the sigmoid function and element-wise multiplication, respectively. $W_4^{\text{pre}} \in \mathbb{R}^{d_e \times d_t}$, $W_5^{\text{pre}} \in \mathbb{R}^{d_e \times d_e}$,

$b_4^{\text{pre}}, b_5^{\text{pre}} \in \mathbb{R}^{d_e}$ are also learnable.

The second network encodes post-action image candidates, including i^{post}, into d_t-dimensional vectors on the shared embedding space. i^{post} is converted into a vector using a pre-trained CNN as follows:

$$\tilde{h}_i^{\text{post}} = W_2^{\text{post}}(\text{ReLU}(W_1^{\text{post}}\text{CNN}(i^{\text{post}}) + b_1^{\text{post}})) + b_2^{\text{post}},$$

where $W_1^{\text{post}}, W_2^{\text{post}} \in \mathbb{R}^{d_i \times d_i}$, and $b_1^{\text{post}}, b_2^{\text{post}} \in \mathbb{R}^{d_i}$ are learnable. The vector on the shared embedding space $\hat{h}^{\text{post}}$ is obtained as:

$$\hat{h}^{\text{post}} = \frac{g(\tilde{h}^{\text{post}})}{||g(\tilde{h}^{\text{post}})||_2},$$

where

$$g(h) = (W_3^{\text{post}}h + b_3^{\text{post}}) \circ \sigma(W_4^{\text{post}}(W_3^{\text{post}}h + b_3^{\text{post}}) + b_4^{\text{post}}),$$

where $W_3^{\text{post}} \in \mathbb{R}^{d_e \times d_i}$, $W_4^{\text{post}} \in \mathbb{R}^{d_e \times d_e}$, and $b_3^{\text{post}}, b_4{}^{textpost} \in \mathbb{R}^{d_e}$ are also learnable. Other post-action candidate images are similarly encoded into the embedding space through the above computations.

Loss function. The training of the two networks is performed by minimizing the following triplet loss (Balntas et al., 2016):

$$\mathcal{L} = \frac{1}{n}\sum_{i=1}^{n} \{\max(D_{i,i} - D_{i,j} + \delta, 0) + \max(D_{i,i} - D_{k,i} + \delta, 0)\}, \tag{2.1}$$

where $D_{i,j}$ represents the distance between $\hat{h}_i^{\text{pre}}$ and $\hat{h}_j^{\text{post}}$) calculated as:

$$D(\hat{h}^{\text{pre}}, \hat{h}^{\text{post}}) = ||\hat{h}^{\text{pre}} - \hat{h}^{\text{post}}||_2.$$

δ refers to the margin. $D_{i,i}$ denotes the distance between the i-th positive example vectors in a minibatch. $D_{i,j}$ denotes the distance when the post-action image vector is replaced by a negative example h_j^{post}, and $D_{k,i}$ denotes the distance when the pre-action image vector is replaced by a negative example h_k^{pre}. The model parameters

are optimized to make the difference between $D_{i,i}$ and $D_{i,j}$, $D_{k,i}$ are greater than δ. In the experiments, we sample negative examples randomly from the minibatch.

Data splits. We used 2,551 visual state changes of objects that both pre-action and post-action states were annotated. The total 200 recipes were divided into 10 splits, 1 split (20 recipe) was used as the test data, and the remaining 9 splits (180 recipe in total) were merged and used as the training data. By changing the split corresponding to the test data, we realized the 10-fold cross-validation. In the above splits, the average number of training and testing examples were 2296.8 and 255.2, respectively.

The pre-action and post-action images that are annotated for the visual state changes were used as i^{pre} and i^{post}. The cooking action a corresponding to the state change is identified from the annotation because all state changes are associated with a certain cooking action of Ac, as explained in Section 2.2. The object o targeted by a is automatically identified by tracing the edge(s) of the r-FG. We used the object with F that can be identified by tracing edges of Targ or Dest from the node of Ac and are closest to the node as o.

Model parameters. We used a 1-layer BiLSTM of 256 dimensions for encoding recipe texts. The number of dimensions was set to $(d_v, d_t, d_i, d_e) = (496, 512, 2048, 128)$. The pre-trained ResNet-152 (He et al., 2016) was used as the CNN by freezing its parameters. All the images were converted into 2048-dimensional feature vectors by using the CNN.

Optimization. AdamW (Loshchilov and Hutter, 2019) was used as the optimization method with an initial learning rate of 1.0×10^{-5}. Models were trained for 350 epochs, and each training step optimizes the parameters using a mini-batch of 4 recipes. The δ of Equation 2.1 was determined as 0.1 from a preliminary experiment on the validation data.

Evaluation metrics. In the evaluation step, the model calculates the similarity between $\hat{h}^{\text{pre}}$ and the vectors of candidate post-action images on the embedding space,

including $\hat{h}^{\text{post}}$, by cosine similarity. The post-action image vectors are then sorted by the similarity in descending order. The evaluation is performed by using the rank of the correct post-action image vector in the order. For this purpose, Recall@5 (R@5) and Median rank (MedR) were used as evaluation metrics. R@5 calculates the ratio of correct post-action image vectors appearing in the top five ranks, and MedR computes the rank of correct image vectors among the candidates, respectively. These metrics have been used in previous studies to evaluate model performance on cross-modal retrieval tasks (Socher et al., 2014; Salvador et al., 2017; Miech et al., 2019).

Experimental results. To investigate the importance of the textual and visual information for predicting the state changes of objects, we conducted experiments by training and testing the model without using the textual information, the pre-action image, or both. This is done by using zero vectors as h_a, h_o, or h_i^{pre}.

Table 2.5 shows the experimental results. The first line represents a random baseline that does not use either textual or visual information. The comparison between the second and third lines shows that the use of visual information is more effective than the use of textual information, showing 33.77 of R@5 and 12.60 of MedR. The comparison between these results and the fourth line shows that the use of both textual and visual information further improves the performance by achieving 37.01 of R@5 and 10.40 of MeR. These results indicate the following two findings. First, the visual modality provides stronger information than the textual modality for improving performance. Second, using both modalities achieves the best result, demonstrating that the textual and visual information provide different cues to solve the task.

Discussion. The remaining issues in this task are (i) cases in which little visual state changes occurred by the action and (ii) cases in which analyzing the history of actions is required to predict the post-action image for the current action. An instruction “add ingredients for dressing to a bowl.” exemplifies (i). In this example, the ingredients, such as “oil” and “salt” change little in appearance before and after

Input		R@5 (↑)	MedR (↓)
Text (a and o)	Pre-action image (i^{pre})		
		2.37	149.00
✓		21.24	26.70
	✓	33.77	12.60
✓	✓	37.01	10.40

Table 2.5: Results . The checkmark symbol ✓indicates the used input for both training and evaluation.

the action. In fact, we have seen that the model struggles to identify the correct post-action images for these cases. A possible solution for this is to prepare appearance information of the food ingredients beforehand and use them when calculating $\hat{h}^{\text{pre}}$ to consider possible visual state changes.

Cases in which the object expressions refer to the previously conducted action expressions exemplify (ii). For example, "1" in an instruction "add 1 to the bowl." refers to the result of the first instruction, and "cut" in an instruction "cut the carrot and place them in the bowl." is an action but is also the object of the next action "place" in r-FG. A possible solution to handle these cases is to consider the history of actions by tracing the edges of an r-FG. In this case, tracing the edges along with r-NE information enables the model to identify the role of the nodes in the recipe. Another solution is to track food state changes through cooking actions using a method such as Neural Process Networks (Bosselut et al., 2018; Nishimura et al., 2021).

In addition to these issues, further performance improvement could be achieved by using information from r-NE tags and r-FG labels that are not used in this experiment. For example, the r-NE tag of Sf provides the state of foods or actions, and the r-FG label of other-mod bridges such expressions and actions. Since these tags and labels provide finer information about the actions, using them is expected to further improve the performance.

2.5 Application

This section describe possible applications of the VRF dataset.

2.5.1 Multimodal commonsense reasoning

Reasoning about procedures in procedures texts with multimodal information is one of the recent trends (Zellers et al., 2019a; Yagcioglu et al., 2018; Alikhani et al., 2019). In the cooking domain, previous work (Bosselut et al., 2018) proposed to simulate state changes of food ingredients by cooking actions at the step level, and a more recent work (Nishimura et al., 2021) applied this idea to video captioning. In this direction of research, our dataset can be used to track the finer state changes of food ingredients at the action level than the instruction level.

2.5.2 Procedural text generation

The generation of procedures from images or videos is an essential direction of research to improve the reproducibility of work Ushiku et al. (2017); Nishimura et al. (2019, 2021). Properly reproducing the procedures requires the generated procedures to be consistent throughout the process. The r-FG expresses the dependencies of actions and objects in the entire recipe. Thus, we consider that generating procedural texts while predicting the r-FG on the currently generated procedure would prevent inconsistent text generation and improve performance in the task.

2.6 Conclusion

This chapter has proposed the Visual Recipe Flow dataset to address the problem of predicting the post-action visual states of objects in cooking recipes. We have described the data collection and annotation procedure and have investigated the quality of the dataset by measuring the inter-annotator agreements. To evaluate the importance of textual and visual information for the task, we have proposed a new multimodal retrieval task and have conducted experiments. Finally, possible applications of this dataset have explained.

Chapter 3

Vision-Language Interpreter for Robot Task Planning

3.1 Introduction

Natural language is a prospective interface for non-experts to instruct robots intuitively (Hatori et al., 2018; Tellex et al., 2020; Liang et al., 2023). Earlier studies have used recurrent neural networks (Hochreiter and Schmidhuber, 1997; Cho et al., 2014) to map abstract linguistic instructions to representations for robots (Arumugam et al., 2017; Hatori et al., 2018; Paxton et al., 2019). Here, the linguistic instructions represent desired goal conditions. More recent studies use large language models (LLMs) (Touvron et al., 2023; OpenAI, 2023; Anil et al., 2023) to directly generate robot plans from the instructions (Huang et al., 2022; Raman et al., 2022; Lin et al., 2023; Singh et al., 2023). These language-guided planners utilize few-shot prompting to solve tasks without training (Brown et al., 2020). The plans are a sequence of discrete symbolic actions (e.g., `pick(a)` and `place(a, b)`) that complete the task. We aim to strengthen the language-guided planners in terms of the improvement of interpretability.[1] Interpretability is essential to gain the trust of the user and provide

[1]We define interpretability as a mechanism to provide insights into the inner workings of the system.

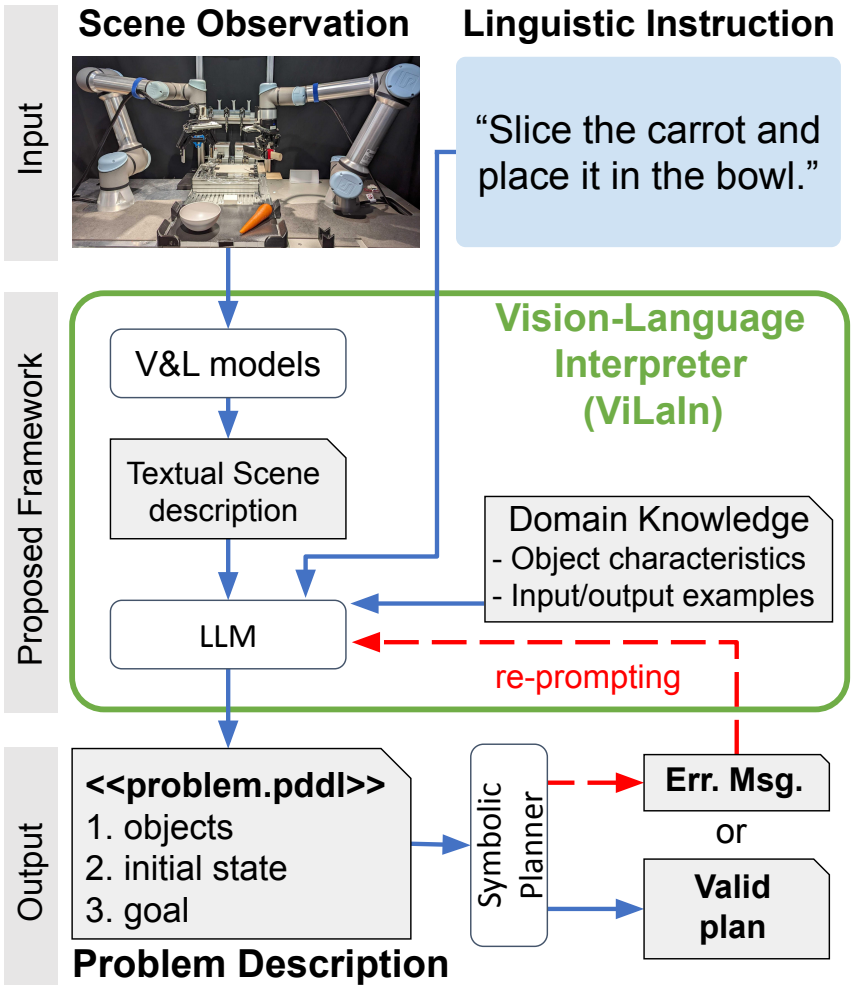

Figure 3.1: Overview of our approach. The vision-language interpreter (ViLaIn) generates a problem description from a linguistic instruction and scene observation. The symbolic planner finds an optimal plan from the generated problem description.

insights into the robot's decision-making process (Gilpin et al., 2018). For example, the identification of failure causes through interpretation leads to continuous improvement of overall performance.

Robot task planning has traditionally been solved using symbolic planning (Karpas and Magazzeni, 2020). Modern symbolic planners use the Planning Domain Definition Language (PDDL) to describe planning problems. In PDDL, a planning problem is defined in two parts: the *domain* that defines the state of variables and actions,

and a *problem description* (PD) that defines the objects of interest, their initial state, and the desired goal state (Haslum et al., 2019; Fox and Long, 2003). The domain and problem are inputs to the planner to find an optimal plan, a sequence of symbolic actions.

Symbolic planners offer several benefits. The domain and problem descriptions are human-readable, especially when variable names are chosen intuitively. Moreover, the obtained plans are guaranteed to be logically correct. Considering these advantages, combining symbolic planning and language-guided planning is a promising research direction to realize interpretable robots. To that end, we proposed generating the PDs from natural language instructions. Since the linguistic instructions only represent the goal conditions, additional information about the environment is required to generate the initial state (e.g., an image representing the current environment). We refer to this additional information as *scene observations*.

We tackle the *multimodal planning problem specification* task, a new task for transforming linguistic instructions and scene observations into logically and semantically correct PDs. The PDs have to be executable by the symbolic planners. This chapter investigates how accurately we can generate such PDs with a state-of-the-art LLM (OpenAI, 2023) and vision-language model (Liu et al., 2023a; Li et al., 2023) without additional training. We propose a Vision-Language Interpreter (ViLaIn), a new framework to solve the PD generation task, illustrated in Figure 3.1. ViLaIn consists of three modules that generate each part of the PDs. The complete PD is assembled by concatenating these parts. Furthermore, ViLaIn can refine the generated PDs via error feedback from the symbolic planner. The planner uses a pair of the generated PD and the domain description to find a plan. We use the state-of-the-art symbolic planner called Fast Downward (Helmert, 2006) throughout this chapter.

To evaluate ViLaIn, we introduce a novel dataset called the problem description generation (ProDG) dataset. The ProDG dataset consists of linguistic instructions, scene observations, and domain and problem descriptions. The descriptions are written in PDDL (Fox and Long, 2003). This dataset covers three domains: cooking as a practical robot domain, and the blocks world and the tower of Hanoi as classical planning domains. We propose four new evaluation metrics to evaluate ViLaIn from

Approach	Input other than linguistic instruction	Output
Huang et al. Huang et al. (2022)	—	Symbolic action
Raman et al. Raman et al. (2022)	—	Symbolic action
Text2Motion Lin et al. (2023)	PDDL scene desc.	Symbolic action
SayCan Ahn et al. (2022)	Image	Pre-defined skill
RT-2 Brohan et al. (2023)	Image	Low-level action
ProgPrompt Singh et al. (2023)	—	Program code
Code as Policies Liang et al. (2023)	Image	Program code
LLM+P Liu et al. (2023b)	Linguistic scene desc.	Problem desc.
ViLaIn (ours)	Image	Problem desc.

Table 3.1: Differences between previous studies and ours.

multiple perspectives.

The main contributions of this work are three-fold:

- Multimodal planning problem specification, a new task to bridge the language-guided planning and symbolic planners with scene observations.
- Vision-Language Interpreter (ViLaIn), a new framework consisting of a state-of-the-art LLM and vision-language model. ViLaIn can refine erroneous PDs by using error messages from the symbolic planner.
- The problem description generation (ProDG) dataset, a new dataset that covers three domains: the cooking domain, the blocks world, and the tower of Hanoi. The dataset comes with new metrics that evaluate ViLaIn from multiple perspectives.

3.2 Related work

This section describes previous work on language-guided planning, symbolic planning, and scene recognition in computer vision. Table 3.1 summarizes the difference between several studies mentioned here and ViLaIn.

3.2.1 Planning from natural language

Task planning from natural language has been actively studied (Huang et al., 2022; Ahn et al., 2022; Liu et al., 2023a). Converting linguistic instructions into symbolic actions via neural networks is a typical approach (Paxton et al., 2019; Sharma et al., 2022). More recent studies (Huang et al., 2022; Raman et al., 2022; Lin et al., 2023; Singh et al., 2023) use LLMs and directly generate plans with few-shot prompting (Brown et al., 2020). However, these language-guided planners have two issues. First, their systems hide the inner workings by generating plans end-to-end. Second, the obtained plans are not guaranteed to be logically correct. ViLaIn resolves these issues by converting instructions into human-readable PDs and driving symbolic planners to find plans with the generated PDs. A recent study uses LLMs to convert linguistic instructions and images into programs to complete robot tasks (Liang et al., 2023). PDs describe tasks more specifically, and their logical correctness is automatically verifiable. In other words, ViLaIn has the potential to deliver validated machine-readable information to other language-guided planners as an auxiliary input.

More recent studies have used LLMs to convert natural language inputs to PDs (Liu et al., 2023b; Xie et al., 2023). However, one study (Liu et al., 2023b) assumes that scene descriptions (the objects and initial state) are provided in natural language, which is not practical for real applications. Another work (Xie et al., 2023) focuses on only generating the goal specifications. Contrary to these studies, ViLaIn uses images for scene descriptions and generates the whole PDs, including the objects and initial states.

3.2.2 Symbolic planning with PDDL

Symbolic planning (automated planning) has been used to solve robotic tasks (Karpas and Magazzeni, 2020). Symbolic planners (Bonet and Geffner, 2001; Helmert, 2006) use domain and problem descriptions to find plans, which are sequences of (symbolic) actions that alter the environment from its initial state to a goal state. The descriptions are written in formal languages, such as PDDL (Fox and Long, 2003) and

PDDLStream (Garrett et al., 2020). Robots execute low-level actions based on the found high-level plans of PDDL (Ahmadzadeh et al., 2015; Wang et al., 2021; Silver et al., 2021). This framework enables robots to solve various problems but assumes a preparation of corresponding PD for each problem. ViLaIn is designed to collaborate with those PDDL-based planning frameworks by translating linguistic instructions into PDs.

3.2.3 Scene recognition for planning problem specification

The generation of the objects and initial state in PD is related to research in computer vision. This section briefly overviews such previous work.

The object part of PDs lists objects required for the task. This work generates the objects from scene observations. This can be viewed as object detection in computer vision. Classical object detectors (Ren et al., 2015; Redmon et al., 2016) have been developed focusing on a fixed number of classes (e.g., person and dog). However, our task handles objects not included in the classes. Hence, we use an open-vocabulary object detector (Zareian et al., 2021; Liu et al., 2023a). These detectors have recently gained attention because they can detect arbitrary objects using text queries.

The *initial state* represents object relationships and their states. Detecting such scene descriptions from images has been addressed on visual relationship detection (Lu et al., 2016; Inayoshi et al., 2020) or scene graph generation (Xu et al., 2017; Yang et al., 2022). Previous work trained a model with PDDL predicates and demonstrated it in real robot domains (Migimatsu and Bohg, 2022). We use a state-of-the-art LLM and vision-language model to generate the initial state.

3.3 Problem statement

We focus on multimodal planning problem specification, a new task for bridging language-guided planning and symbolic planning. The input is a quadruple (L, S, D_D, D_K); a linguistic instruction L, a scene observation S, a domain description D_D, and domain knowledge D_K. L is a sequence of words describing the task. S is an RGB

image describing the initial state of the environment. D_D defines parts common to all problems: object types (e.g., `location` and `tool`), predicates (e.g., `at` and `clear`), and symbolic actions (e.g., `slice` and `pick`). D_K supports D_D by providing more specific information on each problem, such as object characteristics (e.g., the cutting board is round, the counter is black) and actual input/output examples. Note that the examples in D_K use the object types and predicates defined in D_D.

The output is a PD P consisting of (O, I, G): the objects O, the initial state I, and the goal specification G. O consists of objects required for the task completion (e.g., *carrot* and *knife*). I consists of a set of propositions that represent the initial state of the environment (e.g., `(at carrot counter)`). A proposition is formed by providing a predicate with arguments. For example, providing a predicate `(at ?a1 ?a2)` with (`a1`, `a2`) = (`carrot`, `cutting_board`) forms a proposition `(at carrot cutting_board)` meaning "the carrot is at the cutting board." G consists of a set of propositions that represent the desired goal condition of the environment. For example, `(and (at carrot bowl) (is-sliced carrot))` represents the goal condition that "the carrot should be sliced and should be at the bowl." P and D_D are written in PDDL (Fox and Long, 2003), following previous work (Liu et al., 2023b; Xie et al., 2023). We refer to O, I, or G with PDDL (e.g., the PDDL objects). The goal of this task is obtaining a function $M : (L, S, D_D, D_K) \rightarrow (O, I, G)$. P must be machine-readable and executable by the symbolic planner.

3.4 Vision-Language Interpreter

ViLaIn consists of three modules: the object estimator, the initial state estimator, and the goal estimator. We describe these modules in this section.

3.4.1 Object Estimator

The PDDL objects O list objects of interest in the scene observations S. However, the observed objects vary greatly from domain to domain. Further, it must recognize various objects that classical object detectors cannot handle. For this reason, we

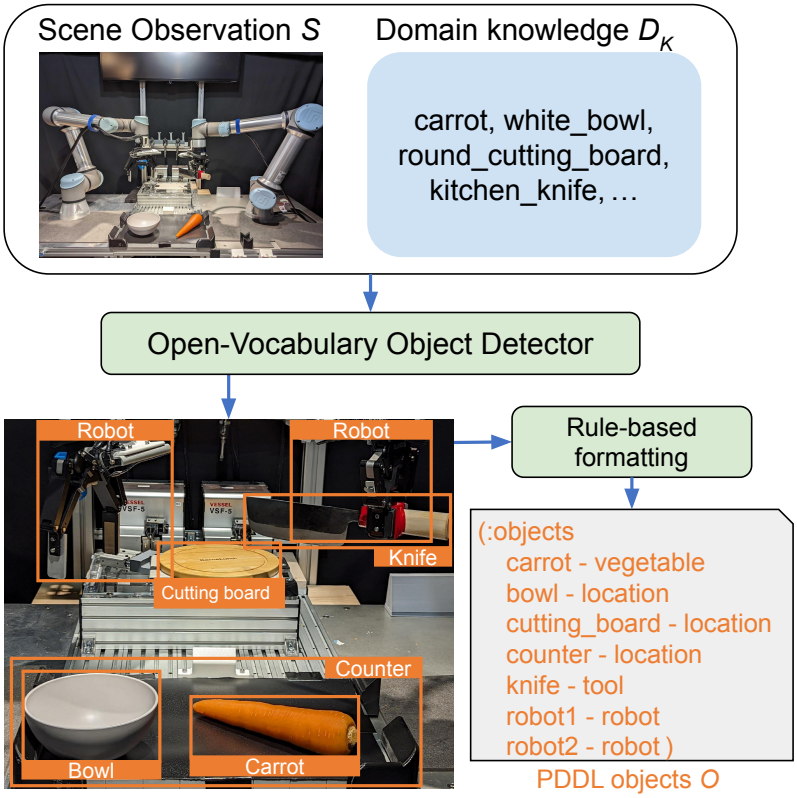

Figure 3.2: The open-vocabulary object detector detects objects from the observation. The text query is provided by the domain knowledge. The detected objects are converted into a PDDL format in a rule-based way.

use Grounding-DINO (Liu et al., 2023a), a state-of-the-art open-vocabulary object detector. Figure 3.2 illustrates the estimator. We assume that the list of objects for the task is known. The object list can be used as the text query. However, we found from preliminary experiments that simply using the object list fails to detect several objects. To address this issue, we elaborate the query using the domain knowledge (e.g., "cutting board" $\rightarrow$ "round cutting board" and "knife" $\rightarrow$ "kitchen knife"). In our setting, these elaborated queries are included in the domain knowledge D_K. The detected objects are converted into a PDDL format by rules.

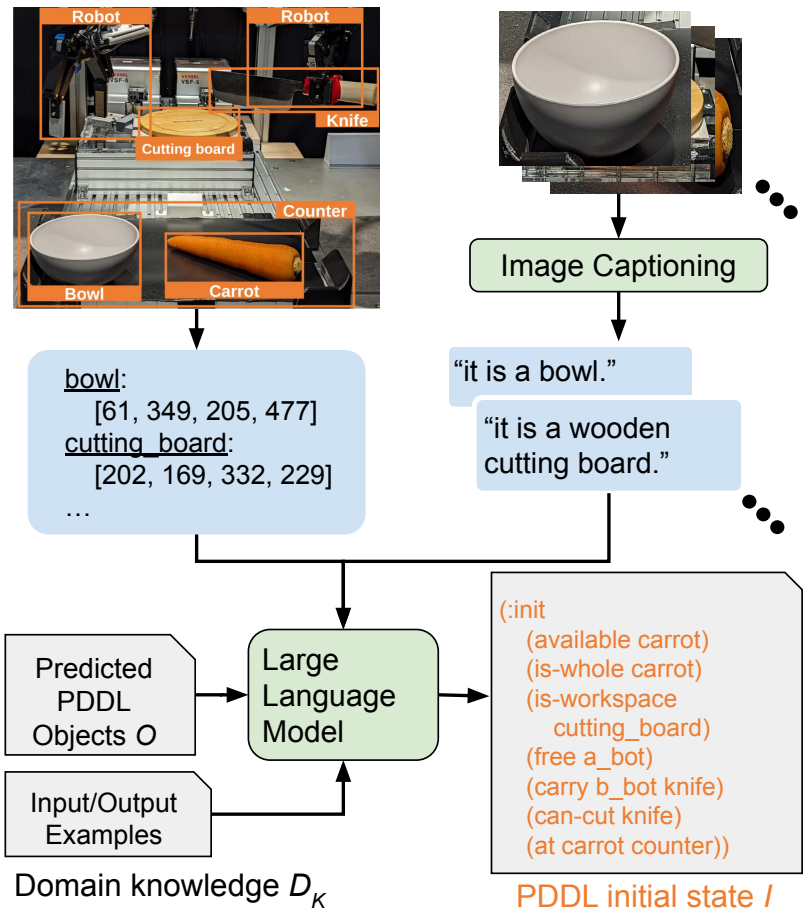

Figure 3.3: The captioning model generates captions for each object. The LLM generates the PDDL initial state from the bounding boxes and the captions using few-shot prompting.

3.4.2 Initial State Estimator

The PDDL initial states I must specify the initial state of the environment using propositions. Here, different predicates from D_D should be used for different domains to represent the propositions. In addition, omitting a single proposition could cause an invalid PD by making reaching the goal from the initial state impossible. We implement the initial state estimator with a combination of an LLM and image captioning model. Figure 3.3 shows the estimator. We use BLIP-2 (Li et al., 2023) as the captioning model and GPT-4 (OpenAI, 2023) as the LLM. Given the objects'

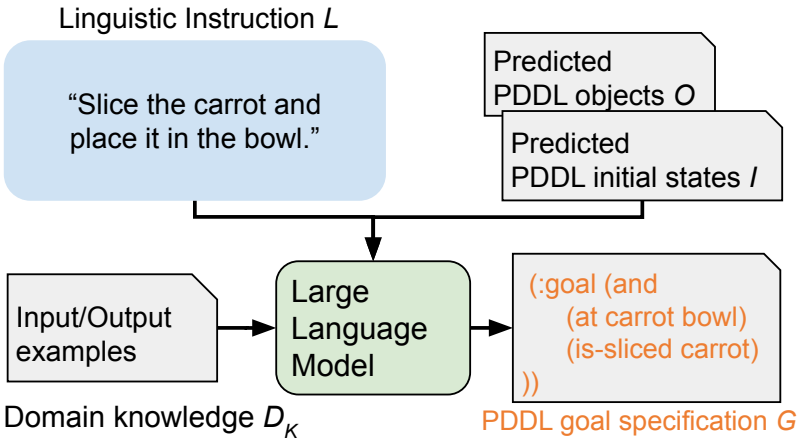

Figure 3.4: The LLM directly generates the PDDL goal specification from the instruction and the PDDL objects and initial state using few-shot prompting.

bounding boxes, BLIP-2 generates captions for each object with a prompt of "Q: what does this object describe? A: ." GPT-4 generates the PDDL initial state I from the bounding boxes and captions. GPT-4 uses few-shot prompting and leverages input/output examples in D_D to derive available predicates.

3.4.3 Goal Estimator

The PDDL goal specifications G must represent the desired goal conditions specified by the linguistic instructions L. Generating G requires O to refer to the object list and I to consider the relationships of the objects. We implement the goal estimator with an LLM, following previous work (Lin et al., 2023; Xie et al., 2023). Figure 3.4 shows the estimator. We use GPT-4 to generate G from L, O, and I. Similarly to Section 3.4.2, GPT-4 uses few-shot prompting with D_K.

3.4.4 Corrective re-prompting

Generated PDs are used by the planner to find plans. The planning might fail in the following two cases. One is when the PDs are syntactically incorrect. Generating propositions with undefined objects in O or undefined predicates in D_D results in such PDs (e.g., create `(at cucumber counter)`, but `cucumber` is not listed in O).

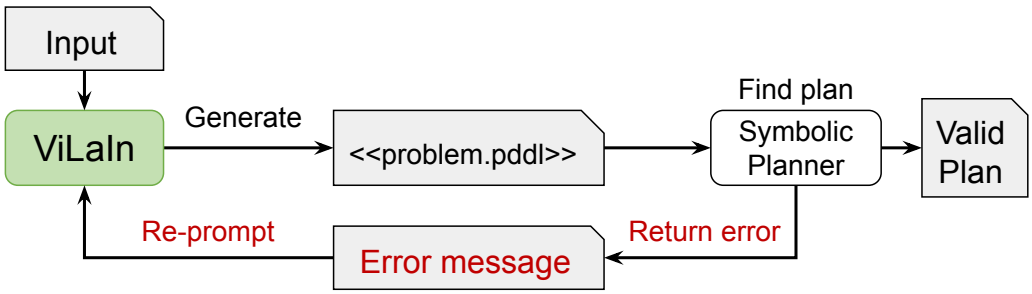

Figure 3.5: ViLaIn can refine the generated problem description via an error message from the planner.

The other is when the generated O is unreachable from the generated I. Contradictory propositions create such a PD (e.g., both of a proposition `(on red_block blue_block)` and the opposite one `(on blue_block red_block)` exist in I). In both cases, the planner stops planning and returns an error message, a clue to refine the erroneous parts. It is ideal if the system automatically refines the PDs via the error messages. ViLaIn has such a mechanism, and we describe it in this section.

When the planning fails, ViLaIn creates a prompt and re-prompts GPT-4 to refine the PD. We refer to this technique as Corrective Re-prompting (CR), following previous work (Raman et al., 2022). Figure 3.5 shows ViLaIn with CR. The prompt consists of input/output examples in D_K, the current input (L and S), the generated problem P, and the error message.

Chain-of-Thought prompting We use Chain-of-thought (CoT) prompting (Wei et al., 2022; Kojima et al., 2022; Zhou et al., 2023a) to further strengthen CR. CoT is a technique for solving complex reasoning tasks by LLMs. CoT introduces an intermediate reasoning step before generating the final output. With CoT, GPT-4 generates an explanation of the error message with a prompt template of "What part of the PDDL problem do you think is causing this error?." GPT-4 then generates the refined problem with the explanation. CR with CoT can be repeated as often as necessary until the planner returns error messages. In the rest of this chapter, ViLaIn generates the PDs using CR with CoT unless otherwise specified. Note that ViLaIn performs CR with CoT only if the planner returns an error message.

Domain	Object types	Predicates	Actions
Cooking	`vegetable`, `location`, `tool`, `robot`	`available`, `is-whole`, `is-sliced`, `free`, `carry`, `can-cut`, `at`, `at-workspace`	`pick`, `place`, `slice`
Blocksworld	`block`, `robot`	`on`, `ontable`, `clear`, `handempty`, `handfull`, `holding`	`pick-up`, `put-down`, `stack`, `unstack`
Hanoi	`disk`, `peg`	`clear`, `on`, `smaller`, `move`	`move`

Table 3.2: Defined object types, predicates, and actions in the domain descriptions.

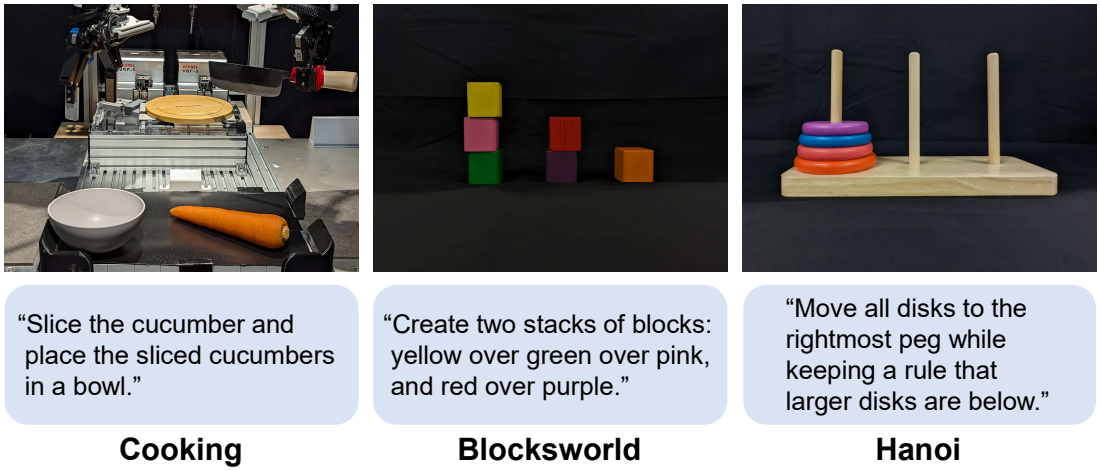

Figure 3.6: Examples of scene observations and linguistic instructions.

3.5 Dataset

The ProDG dataset consists of three domains: cooking, the blocks world (Blocksworld), and the tower of Hanoi (Hanoi).

Cooking is a simplified task of making a salad. Planning is simpler than the other two domains because it only considers slicing vegetables and placing them in the bowl. Cooking actions are supposed to be performed by two robot arms installed on both sides of the environment. The left and right robot arms are named `a_bot` and `b_bot`, respectively, in O. This domain handles a greater variety of objects than the other domains. G represents the vegetable state and location.

Blocksworld is a classical planning domain (Gupta and Nau, 1992). Fewer types of objects than the cooking appear, but a longer horizon planning is required. Seven colored blocks without duplicates are used for each problem. A robot arm does not

always grab anything at first. G specifies the relationships of the blocks.

Hanoi is a classical planning domain (Alford et al., 2009). Similarly to Blocksworld, a longer horizon planning with fewer types of objects than the cooking domain is required. Ten disks with six colors and three pegs are used. Disks of the same color are named by the number in order of increasing width (e.g., `blue_disk1` and `blue_disk2`). The three pegs are named by the number from left to right (e.g., `peg1`, `peg2`, and `peg3`). I and G specify the positions of the disks. Completing this task requires correctly recognizing the disk sizes since L only instructs the rule of the task, "larger disks are below," but mentions no concrete objects.

Each domain has one domain description and ten PDs. Table 3.2 shows object types, predicates, and actions in the domain descriptions. Each problem has one linguistic instruction and one scene observation. Figure 3.6 shows examples of linguistic instructions L and scene observations S. For the Hanoi domain, L is identical through all problems. This aims to investigate whether ViLaIn can generate different G based on O and I. The descriptions for the cooking domain were created from scratch, while those for the Blocksworld and Hanoi domains were created based on the PDDL files in pddlgym (Silver and Chitnis, 2020). We confirmed that all the created PDs are syntactically correct and have solutions using Fast Downward (Helmert, 2006) and VAL, a plan validation software.[2]

3.5.1 Evaluation metrics

In PD generation, previously proposed metrics roughly calculate the planning success rate or are domain-specific ones (Liu et al., 2023b; Xie et al., 2023). It would be ideal to have metrics that evaluate PDs from multiple perspectives regardless of domain. To this end, we introduce a new suite of metrics: $\mathrm{R}_{\mathrm{syntax}}$ and $\mathrm{R}_{\mathrm{plan}}$ for logical correctness and $\mathrm{R}_{\mathrm{part}}$ and $\mathrm{R}_{\mathrm{all}}$ for semantic correctness. We describe these metrics below.

$\mathbf{R_{syntax}}$ PDs must be syntactically correct. $\mathrm{R}_{\mathrm{syntax}}$ calculates the ratio of such PDs. A PD is considered to be syntactically correct if VAL returns no warnings and exit

[2] `https://github.com/KCL-Planning/VAL`

codes for a pair of the domain and the generated PD.

$\mathbf{R_{plan}}$ Even if the PDs are syntactically correct, they might not have valid plans due to incorrect objects in O and incorrect or contradictory propositions in I and G. $\mathrm{R_{plan}}$ calculates the ratio of the PDs having valid plans. The plans are obtained using Fast Downward (Helmert, 2006). A plan is considered to be valid if VAL returns no error messages.

$\mathbf{R_{part}}$ and $\mathbf{R_{all}}$ The above two metrics ignore whether the PDs are written about our intended tasks. For example, the PD might be about an unintended task while it is syntactically correct and has a valid plan. $\mathrm{R_{part}}$ evaluates how close the generated problems are to the ground truth ones. $\mathrm{R_{part}}$ calculates the recall of the problem parts between the ground truth and generated ones. $\mathrm{R_{part}}$ is independently computed for O, I, and G. The recall of object labels is calculated for O, while the recall of propositions is computed for I and G. Unlike $\mathrm{R_{part}}$, $\mathrm{R_{all}}$ calculates the ratio of problems containing all the ground truth object labels and propositions. Thus, $\mathrm{R_{all}}$ can be viewed as a harder metric than $\mathrm{R_{part}}$.

3.6 Experiments

We conduct experiments to investigate how accurately ViLaIn can generate PDs on the ProDG dataset. This section first describes the generation settings of ViLaIn and then discusses experimental results.

3.6.1 Generation settings of ViLaIn

GPT-4 used few-shot prompting with three input/output examples in the same domain as the current task. ViLaIn can refine erroneous PDs by CR n times. PDs with corrected grammatical errors can still have semantic errors, causing no valid solutions. In such cases, CR should be performed at least twice. Thus, we set n to two. For evaluation, we generated ten PDs per problem by varying the example combinations. The resulting 100 problems per domain are used to evaluate ViLaIn.

Domain	R_{syntax}	R_{plan}	R_{part}			R_{all}
			O	I	G	
Cooking	0.99	0.99	1.00	0.93	0.93	0.71
Blocksworld	0.99	0.94	0.98	0.79	0.89	0.36
Hanoi	1.00	0.58	0.89	0.46	0.33	0.12

Table 3.3: Performance on the ProDG dataset.

3.6.2 Evaluation of generation results by ViLaIn

Table 3.3 shows the results. The R_{syntax} scores are more than 99% in all the three domains. This means that ViLaIn can generate syntactically correct PDs for these domains utilizing the three input/output examples. The R_{plan} scores indicate that 94% or more PDs have valid plans in the cooking and Blocksworld domains. However, in the Hanoi domain, the R_{plan} score is only 58% due to its challenging setting. We found from the outputs that ViLaIn tends to omit some propositions in this domain, making the PDs invalid.

For R_{part}, the scores on I and G are smaller than those on O. This implies that generating I and G is more challenging than O. We found that mistakenly detected objects cause this. Predicates such as `on` or `at` take two objects as arguments. Propositions created with the predicates and mistakenly detected objects affect other propositions. For example, `(on red_block blue_block)` can be `(on red_block green_block) (on green_block blue_block)` with a mistakenly detected `green_block`, making them all incorrect propositions. We consider that generating these incorrect propositions causes such results.

Finally, the R_{all} score is 71% in the cooking domain, 36% in the Blocksworld domain, and 12% in the Hanoi domain. The scores in the cooking and Hanoi domains make sense considering the R_{plan} and R_{part} scores. However, the score is unexpectedly low in the Blocksworld domain. We found that PDs in the Blocksworld domain tend to contain a few incorrect propositions of block relationships. In some cases, the block positioning is mistakenly reversed (e.g., `(on blue_block red_block) (on red_block green_block)` is reversed to `(on green_block red_block) (on red_block blue_block)`). We consider that these lead to the low R_{all} score in this domain.

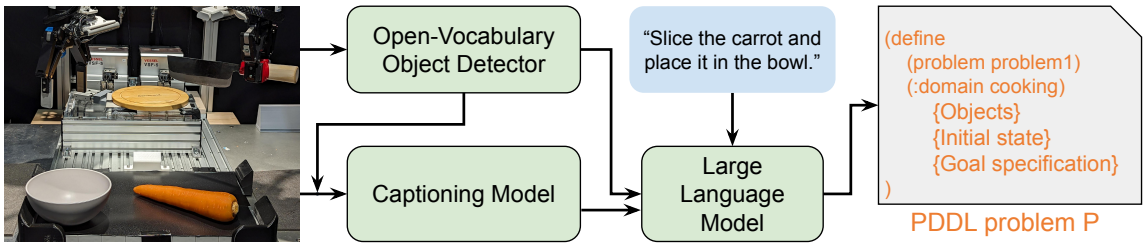

Figure 3.7: ViLaIn$_{\text{whole}}$ generates the whole problem description at once.

Domain	R_{syntax}	R_{plan}	R_{all}
Cooking	1.00 (+0.01)	1.00 (+0.01)	0.54 (-0.17)
Blocksworld	0.99 (+0.00)	0.99 (+0.05)	0.13 (-0.23)
Hanoi	1.00 (+0.00)	0.94 (+0.36)	0.21 (+0.09)

Table 3.4: Generating the whole problem descriptions at once.

3.6.3 Generating the whole problem at once

ViLaIn generates the parts of PDs using different modules. If a single module can generate the whole problem at once, it greatly simplifies the system. Here, we consider a variant of ViLaIn generating the whole PD at once, as illustrated in Figure 3.7. We refer to this model as ViLaIn$_{\text{whole}}$. The generation is performed with few-shot prompting as the original model.

Table 3.4 shows the results with R_{syntax}, R_{plan}, and R_{all}. Values inside parenthesis indicate gains from ViLaIn. In the cooking and Blocksworld domains, ViLaIn$_{\text{whole}}$ slightly improves R_{plan} but worsens R_{all}. This means that using three modules is more effective for these domains. In the Hanoi domain, ViLaIn$_{\text{whole}}$ outperforms ViLaIn in both R_{plan} and R_{all}. When considered with Section 3.6.2, this means that ViLaIn$_{\text{whole}}$ generates more correct propositions than ViLaIn. Generating the whole PDs makes the distance between tokens of O and I or G closer. We consider that this might work effectively and result in these improvements.

3.6.4 Generating PDs without CR and CoT

ViLaIn uses corrective re-prompting (CR) and chain-of-thought (CoT) prompting. The CR is performed twice at most as described in Section 3.6.1. Since all the PDs

CR configurations		R_{syntax}	R_{plan}	R_{all}
CR (n times)	CoT			
2	✓	0.99	0.99	0.71
1	✓	0.99	0.94	0.68
1		0.97	0.85	0.59
0		0.60	0.18	0.09

Table 3.5: Performance without CR and CoT.

so far are generated using CR with CoT, the impact of CR on performance is still unknown. Here, we investigate performance without CR and CoT, considering the following configurations: (i) CR with CoT ($n = 1$ in Section 3.6.1), (ii) CR without CoT ($n = 1$), and (iii) without CR ($n = 0$).

Table 3.5 shows the results in the cooking domain. The first line is the same result in Table 3.3. First, performing CR with CoT only once (the first line) slightly drops R_{plan} and R_{all}, meaning that repeating CR is effective. Next, removing CoT (the third line) worsens all the scores. This demonstrates that the introduced intermediate reasoning step by CoT has a large impact on performance. Finally, removing CR (the fourth line) degrades the scores significantly. This model tends to suffer from *hallucinations* (Maynez et al., 2020)[3], such as propositions with undefined objects (e.g., `(at cucumber counter)` in I while the cucumber is not defined in O). We found that CR effectively refines these incorrect propositions and makes the PDs consistent.

3.7 Conclusion

This chapter has tackled multimodal planning problem specification, a new task for connecting language-guided planning and symbolic planner. We have proposed Vision-language interpreter (ViLaIn) that generates problem descriptions (PDs) from linguistic instructions and scene observations. A novel dataset called the problem description generation (ProDG) dataset has proposed with new metrics to evaluate

[3]Also referred to as *confabulations*. Generating factually incorrect texts by LLMs is a common problem in natural language processing.

ViLaIn. The experimental results show that ViLaIn can generate syntactically correct PDs, and more than half of the PDs have valid plans.

Chapter 4

Flow Graph Prediction of Open-Domain Procedural Texts

4.1 Introduction

Procedural texts provide a list of steps to perform a variety of tasks, such as cooking or assembling furniture. Each step is a linguistic instruction to achieve a subgoal, which can include multiple actions and target objects. In recent years, research on understanding procedural texts has been gaining a lot of attention (Mori et al., 2014; Kiddon et al., 2015; Bosselut et al., 2018; Dalvi et al., 2018; Tandon et al., 2020). Among these studies, understanding the workflow of the whole procedural text (Mori et al., 2014; Kiddon et al., 2015) is particularly important for reasoning about the relationships between the steps (Zhang et al., 2020a) or automating work by robots based on the procedural texts (Bollini et al., 2013).

In this direction of research, previous work have proposed a recipe flow graph (r-FG) as a representation for the understanding of cooking recipes together with corpora (Mori et al., 2014; Yamakata et al., 2020). As shown in the left figure of Figure 4.1, the r-FG forms a directed acyclic graph (DAG). Here, nodes represent expressions related to the procedures in a cooking recipe, while edges represent the relationships between the nodes. The r-FG captures the dependencies between the procedures at the document level. Previous work has proposed a framework for

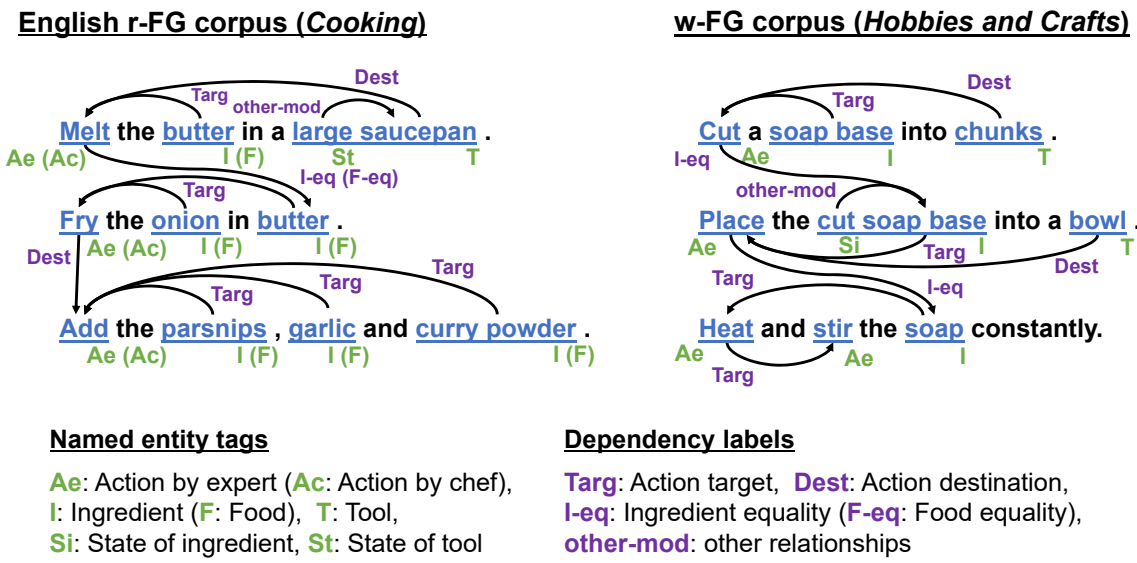

Figure 4.1: Example of flow graph annotation. The left shows the annotation in the English r-FG corpus, while the right shows the annotation in the w-FG corpus (*Hobbies and Crafts*).

automatic prediction of r-FG (Maeta et al., 2015). This framework predicts an r-FG in two stages: node prediction and edge prediction. While these developments have been made, the r-FG has not yet been applied to procedural texts in other domains because the representation is designed for the cooking domain. The development of a generalized flow graph representation would be a promising direction to realize procedural text understanding systems beyond the cooking domain.

This chapter proposes to extend the r-FG representation to handle non-cooking domains. Specifically, we focus on English articles from wikiHow, a website providing user-generated procedural texts in a wide range of domains, and propose a wikiHow flow graph (w-FG) representation. The w-FG extends the r-FG by treating foods in cooking recipes as ingredients of the final product. The w-FG is compatible with the r-FG, and the existing cooking annotations in r-FG can automatically be converted into annotations in w-FG. With this generalized representation, we investigate the flow graph prediction performance of English procedural texts in domains other than cooking. To predict flow graph performance in such domains, we introduce the w-FG corpus, a new flow graph corpus from articles on wikiHow. The w-FG corpus

Named entity tag	Meaning
I (F)	Ingredient (Food)
T	Tool
D	Duration
Q	Quantity
Ae (Ac)	Action by expert (chef)
Ae2 (Ac2)	Discontinuous Ae (Ac)
Ai (Af)	Action by ingredient (food)
At	Action by tool
Si (Sf)	State of ingredient (food)
St	State of tool

Table 4.1: Named entity tags and their meanings. The inside of the parenthesis represents a tag and its meaning in English-FG.

Dependency label	Meaning
Agent	Action agent
Targ	Action target
Dest	Action destination
T-comp	Tool complement
I-comp (F-comp)	Ingredient (Food) complement
I-eq (F-eq)	Ingredient (Food) equality
I-part-of (F-part-of)	Ingredient (Food) part-of
I-set (F-set)	Ingredient (Food) set
T-eq	Tool equality
T-part-of	Tool part-of
A-eq	Action equality
V-tm	Head of clause for timing
other-mod	Other relationships

Table 4.2: Dependency labels and their meanings. The inside of the parenthesis represents a label and its meaning in English-FG.

targets four domains selected from wikiHow's top categories: *Food and Entertaining*, *Hobbies and Crafts*, *Home and Garden*, and *Cars & Other Vehicles*. This corpus provides annotations of 30 articles for each domain. The w-FG corpus is available on the web.[1].

One common issue for r-FG-based representation is that its annotation procedure is dense and complicated. This means that obtaining large-scale annotations in new

[1] `https://github.com/kskshr/wikiHow-FG-Corpus`

domains is practically unrealistic. A possible solution to this issue is to utilize the existing annotations in the cooking domain as an additional resource. Based on this idea, we consider domain adaptation of models by first training on the existing r-FG corpus (Yamakata et al., 2020), which we call the English r-FG corpus in this chapter, and then fine-tuning on the target domain data of the w-FG corpus. The flow graph prediction is performed in the node and edge predictions, following previous work (Maeta et al., 2015). We experimentally show that the domain adaptation significantly improves performance compared with models trained only on the English r-FG or w-FG corpus.

4.2 Recipe flow graph

This section briefly explains the r-FG representation and the framework for automatic r-FG prediction.

4.2.1 Flow graph representation

As shown in the left figure of Figure 4.1, an r-FG is defined as a directed acyclic graph $G(V, E)$, consisting of a set of nodes V and a set of edges E. V is a set of representations related to procedures such as foods and tools, and E is a set of labeled edges representing dependencies between the nodes. The graph is connected, and there is a special root node corresponding to the final product. In previous work (Yamakata et al., 2020), the cooking action representation (Ac) that appeared last is used as the root node. Currently, the Japanese r-FG corpus (Mori et al., 2014) and the English r-FG corpus (Yamakata et al., 2020) are provided. Here, the English r-FG (Yamakata et al., 2020) adds a few new tags and labels to handle English-specific expressions. As shown in Table 4.1 and Table 4.2, the English r-FG uses 10 types of named entity tags and 13 types of dependency labels. Since this chapter focuses on English procedural texts, our proposed w-FG is based on the English r-FG representation.

4.2.2 Flow graph prediction

A framework proposed by previous work (Maeta et al., 2015) predicts an r-FG in two stages of node and edge prediction.

The node prediction identifies expressions corresponding to the nodes with r-FG tags. This is formulated as a sequence labeling problem, and prediction is performed using a named entity recognizer (NER). While it is common to perform named entity recognition at the sentence level (Lample et al., 2016), previous work performed prediction at the article level (Yamakata et al., 2020). Following the previous work, this chapter performs node prediction at the article level.[2]

The edge prediction predicts dependencies between the nodes with labels. This is formulated as a problem of finding a maximum spanning tree as follows:

$$\hat{G} = \underset{G \in \mathcal{G}}{\operatorname{argmax}} \sum_{(u,v,l)} s(u, v, l), \tag{4.1}$$

where $s(u, v, l)$ represents the score for a directed edge from the node u to v with the label l. This is solved using the Chu-Liu-Edmonds algorithm (Chu and Liu, 1965; Edmonds, 1967). The score for labeled edges is computed using a graph-based dependency parser (McDonald et al., 2005).

4.3 wikiHow flow graph representation

A wikiHow flow graph (w-FG) is a flow graph representation based on the English r-FG. Table 4.1 and Table 4.2 show the named entity tags and dependency labels in the w-FG. In the r-FG, food ingredients and intermediate products of a procedure are treated as food (F), and they are incorporated into the final product of the recipe. For example, in a salad recipe, foods such as lettuce and dressing are treated as the ingredients of the salad, the final product of the recipe. The w-FG generalizes the r-FG by treating all materials incorporated into the final product, including foods, as ingredients (I) and handles procedural texts other than cooking recipes. For example,

[2]Our preliminary experiments have shown that prediction at the article level can improve accuracy by up to 10% compared to the sentence level.

in a desk assembly manual, materials such as desk legs and screws are treated as the ingredients of the desk, the final product of the procedural text. In the w-FG, the named entity tag of food (F) is changed to ingredient (I), and other tags and labels related to foods are replaced with those related to ingredients (e.g., State of foods (Sf) → State of ingredients (Si), Food equality (F-eq) → Ingredient equality (I-eq)). The w-FG is compatible with the r-FG, and the annotations in the r-FG can automatically be converted into those in the w-FG.

4.3.1 Flow graph prediction

As in previous work (Maeta et al., 2015; Yamakata et al., 2020), the flow graph prediction is performed in two stages: node prediction and edge prediction. The prediction model is obtained by training on w-FG annotations. We assume that only a small amount of the target domain data is available for training, considering the huge annotation cost of the w-FG. To remedy this issue, we consider domain adaptation from the existing r-FG corpus to the target domain data, as mentioned in Section 4.1. Here, the domain adaptation is performed by first training a model on the r-FG corpus and then by fine-tuning it on the target domain data. The remainder of this section describes task definition in Section 4.3.2 and data augmentation techniques to further address the low-resource problem in Section 4.3.3.

4.3.2 Task definition

Given N examples of flow graph $(V_1^C, E_1^C), \cdots, (V_N^C, E_N^C)$ in the cooking domain and M examples of flow graph $(V_1^T, E_1^T), \cdots, (V_M^T, E_M^T)$, the goal of this task is to maximize the prediction performance of the node prediction model F_{Node} and the edge prediction model F_{Edge} in the target domain. Here, we assume that

$$F_{\text{Node}} : D \rightarrow V, \tag{4.2}$$

$$F_{\text{Edge}} : (D, V) \rightarrow E, \tag{4.3}$$

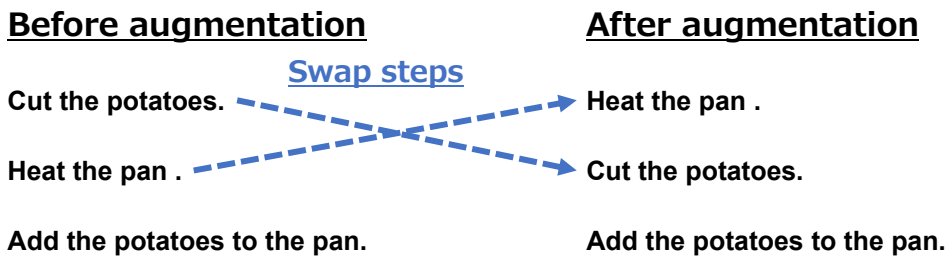

Figure 4.2: Example of swapping steps.

where D is a procedural text. M is a small number in this setting (namely, $M = 5$ in the experiment), and this task can be regarded as a low-resource domain adaptation (Xu et al., 2021) task. In addition, if the data in the cooking domain or the target domain is not used, it can be regarded as a few-shot or zero-shot setting.

4.3.3 Data augmentation

Data augmentation is a promising approach to improve performance in low-resource settings (Fadaee et al., 2017; Ding et al., 2020). In this study, we consider two types of data augmentation techniques: step swapping and word replacement.

Step swapping augments an example by replacing two arbitrary steps in an article as illustrated in Figure 4.2. However, randomly choosing and swapping two steps might break their causal relationship. For example, the two steps of "1. Cut the potatoes." and "2. Add the potatoes to the pan." in Figure 4.2 cannot be swapped. In this work, we augment examples while keeping this constraint by using flow graph annotations.[3]

word replacement augments an example by replacing a word in the procedural text with an arbitrary word. For example, "heat the pan." is augmented by replacing "pan" with "cooking_pan." However, replacing a word with unrelated words may

[3]In fact, the duration between the steps should be considered. For example, the two steps of "1. Heat the pan." and "2. Add the potatoes to the pan." might not be swapped because leaving the heated pan for a long time affects the next step. In this chapter, we ignore such implicit constraints and focus only on explicit constraints provided by the flow graph.

significantly change the meaning of the procedure. Therefore, we use WordNet (Dai and Adel, 2020) and replace words with their synonyms. Each word is replaced with a probability of $p(0 \leq p \leq 1)$, where p is a hyperparameter. While the word may have multiple synonyms, in which case one is randomly selected from all the synonyms. This chapter targets only words annotated with the named entity tags of Ae, I, or T for augmentation.

4.4 w-FG corpus

The w-FG corpus is a novel flow graph corpus of articles from wikiHow.[4] wikiHow provides more than 230,000 procedural texts and has been widely used as a language resource of procedural texts in recent years (Zhou et al., 2019; Zellers et al., 2019b; Zhang et al., 2020a; Zhou et al., 2022; Lin et al., 2022). In the following, the data collection, annotation procedure, statistics, and inter-annotator agreements are described in order.

4.4.1 Data collection

Four target domains were selected from the top wikiHow categories: *Food and Entertaining*, *Hobbies and Crafts*, *Home and Garden*, *Cars & Other Vehicles*. *Food and Entertaining* mostly targets cooking, and thus, the included procedures are relatively close to those in the English r-FG corpus. *Hobbies and Crafts* mainly targets crafting, which is different from cooking in that inedible materials are used, but have in common that a final product is obtained by assembling ingredients. The other two domains of *Home and Garden* and *Cars & Other Vehicles* contain procedural texts for gardening and vehicle maintenance, respectively. The procedures used in these domains are more diverse than the other two domains, including non-assembly tasks.

30 wikiHow articles for each domain were collected from the wikiHow corpus (Zhang et al., 2020a). Table 4.3 shows examples of article titles for each domain. During the data collection, we filtered low-quality articles by collecting those that (i) had at least

[4] `https://www.wikihow.com`, accessed on December 20, 2022.

Domain	Article titles
Food and Entertaining	*Cooking acorn squash*, *Making lavender tea*, *Baking a cherry pie*
Hobbies and Crafts	*Making a bar soap*, *Making a duct tape bow*, *Making a paper box*
Home and Garden	*Cleaning a mattress pad*, *Installing a microwave*, *Making a scented candle*
Cars & Other Vehicles	*Fixing a slipped bike chain*, *Cleaning car window*, *Cleaning tail lights*

Table 4.3: Examples of article titles for each domain.

Domain	# characters	# words	# steps	# tags	# labels
Food and Entertaining	10,167	2,761	224	1,123	1,127
Hobbies and Crafts	9,407	2,556	247	1,048	1,059
Home and Garden	7,700	2,010	205	887	882
Cars & Other Vehicles	6,432	1,622	173	613	609

Table 4.4: Statistics of the w-FG corpus.

25 words in the article and (ii) were rated at least 50% by users. We also manually excluded articles for which the task was ambiguous or the final product was not a physical object. In this study, paragraph headings are used as steps, following previous work (Zhang et al., 2020a; Zhou et al., 2022). All the steps were segmented into words by using Stanza (Qi et al., 2020) before the annotation.

4.4.2 Annotation procedure

The annotation of the wikiHow articles was done by requesting a human annotator. The annotator was trained by annotating 10 recipes randomly collected from the English r-FG corpus with annotation instructions until the agreement with the existing annotations exceeded 80%. We then instructed the annotator with more detailed annotation standards and requested them to annotate the 120 articles. The annotation was performed by using the flow graph annotation tool (Shirai et al., 2022). The annotation took 40 hours in total.

Named entity tag	*Food and Entertaining*	*Hobbies and Crafts*	*Home and Garden*	*Cars & Other Vehicles*
I	380	419	250	218
T	136	56	186	91
D	41	9	17	4
Q	73	46	32	14
Ae	315	310	270	202
Ae2	15	38	19	23
Ai	28	22	17	11
At	0	0	0	0
Si	84	147	42	35
St	60	15	61	27
Total	1,132	1,062	894	625

Table 4.5: The number of annotations for each named entity tag.

4.4.3 Statistics

Table 4.4 shows the statistics of the w-FG corpus. We can see that each article consists of an average of 7.1 steps, and each step consists of an average of 10.5 words. It also shows that each article is annotated with an average of 30.6 named entity tags and 30.6 dependency labels. The number of words included in *Food and Entertaining* and *Hobbies and Crafts* are fewer than the other two domains, and the number of annotated tags and labels for the former two domains are also smaller than the other two domains.

Table 4.5 shows the number of annotations for each tag. We can see that Ae, I, and T are frequently appeared across the domains. Among the tags, At did not appear in this annotation, but it also has a very low frequency of 15 in the English r-FG corpus. Considering the number of articles in Food and Entertaining is 30, one-tenth the size of the English r-FG corpus (300 articles), this value is reasonable.

Figure 4.3 shows the frequency distribution of the top 10 expressions annotated with Ae, I, and T for each domain. For Ae, *add* and *cut* frequently appear in Food and Entertaining and Hobbies and Crafts, while *remove* and *use* are more used in Home and Garden and Cars & Other Vehicles. These verbs characterize the procedures used in these procedures. It should be noted that expressions used as I in certain domains can be used as T in other domains (e.g., *water* is frequently used as I in Food and

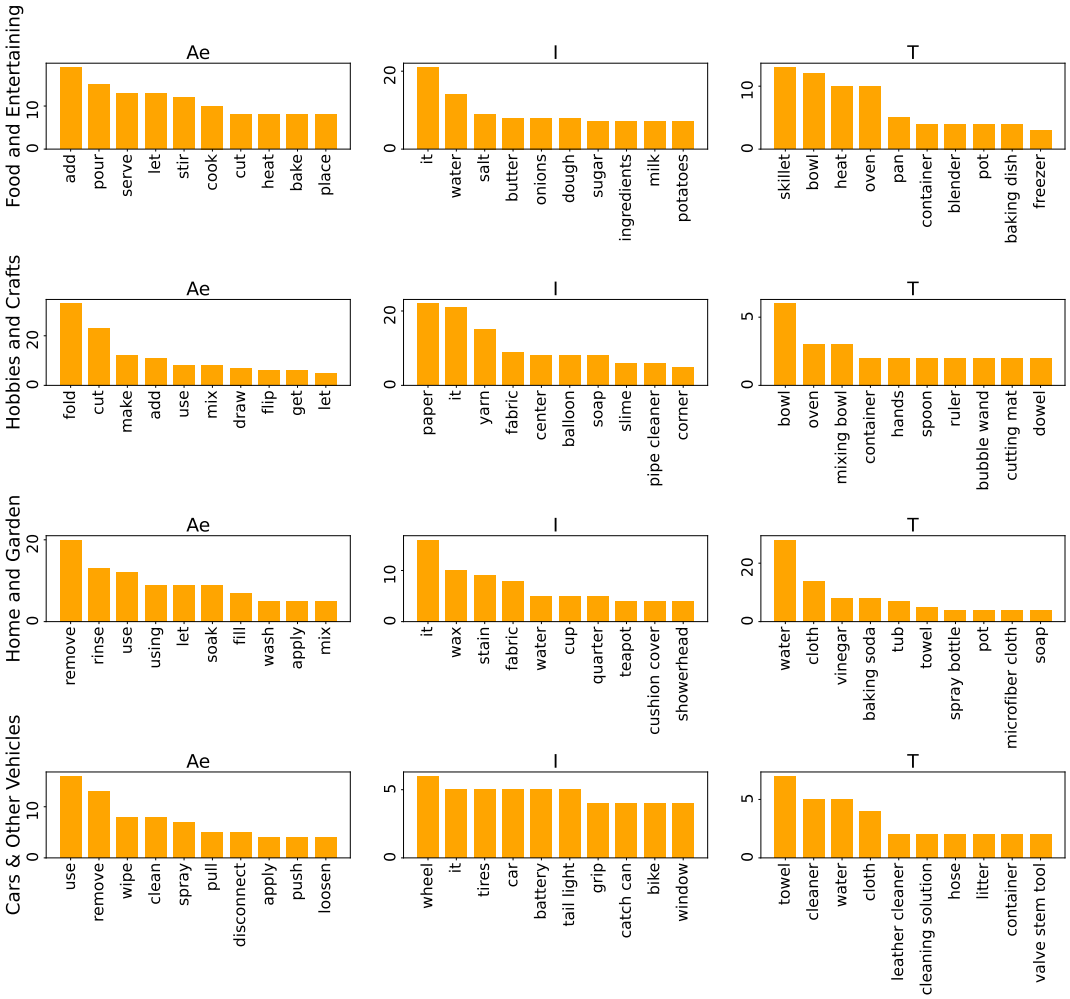

Figure 4.3: The top 10 high frequent expressions for Ae, I, and T.

Entertaining, while it tends to be used as T in Home and Garden and Cars & Other Vehicles).

Table 4.6 shows the number of annotations for each label. It can be seen that Targ, Dest, I-eq, and other-mod have high frequencies across all domains. In addition, the frequency of T-comp is high in *Home and Garden* and *Cars & Other Vehicles*, indicating that actions using tools are especially common in these domains. Further, the frequency of I-part-of is higher in *Hobbies and Crafts* than in the other three domains, suggesting that actions on the part of the ingredients frequently appear in this domain.

Dependency label	*Food and Entertaining*	*Hobbies and Crafts*	*Home and Garden*	*Cars & Other Vehicles*
Agent	46	49	25	20
Targ	396	341	301	231
Dest	151	145	100	47
T-comp	25	19	64	42
I-comp	7	4	3	3
I-eq	146	149	110	68
I-part-of	29	124	63	76
Set	8	7	0	2
T-eq	21	8	18	10
T-part-of	5	3	16	6
A-eq	4	6	2	6
V-tm	32	9	2	2
other-mod	254	212	182	109
Total	1,124	1,076	886	622

Table 4.6: The number of annotations for each dependency label.

Annotation type	Agreement
Node annotation	89.68%
Edge annotation	68.79%

Table 4.7: Inter-annotator agreements.

4.4.4 Inter-annotator agreement

To evaluate the quality of the annotations, we asked another annotator to re-annotate 3 articles for each domain and then calculated the inter-annotator agreement by F1. Here, the annotator was trained in the same way as explained in Section 4.4.2. Table 4.7 shows the results. The agreement on the node annotation and edge annotation were was 89.68% and 68.79%, respectively. Given the complexity of flow graph annotation and the number of domains handled in this work, we consider that these scores of over 68% would be acceptable to build machine-learning models on them.

4.5 Node prediction

As explained in Section 4.3, the flow graph prediction is performed in the two stages of the node prediction and the edge prediction. This section evaluates the node prediction performance on the w-FG corpus. We use a neural network-based NER model to predict nodes. This section first explains experimental settings and then shows the experimental results. After that, the node prediction performance at the tag level is investigated as an ablation study.

4.5.1 Experimental settings

Model. We used a BiLSTM-CRF model as the named entity recognizer (Lample et al., 2016). As an encoder, we used the pre-trained DeBERTa (He et al., 2021) instead of BiLSTM or BERT (Devlin et al., 2019).[5] The total number of parameters of this node prediction model was 140M.

Training. The domain adaptation was performed by first training the model on the English r-FG corpus and then by fine-tuning it on the target domain data of the w-FG corpus. We also trained the model without training on the English r-FG corpus or the w-FG corpus to investigate the effect of the domain adaptation. During training, all parameters, including the pre-trained DeBERTa, were tuned.

The model parameters were tuned by using AdamW (Loshchilov and Hutter, 2019) with an initial learning rate of 5.0×10^{-5} and the weight decay of 1.0×10^{-5}. The learning rate schedule was performed with the warmup for S_{w} steps and the cosine-annealing (Loshchilov and Hutter, 2019) for S_{d} steps. The minibatch was created from B articles. We set $(B, S_{\mathrm{w}}, S_{\mathrm{d}}) = (5, 500, 4500)$ for the training on the English r-FG corpus and $(B, S_{\mathrm{w}}, S_{\mathrm{d}}) = (3, 100, 900)$ for training on the w-FG corpus. These hyperparameters were adjusted using the development data. The data augmentation described in Section 4.3.3 was performed only for training the domain adaptation model. The step swapping was performed to augment 5 articles for each article, and

[5] In our preliminary experiments, we confirmed that using DeBERTa instead of BERT improves the accuracy by 0.47% on the English r-FG corpus.

the word replacement was performed to augment 10 articles for each 1 article, setting $p = 0.5$.

Evaluation. The English r-FG corpus was split into 80% for the train data, 10% for the development data, and the remaining 10% for the test data. The w-FG corpus was split into 6 splits, in which each split contains 5 articles. We used a 1 split for the train data, another 1 split for the development data, and the remaining 4 splits for the test data. To obtain more reliable results, we performed 6-fold cross-validation by varying the choice of the split for the test data. Precision, recall, and F1 were used as evaluation metrics, following previous work (Maeta et al., 2015).

Model configuration. In the following, we refer to models that perform domain adaptation as **DA**, models trained only on the English r-FG corpus as **COOK**, and models trained only on the w-FG corpus as **TGT**, respectively.

4.5.2 Experimental results

Table 4.8 shows the results. The scores of the **TGT** model show that it can predict nodes with 66.9% or more F1 values using only 5 annotated articles as the training data. The **COOK** model results show that training on the English r-FG corpus achieves competitive scores as **TGT**. In *Food and Entertaining*, the **COOK** model outperforms **TGT** by a large margin of 10.3% F1, but this is because these domains mainly deal with the same domain of cooking. The **DA** models achieve the best scores in all domains. In the three domains other than *Food and Entertaining*, the **DA** model improves F1 by more than 9.6%. This indicates that pre-training on the English r-FG corpus is effective in this setting.

For the results of the data augmentation, the step swapping shows a small improvement of 0.4% and 0.3% in the two domains of *Food and Entertaining* and *Home and Garden*, respectively. The word replacement did not achieve any improvements in all domains. These results indicate that the proposed data augmentation techniques are not effective in the node prediction.

Domain	Model	Data augmentation		Precision	Recall	F1
		Step swap.	Word replace.			
Food and Entertaining	TGT			0.770	0.784	0.777
	COOK			0.884	0.877	0.880
	DA			0.890	0.892	0.891
	DA	✓		**0.894**	**0.895**	**0.895**
	DA		✓	0.885	0.891	0.888
Hobbies and Crafts	TGT			0.698	0.707	0.702
	COOK			0.703	0.684	0.693
	DA			**0.794**	**0.805**	**0.799**
	DA	✓		0.784	0.795	0.789
	DA		✓	0.781	0.790	0.785
Home and Garden	TGT			0.663	0.676	0.669
	COOK			0.734	0.742	0.738
	DA			0.780	0.786	0.783
	DA	✓		**0.787**	**0.791**	**0.786**
	DA		✓	0.765	0.773	0.769
Cars & Other Vehicles	TGT			0.650	0.690	0.669
	COOK			0.646	0.695	0.670
	DA			**0.748**	**0.784**	**0.765**
	DA	✓		0.734	**0.784**	0.761
	DA		✓	0.729	0.772	0.750

Table 4.8: Results of the node prediction. The checkmark ✓ represents the used data augmentation technique.

4.5.3 Tag-level prediction performance

The expressions corresponding to named entity tags differ significantly across domains. Thus, we can expect that the lower the overlap of expressions between the English r-FG corpus and the w-FG corpus, the more domain-dependent expressions are learned during fine-tuning, and thus the larger improvement from the **COOK** to **DA** model is achieved. To investigate this, we calculated the overlap ratio of expressions for each tag across the corpora and evaluated the prediction performance at the tag level. Here, we evaluated the performance by F1 and targeted only the most frequently appearing tags of Ae, C, and T.

Table 4.9 shows the results. For Ae, unlike expected, the improvement from **COOK** to **DA** is small, regardless of the overlap rate between corpora. This implies that human action expressions can easily be identified regardless of the training data

Domain	Ae			I			T		
	Overlap	F1		Overlap	F1		Overlap	F1	
		COOK	DA		COOK	DA		COOK	DA
Food and Entertaining	92.06%	0.941	0.952	72.11%	0.932	0.933	77.94%	0.896	0.882
Hobbies and Crafts	69.03%	0.943	0.951	10.33%	0.717	0.833	51.79%	0.398	0.588
Home and Garden	65.19%	0.954	0.961	18.40%	0.716	0.795	43.55%	0.567	0.678
Cars & Other Vehicles	46.04%	0.905	0.919	6.88%	0.666	0.805	27.47%	0.459	0.557

Table 4.9: Node prediction performance for Ae, I, and T. The overlap ratio represents the overlap of expressions between the English r-FG corpus and the target domain data of the wikiHow-FG corpus.

domain. For C and T, we can see significant performance improvement by domain adaptation in all domains except for *Food and Entertaining*. The overlap ratio in these three domains is lower than that in Food and Entertaining, indicating that the fine-tuning has a larger impact on the prediction performance of the C and T tags.

4.6 Edge prediction

After the node prediction, the edge prediction is performed to predict dependencies between the nodes as labeled edges. We use a graph-based dependency parser as the edge prediction model. Following the previous work (Yamakata et al., 2020), the action expression (Ae) that appeared last in a procedural text is used as the root node. This section describes experimental settings and results when the ground-truth nodes are provided. Then, the results of the pipeline experiment that predicts the edges based on the predicted nodes are provided.

4.6.1 Experimental settings

Model. We used a biaffine dependency parser (Dozat and Manning, 2018) as the dependency parser.[6] This model uses different modules for edge and label prediction, and the loss function is designed as a weighted sum of the losses in each module as

[6]Previous work used a liner model to perform the edge prediction (Yamakata et al., 2020). In our preliminary experiment, we confirmed that the biaffine dependency parser can achieve higher performance on the English r-FG corpus. The results of this preliminary experiment are provided in Section 4.6.4.

follows:

$$l = \lambda l^{\text{Edge}} + (1 - \lambda) l^{\text{Label}}, \tag{4.4}$$

where λ controls the strength of each loss, and we set to 0.5. We used the pre-trained DeBERTa (He et al., 2021) instead of the BiLSTM as the language encoder.[7]. The total number of parameters for this model was 1.49M.

Training. Similarly to Section 4.5.1, we trained models trained by domain adaptation and those trained only on the English r-FG corpus or the w-FG corpus. The model parameters were optimized using AdamW (Loshchilov and Hutter, 2019), and the learning rate schedule consisting of the warmup and cosine-annealing was performed. For these hyperparameters, we used the same values as in Section 4.5.1.

Evaluation. We used the same splits for the English r-FG corpus and w-FG corpus in Section 4.5.1. Similarly to the node prediction experiments, we performed the 6-fold cross-validation. Evaluation metrics of precision, recall, and F1 were calculated based on labeled edges (u, v, l), where u, v, and l are the starting node, ending node, and dependency label, respectively.

Model configuration. As with Section 4.5, we refer to each model by **COOK**, **TGT**, and **DA**.

4.6.2 Experimental results

Table 4.10 shows the edge prediction results based on the ground-truth nodes. Unlike the node prediction experiments, we can see that the scores of the **TGT** models are 33.8% or lower F1 in all domains. On the other hand, the score of the **COOK** model is 58.7% or more in all domains, doubling the scores of the **TGT** model in each domain. These results suggest that the edge prediction model requires more training data than the node prediction model to outperform the **COOK** model, which is trained on the cooking domain data. The **DA** model outperforms both the **TGT**

[7]Section 4.6.4 provides the results by varying the language encoder.

Domain	Model	Data augmentation		Precision	Recall	F1
		Step swap.	Word replace.			
Food and Entertaining	TGT			0.335	0.338	0.337
	COOK			0.725	0.731	0.728
	DA			0.750	**0.756**	**0.753**
	DA	✓		0.747	0.752	0.750
	DA		✓	**0.761**	0.752	0.749
Hobbies and Crafts	TGT			0.285	0.281	0.283
	COOK			0.613	0.605	0.609
	DA			0.649	0.640	0.644
	DA	✓		0.646	0.638	0.642
	DA		✓	**0.653**	**0.644**	**0.648**
Home and Garden	TGT			0.229	0.232	0.231
	COOK			0.644	0.649	0.646
	DA			0.659	0.665	0.662
	DA	✓		0.656	0.662	0.659
	DA		✓	**0.674**	**0.680**	**0.677**
Cars & Other Vehicles	TGT			0.154	0.155	0.154
	COOK			0.587	0.590	0.587
	DA			0.607	0.610	0.609
	DA	✓		0.607	0.610	0.608
	DA		✓	**0.617**	**0.620**	**0.618**

Table 4.10: Results of the edge prediction. The checkmark ✓ represents the used data augmentation technique.

and **COOK** models in all domains, achieving the best scores. This indicates that the domain adaptation from the English r-FG corpus to the target domain is as effective as the node prediction. In addition, for the data augmentation results, the word replacement slightly improves the scores by up to 0.15% in all domains except for *Food and Entertaining*. This implies that data augmentation by replacing node expressions might have a positive effect on training.

4.6.3 Pipeline experiments

In more practical settings, flow graph prediction must be performed on raw procedural texts. In this scenario, the edge prediction model needs to predict the edges based on the predicted nodes. In this case, the edge prediction performance is expected to be lower than the results in Section 4.6.2, propagating the errors from the node

Domain	F1
Food and Entertaining	0.679 (-9.8%)
Hobbies and Crafts	0.501 (-22.2%)
Home and Garden	0.494 (-25.4%)
Cars & Other Vehicles	0.449 (-26.3%)

Table 4.11: Results of the pipeline experiments. The value inside the parenthesis represents the difference from Table 4.10.

prediction step. To investigate the prediction performance in this setting, we perform edge prediction based on the predicted nodes provided in Table 4.5.2. The evaluation was performed by calculating the F1 based on tuples of (u, v, l, n_u, n_v), where n_u and n_v are named entity tags for the starting and ending nodes u and v, respectively.

Table 4.11 shows the edge prediction results based on the predicted nodes. These scores indicate that edges can be predicted with F1 ranging from 44.9% to 67.9% in this setting. The performance drop from Table 4.10 was 9.8% for *Food and Entertaining* and 24.6% or larger in the other three domains. The difference of F1 scores between *Food and Entertaining* (89.1%) and the other three domains (between 76.5% and 79.9%) is 9.2% or larger, and this difference seems to lead to the performance difference in Table 4.11.

4.6.4 Further experiments

Language model	Precision	Recall	F1
(Yamakata et al., 2020)	0.737	0.686	0.711
BERT	0.737	0.703	0.720
RoBERTa	0.754	0.719	0.736
Longformer	0.751	0.716	0.733
ALBERT	0.744	0.710	0.727
DeBERTa	**0.756**	**0.721**	**0.738**

Table 4.12: Edge prediction results on the English r-FG corpus when varying the language model.

Edge prediction performance on the English-FG corpus. We compared the performance of edge prediction with the previous work (Yamakata et al., 2020) in

the previous work. In order to match the experimental setting with the previous work, we performed a 10-fold cross-validation by using 80% of the entire corpus for the training data, 10% for the development data, and the remaining 10% for the test data. In addition, we investigated the performance when using the pre-trained BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2019), DeBERTa (He et al., 2021), Longformer (Beltagy et al., 2020), and ALBERT (Lan et al., 2020) as the language encoder.

The results are shown in Table 4.12. This indicates that a neural network-based architecture with a pre-trained language encoder always outperforms the linear model used in the previous work. In addition, we can also see that using DeBERTa achieves the best result.

Comparison of language models for edge prediction on the w-FG corpus. We compared the edge prediction performance on the w-FG corpus when using different language models as the language encoder. The choice of the language encoder is selected from BERT, RoBERTa, Longformer, ALBERT, and DeBERTa.

Table 4.13 shows the results of the **DA** model. The models using BERT and ALBERT achieve the best F1 in *Home and Garden* and *Cars & Other Vehicles*. In the other two domains, the model using DeBERTa works best.

4.7 Related work

In addition to the textual annotations for the r-FG (Mori et al., 2014; Yamakata et al., 2020), recent work provided cross-modal annotations for the r-FG corpus (Nishimura et al., 2020; Shirai et al., 2022). Nishimura et al. (2020) annotated cooking steps with images with bounding boxes that indicate the location of textual expressions in the steps, such as foods and actions. Shirai et al. (2022) annotated cooking actions in the steps with image pairs representing pre-action and post-action states of objects. Since the steps of wikiHow articles are paired with visual illustrations, the w-FG corpus can be extended to have cross-modal annotations as in previous work.

Domain	Language model	F1
Food and Entertaining	BERT	0.729
	RoBERTa	0.736
	Longformer	0.735
	ALBERT	0.725
	DeBERTa	**0.753**
Hobbies and Crafts	BERT	0.637
	RoBERTa	**0.644**
	Longformer	0.631
	ALBERT	0.603
	DeBERTa	**0.644**
Home and Garden	BERT	**0.672**
	RoBERTa	0.665
	Longformer	0.662
	ALBERT	0.627
	DeBERTa	0.662
Cars & Other Vehicles	BERT	0.611
	RoBERTa	0.610
	Longformer	0.600
	ALBERT	**0.623**
	DeBERTa	0.609

Table 4.13: Edge prediction results on the wikiHow-FG corpus when varying the language model.

Other than the r-FG and w-FG, there are several studies that represent procedural texts as a graph structure. In the cooking domain, Kiddon et al. (2015) proposed to acquire graph representations of cooking recipes using unsupervised learning. Some previous work provided human-annotated graph representations of cooking recipes (Pan et al., 2020; Papadopoulos et al., 2022), similarly to the r-FG. In the biochemistry domain, an approach to transform protocols into graph representations was proposed for automating experiments (Kulkarni et al., 2018; Tamari et al., 2021). In the material science domain, a corpus representing synthesis procedures as a directed acyclic graph was proposed for analyzing synthesis processes in the scientific literature (Mysore et al., 2019; Kuniyoshi et al., 2020). Since the procedures in the biochemistry and material science domains are performed on physical objects, these procedural texts are also possible to represent in the w-FG.

wikiHow has widely been used as a language resource, providing knowledge of

diverse procedures in previous work. Zhou et al. (2019) and Zhou et al. (2022) used a knowledge base of procedures. Zellers et al. (2019b) created a dataset for commonsense reasoning based on wikiHow articles. Zhang et al. (2020a) and Zhang et al. (2020b) proposed a task of inferring goals from procedures on wikiHow articles. Lin et al. (2022) and Zhou et al. (2023b) utilized procedural knowledge from wikiHow for understanding procedures in working videos. This work followed this research trend and created a dataset based on wikiHow articles.

4.8 Conclusion

This chapter has proposed a wikiHow flow graph (w-FG) based on the English recipe flow graph to represent procedural texts in non-cooking domains as flow graphs. To evaluate flow graph prediction performance in non-cooking domains, we have newly created the w-FG corpus based on wikiHow articles. In experiments, we have assumed a low-resource setting and performed domain adaptation from the existing cooking corpus to the target domain data of the w-FG corpus. We have experimentally shown that the domain adaptation approach effectively works to predict the nodes and labeled edges of the flow graph in a low-resource setting.

Chapter 5

Conclusion

5.1 Summary

We have described our contributions towards realizing intelligent robots that interpret and act on procedural texts.

In Chapter 2, we have proposed the Visual Recipe Flow (VRF) dataset, a new multimodal cooking dataset for predicting post-action visual states of objects. The data collection, annotation procedure, statistics, and the quality of the annotations by inter-annotator agreements have been explained in order. The results of multimodal retrieval experiments have shown that textual and visual annotations both play an important role in predicting the post-action states of objects.

In Chapter 3, we have proposed a Vision-Language Interpreter (ViLaIn), a novel framework that converts linguistic instructions and scene observations into problem descriptions and drives a symbolic planner to find logically correct plans of symbolic actions. We have created the Problem Description Generation (ProDG) dataset, a new dataset for the evaluation of ViLaIn. We have experimentally shown that ViLaIn can generate syntactically correct problem descriptions and find valid plans with high accuracy.

In Chapter 4, we have extended a recipe flow graph representation and proposed a wikiHow flow graph (w-FG) to handle procedural texts in non-cooking domains. We have constructed the w-FG corpus from 120 wikiHow articles in 4 domains to

investigate flow graph prediction performance in non-cooking domains. We have experimentally shown that domain adaptation from the r-FG corpus to the w-FG corpus significantly improves performance both on node prediction and edge prediction in a low-resource setting.

5.2 Future Work

Building models that predict post-action visual states of objects Based on the VRF dataset proposed in this thesis, the next step in this direction would be to build models that predict the post-action visual states of objects. To achieve our goal, we need to develop a conditional generative model that predicts the post-action image based on the pre-action image and step. Recent developments in image generation models (Ho et al., 2020; Rombach et al., 2022; Zhang et al., 2023) have significantly decreased the difficulty of realizing models that predict in pixel space. Similarly to our interest, Brooks et al. (2023) developed a model that edits an image based on an instruction, and Souček et al. (2023) proposed GenHowTo that generates a post-action image based on the pre-action image and a text prompt. In order to apply these models to our study, we need to provide information on the cooking workflow. We believe that the annotations of the VRF dataset would be useful for this purpose.

Converting whole procedural texts into plans In this thesis, we have evaluated ViLaIn with single instructions. However, actual cooking recipes consist of more instructions with complex dependencies of actions. We leave the investigation of whether ViLaIn can capture those dependencies and find valid plans for future work. In addition, cooking recipes provide quantitative information not considered in this thesis, such as duration of actions and quantity of foods. We may tackle these problems by using PDDLs that include such numerical information. Further, cooking instructions written for humans are often ambiguous or lack some information for execution, and these also make robot execution of cooking recipes challenging. Handling such tacit knowledge by machines is a challenging but interesting theme.

Bibliography

Shinsuke Mori, Hirokuni Maeta, Yoko Yamakata, and Tetsuro Sasada. Flow graph corpus from recipe texts. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation*, pages 2370–2377, 2014.

Yoko Yamakata, Shinsuke Mori, and John A Carroll. English recipe flow graph corpus. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 5187–5194, 2020.

Hirokuni Maeta, Tetsuro Sasada, and Shinsuke Mori. A framework for procedural text understanding. In *Proceedings of the 14th International Conference on Parsing Technologies*, pages 50–60, 2015.

Chloé Kiddon, Ganesa Thandavam Ponnuraj, Luke Zettlemoyer, and Yejin Choi. Mise en place: Unsupervised interpretation of instructional recipes. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 982–992, 2015.

Tomáš Souček, Jean-Baptiste Alayrac, Antoine Miech, Ivan Laptev, and Josef Sivic. Look for the change: Learning object states and state-modifying actions from untrimmed web videos. *arXiv preprint arXiv:2203.11637*, 2022.

Tomáš Souček, Dima Damen, Michael Wray, Ivan Laptev, and Josef Sivic. Genhowto: Learning to generate actions and state transformations from instructional videos. *arXiv preprint arXiv:2312.07322*, 2023.

Matt MacMahon, Brian Stankiewicz, and Benjamin Kuipers. Walk the talk: connecting language, knowledge, and action in route instructions. In *Proceedings of*

the 21st National Conference on Artificial Intelligence - Volume 2, AAAI'06, page 1475–1482. AAAI Press, 2006.

Cynthia Matuszek, Dieter Fox, and Karl Koscher. Following directions using statistical machine translation. In *Proceedings of the 2010 5th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 251–258, 2010.

Kevin Lin, Christopher Agia, Toki Migimatsu, Marco Pavone, and Jeannette Bohg. Text2Motion: From natural language instructions to feasible plans. *arXiv preprint arXiv:2303.12153*, 2023.

Reiko Hamada, Jun Okabe, Ichiro Ide, Shin'ichi Satoh, Shuichi Sakai, and Hidehiko Tanaka. Cooking navi: Assistant for daily cooking in kitchen. In *Proceedings of the 13th Annual ACM International Conference on Multimedia*, page 371–374. Association for Computing Machinery, 2005.

Atsushi Hashimoto, Naoyuki Mori, Takuya Funatomi, Yoko Yamakata, Koh Kakusho, and Michihiko Minoh. Smart kitchen: A user centric cooking support system. In *Proceedings of the 12th Information Processing and Management of Uncertainty in Knowledge-Based Systems*, volume 8, pages 848–854, 2008.

Mario Bollini, Stefanie Tellex, Tyler Thompson, Nicholas Roy, and Daniela Rus. Interpreting and executing recipes with a cooking robot. In *Experimental Robotics*, pages 481–495. Springer, 2013.

Naruki Yoshikawa, Marta Skreta, Kourosh Darvish, Sebastian Arellano-Rubach, Zhi Ji, Lasse Bjørn Kristensen, Andrew Zou Li, Yuchi Zhao, Haoping Xu, Artur Kuramshin, et al. Large language models for chemistry robotics. *Autonomous Robots*, pages 1–30, 2023.

Yoshio Momouchi. Control structures for actions in procedural texts and PT-chart. In *Proceedings of the 8th International Conference on Computational Linguistics*, 1980.

Reiko Hamada, Ichiro Ide, Shuichi Sakai, and Hidehiko Tanaka. Structural analysis of cooking preparation steps in japanese. In *Proceedings of the Fifth International Workshop on on Information Retrieval with Asian Languages*, IRAL '00, page 157–164. Association for Computing Machinery, 2000.

Dan Tasse and Noah A Smith. Sour cream: Toward semantic processing of recipes. *Tech. Rep. CMU-LTI-08-005*, 2008.

Shinsuke Mori, Tetsuro Sasada, Yoko Yamakata, and Koichiro Yoshino. A machine learning approach to recipe text processing. In *Proceedings of the 1st Cooking with Computer Workshop*, pages 29–34, 2012.

Jermsak Jermsurawong and Nizar Habash. Predicting the structure of cooking recipes. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 781–786. Association for Computational Linguistics, 2015.

Yoko Yamakata, John Carroll, and Shinsuke Mori. A comparison of cooking recipe named entities between japanese and english. In *Proceedings of the 9th Workshop on Multimedia for Cooking and Eating Activities in Conjunction with The 2017 International Joint Conference on Artificial Intelligence*, CEA2017, page 7–12. Association for Computing Machinery, 2017.

Mikael Henaff, Jason Weston, Arthur Szlam, Antoine Bordes, and Yann LeCun. Tracking the world state with recurrent entity networks. In *Proceedings of the 2017 International Conference on Learning Representations*, 2017.

Yangfeng Ji, Chenhao Tan, Sebastian Martschat, Yejin Choi, and Noah A. Smith. Dynamic entity representations in neural language models. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1830–1839. Association for Computational Linguistics, 2017.

Ashutosh Modi, Ivan Titov, Vera Demberg, Asad Sayeed, and Manfred Pinkal. Modeling semantic expectation: Using script knowledge for referent prediction. *Transactions of the Association for Computational Linguistics*, 5:31–44, 2017.

Antoine Bosselut, Omer Levy, Ari Holtzman, Corin Ennis, Dieter Fox, and Yejin Choi. Simulating action dynamics with neural process networks. In *Proceedings of the 6th International Conference on Learning Representations*, 2018.

Taichi Nishimura, Atsushi Hashimoto, Yoshitaka Ushiku, Hirotaka Kameko, and Shinsuke Mori. *State-Aware Video Procedural Captioning*, page 1766–1774. Association for Computing Machinery, New York, NY, USA, 2021.

Bhavana Dalvi, Lifu Huang, Niket Tandon, Wen-tau Yih, and Peter Clark. Tracking state changes in procedural text: a challenge dataset and models for process paragraph comprehension. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1595–1604, 2018.

Niket Tandon, Keisuke Sakaguchi, Bhavana Dalvi, Dheeraj Rajagopal, Peter Clark, Michal Guerquin, Kyle Richardson, and Eduard Hovy. A dataset for tracking entities in open domain procedural text. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6408–6417, 2020.

Vittorio Ferrari and Andrew Zisserman. Learning visual attributes. In *Proceedings of the 2007 Advances in Neural Information Processing Systems*, volume 20. Curran Associates, Inc., 2007.

Ali Farhadi, Ian Endres, Derek Hoiem, and David Forsyth. Describing objects by their attributes. In *Proceedings of the 2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1778–1785, 2009.

Devi Parikh and Kristen Grauman. Relative attributes. In *Proceedings of the 2011 International Conference on Computer Vision*, pages 503–510, 2011.

Alireza Fathi and James M. Rehg. Modeling actions through state changes. In *Proceedings of the 2013 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2013.

Phillip Isola, Joseph J. Lim, and Edward H. Adelson. Discovering states and transformations in image collections. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

Jean-Baptiste Alayrac, Ivan Laptev, Josef Sivic, and Simon Lacoste-Julien. Joint discovery of object states and manipulation actions. In *Proceedings of the 2017 IEEE International Conference on Computer Vision (ICCV)*, 2017.

Kristen Grauman, Andrew Westbury, Eugene Byrne, Zachary Chavis, Antonino Furnari, Rohit Girdhar, Jackson Hamburger, Hao Jiang, Miao Liu, Xingyu Liu, et al. Ego4d: Around the world in 3,000 hours of egocentric video. In *Proceedings of the 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 18995–19012, June 2022.

Tomáš Souček, Jean-Baptiste Alayrac, Antoine Miech, Ivan Laptev, and Josef Sivic. Look for the change: Learning object states and state-modifying actions from untrimmed web videos. In *Proceedings of the 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13956–13966, June 2022.

Nirat Saini, Khoi Pham, and Abhinav Shrivastava. Disentangling visual embeddings for attributes and objects. In *Proceedings of the 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13658–13667, June 2022.

Nirat Saini, Hanyu Wang, Archana Swaminathan, Vinoj Jayasundara, Bo He, Kamal Gupta, and Abhinav Shrivastava. Chop & learn: Recognizing and generating object-state compositions. In *Proceedings of the 2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 20247–20258, October 2023.

Stefanie Tellex, Nakul Gopalan, Hadas Kress-Gazit, and Cynthia Matuszek. Robots that use language. *Annual Review of Control, Robotics, and Autonomous Systems*, 3:25–55, 2020.

S.R.K. Branavan, Harr Chen, Luke Zettlemoyer, and Regina Barzilay. Reinforcement learning for mapping instructions to actions. In *Proceedings of the Joint Conference*

of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP, pages 82–90. Association for Computational Linguistics, 2009.

David Chen and Raymond Mooney. Learning to interpret natural language navigation instructions from observations. *Proceedings of the 2011 AAAI Conference on Artificial Intelligence*, 25(1):859–865, Aug. 2011.

Ishika Singh, Valts Blukis, Arsalan Mousavian, Ankit Goyal, Danfei Xu, Jonathan Tremblay, Dieter Fox, Jesse Thomason, and Animesh Garg. Progprompt: Generating situated robot task plans using large language models. In *Proceedings of the 2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11523–11530, 2023.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *Proceedings of the 2020 Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901, 2020.

OpenAI. GPT-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Michael Beetz, Ulrich Klank, Ingo Kresse, Alexis Maldonado, Lorenz Mösenlechner, Dejan Pangercic, Thomas Rühr, and Moritz Tenorth. Robotic roommates making pancakes. In *Proceedings of the 2011 11th IEEE-RAS International Conference on Humanoid Robots*, pages 529–536, 2011.

Ashvin V Nair, Vitchyr Pong, Murtaza Dalal, Shikhar Bahl, Steven Lin, and Sergey Levine. Visual reinforcement learning with imagined goals. In *Proceedings of the*

2018 Advances in Neural Information Processing Systems, volume 31. Curran Associates, Inc., 2018.

Ivan Kapelyukh, Vitalis Vosylius, and Edward Johns. Dall-e-bot: Introducing web-scale diffusion models to robotics. *IEEE Robotics and Automation Letters*, 8(7): 3956–3963, 2023.

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 Language Resources and Evaluation Conference*, pages 4275–4284, 2020.

Liang-Ming Pan, Jingjing Chen, Jianlong Wu, Shaoteng Liu, Chong-Wah Ngo, Min-Yen Kan, Yugang Jiang, and Tat-Seng Chua. *Multi-Modal Cooking Workflow Construction for Food Recipes*, page 1132–1141. Association for Computing Machinery, 2020.

Wenlong Huang, Pieter Abbeel, Deepak Pathak, and Igor Mordatch. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of the 2022 International Conference on Machine Learning (ICML)*, pages 9118–9147, 2022.

Malte Helmert. The fast downward planning system. *Journal of Artificial Intelligence Research*, 26:191–246, 2006.

Yixin Zhang, Yoko Yamakata, and Keishi Tajima. Mirecipe: A recipe dataset for stage-aware recognition of changes in appearance of ingredients. In *Proceedings of the 3rd ACM International Conference on Multimedia in Asia*, pages 1–7. Association for Computing Machinery, 2021.

Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R Brown, Adam Santoro, Aditya Gupta, Adrià Garriga-Alonso, et al. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *arXiv preprint arXiv:2206.04615*, 2022.

Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katie Millican, Malcolm Reynolds, et al. Flamingo: a visual language model for few-shot learning. *arXiv preprint arXiv:2204.14198*, 2022.

Graham Neubig, Yosuke Nakata, and Shinsuke Mori. Pointwise prediction for robust, adaptable japanese morphological analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 529–533, 2011.

Antoine Miech, Dimitri Zhukov, Jean-Baptiste Alayrac, Makarand Tapaswi, Ivan Laptev, and Josef Sivic. Howto100m: Learning a text-video embedding by watching hundred million narrated video clips. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2630–2640, 2019.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. Neural architectures for named entity recognition. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 260–270. Association for Computational Linguistics, 2016.

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

Vassileios Balntas, Edgar Riba, Daniel Ponsa, and Krystian Mikolajczyk. Learning local feature descriptors with triplets and shallow convolutional neural networks. In *Proceedings of the British Machine Vision Conference*, pages 119.1–119.11, September 2016.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *Proceedings of the 7th International Conference on Learning Representations*, 2019.

Richard Socher, Andrej Karpathy, Quoc V Le, Christopher D Manning, and Andrew Y Ng. Grounded compositional semantics for finding and describing images with sentences. *Transactions of the Association for Computational Linguistics*, pages 207–218, 2014.

Amaia Salvador, Nicholas Hynes, Yusuf Aytar, Javier Marin, Ferda Ofli, Ingmar Weber, and Antonio Torralba. Learning cross-modal embeddings for cooking recipes and food images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

Rowan Zellers, Yonatan Bisk, Ali Farhadi, and Yejin Choi. From recognition to cognition: Visual commonsense reasoning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, June 2019a.

Semih Yagcioglu, Aykut Erdem, Erkut Erdem, and Nazli Ikizler-Cinbis. RecipeQA: A challenge dataset for multimodal comprehension of cooking recipes. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1358–1368, 2018.

Malihe Alikhani, Sreyasi Nag Chowdhury, Gerard de Melo, and Matthew Stone. CITE: A corpus of image-text discourse relations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 570–575, 2019.

Atsushi Ushiku, Hayato Hashimoto, Atsushi Hashimoto, and Shinsuke Mori. Procedural text generation from an execution video. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 326–335, 2017.

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*, pages 409–414, 2019.

Jun Hatori, Yuta Kikuchi, Sosuke Kobayashi, Kuniyuki Takahashi, Yuta Tsuboi, Yuya Unno, Wilson Ko, and Jethro Tan. Interactively picking real-world objects with unconstrained spoken language instructions. In *Proceedings of the 2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3774–3781, 2018.

Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and Andy Zeng. Code as policies: Language model programs for embodied control. In *Proceedings of the 2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9493–9500, 2023.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734. Association for Computational Linguistics, 2014.

Dilip Arumugam, Siddharth Karamcheti, Nakul Gopalan, Lawson Wong, and Stefanie Tellex. Accurately and efficiently interpreting human-robot instructions of varying granularities. In *Proceedings of the 2017 Robotics: Science and Systems (RSS)*, 2017.

Chris Paxton, Yonatan Bisk, Jesse Thomason, Arunkumar Byravan, and Dieter Foxl. Prospection: Interpretable plans from language by predicting the future. In *Proceedings of the 2019 International Conference on Robotics and Automation (ICRA)*, pages 6942–6948, 2019.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

Rohan Anil, Andrew M Dai, Orhan Firat, Melvin Johnson, Dmitry Lepikhin, Alexandre Passos, Siamak Shakeri, Emanuel Taropa, Paige Bailey, Zhifeng Chen, et al. PaLM 2 technical report. *arXiv preprint arXiv:2305.10403*, 2023.

Shreyas Sundara Raman, Vanya Cohen, Eric Rosen, Ifrah Idrees, David Paulius, and Stefanie Tellex. Planning with large language models via corrective re-prompting. In *NeurIPS 2022 Foundation Models for Decision Making Workshop*, 2022.

Leilani H Gilpin, David Bau, Ben Z Yuan, Ayesha Bajwa, Michael Specter, and Lalana Kagal. Explaining explanations: An approach to evaluating interpretability of machine learning. *arXiv preprint arXiv:1806.00069*, page 118, 2018.

Erez Karpas and Daniele Magazzeni. Automated planning for robotics. *Annual Review of Control, Robotics, and Autonomous Systems*, 3(1):417–439, 2020.

Patrik Haslum, Nir Lipovetzky, Daniele Magazzeni, Christian Muise, Ronald Brachman, Francesca Rossi, and Peter Stone. *An introduction to the planning domain definition language*, volume 13. Springer, 2019.

Maria Fox and Derek Long. PDDL2.1: An extension to PDDL for expressing temporal planning domains. *Journal of artificial intelligence research*, 20:61–124, 2003.

Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Chunyuan Li, Jianwei Yang, Hang Su, Jun Zhu, et al. Grounding DINO: Marrying DINO with grounded pre-training for open-set object detection. *arXiv preprint arXiv:2303.05499*, 2023a.

Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. BLIP-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *Proceedings of the 2023 International Conference on Machine Learning (ICML)*, volume 202 of *Proceedings of Machine Learning Research*, pages 19730–19742, 2023.

Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman,

et al. Do as I can, not as I say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*, 2022.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, et al. RT-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*, 2023.

Bo Liu, Yuqian Jiang, Xiaohan Zhang, Qiang Liu, Shiqi Zhang, Joydeep Biswas, and Peter Stone. LLM+P: Empowering large language models with optimal planning proficiency. *arXiv preprint arXiv:2304.11477*, 2023b.

Pratyusha Sharma, Antonio Torralba, and Jacob Andreas. Skill induction and planning with latent language. In *Proceedings of the 2022 Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1713–1726. Association for Computational Linguistics, 2022.

Yaqi Xie, Chen Yu, Tongyao Zhu, Jinbin Bai, Ze Gong, and Harold Soh. Translating natural language to planning goals with large-language models. *arXiv preprint arXiv:2302.05128*, 2023.

Blai Bonet and Héctor Geffner. Planning as heuristic search. *Artificial Intelligence*, 129(1):5–33, 2001.

Caelan Reed Garrett, Tomás Lozano-Pérez, and Leslie Pack Kaelbling. PDDLStream: Integrating symbolic planners and blackbox samplers via optimistic adaptive planning. *Proceedings of the 2020 International Conference on Automated Planning and Scheduling (ICAPS)*, 30(1):440–448, 2020.

Seyed Reza Ahmadzadeh, Ali Paikan, Fulvio Mastrogiovanni, Lorenzo Natale, Petar Kormushev, and Darwin G. Caldwell. Learning symbolic representations of actions from human demonstrations. In *Proceedings of the 2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3801–3808, 2015.

Zi Wang, Caelan Reed Garrett, Leslie Pack Kaelbling, and Tomás Lozano-Pérez. Learning compositional models of robot skills for task and motion planning. *The International Journal of Robotics Research*, 40(6-7):866–894, 2021.

Tom Silver, Rohan Chitnis, Joshua Tenenbaum, Leslie Pack Kaelbling, and Tomás Lozano-Pérez. Learning symbolic operators for task and motion planning. In *Proceedings of the 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3182–3189, 2021.

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Proceedings of the 2015 Advances in Neural Information Processing Systems (NeurIPS)*, volume 28, 2015.

Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

Alireza Zareian, Kevin Dela Rosa, Derek Hao Hu, and Shih-Fu Chang. Open-vocabulary object detection using captions. In *Proceedings of the 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14393–14402, 2021.

Cewu Lu, Ranjay Krishna, Michael Bernstein, and Li Fei-Fei. Visual relationship detection with language priors. In *Proceedings of the 2016 European Conference on Computer Vision (ECCV)*, pages 852–869. Springer, 2016.

Sho Inayoshi, Keita Otani, Antonio Tejero-de Pablos, and Tatsuya Harada. Bounding-box channels for visual relationship detection. In *Proceedings of the 2020 European Conference on Computer Vision (ECCV)*, pages 682–697. Springer, 2020.

Danfei Xu, Yuke Zhu, Christopher B Choy, and Li Fei-Fei. Scene graph generation by iterative message passing. In *Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR))*, pages 5410–5419, 2017.

Jingkang Yang, Yi Zhe Ang, Zujin Guo, Kaiyang Zhou, Wayne Zhang, and Ziwei Liu. Panoptic scene graph generation. In *Proceedings of the 2022 European Conference on Computer Vision (ECCV)*, pages 178–196. Springer, 2022.

Toki Migimatsu and Jeannette Bohg. Grounding predicates through actions. In *Proceedings of the 2022 International Conference on Robotics and Automation (ICRA)*, pages 3498–3504. IEEE, 2022.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Proceedings of the 2022 Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, pages 24824–24837. Curran Associates, Inc., 2022.

Takeshi Kojima, Shixiang (Shane) Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. In *Proceedings of the 2022 Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, pages 22199–22213. Curran Associates, Inc., 2022.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc V Le, et al. Least-to-most prompting enables complex reasoning in large language models. In *Proceedings of the 2023 International Conference on Learning Representations (ICLR)*, 2023a.

Naresh Gupta and Dana S Nau. On the complexity of blocks-world planning. *Artificial intelligence*, 56(2-3):223–254, 1992.

Ronald Alford, Ugur Kuter, and Dana S Nau. Translating HTNs to PDDL: A small amount of domain knowledge can go a long way. In *Proceedings of the 2009 International Joint Conference on Artificial Intelligence (IJCAI)*, volume 9, pages 1629–1634, 2009.

Tom Silver and Rohan Chitnis. PDDLGym: Gym environments from PDDL problems. In *Proceedings of the 2020 International Conference on Automated Planning and Scheduling (ICAPS) PRL Workshop*, 2020.

Joshua Maynez, Shashi Narayan, Bernd Bohnet, and Ryan McDonald. On faithfulness and factuality in abstractive summarization. In *Proceedings of the 2020 Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1906–1919. Association for Computational Linguistics, 2020.

Li Zhang, Qing Lyu, and Chris Callison-Burch. Reasoning about goals, steps, and temporal ordering with WikiHow. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pages 4630–4639. Association for Computational Linguistics, 2020a.

Yoeng-Jin Chu and Tseng-Hong Liu. On the shortest arborescence of a directed graph. *Science Sinica*, 14:1396–1400, 1965.

Jack Edmonds. Optimum branchings. *Journal of Research of the National Bureau of Standards: Mathematics and mathematical physics. B*, 71:233–240, 1967.

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajič. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 523–530. Association for Computational Linguistics, 2005.

Haoran Xu, Seth Ebner, Mahsa Yarmohammadi, Aaron Steven White, Benjamin Van Durme, and Kenton Murray. Gradual fine-tuning for low-resource domain adaptation. In *Proceedings of the Second Workshop on Domain Adaptation for NLP*, pages 214–221. Association for Computational Linguistics, 2021.

Marzieh Fadaee, Arianna Bisazza, and Christof Monz. Data augmentation for low-resource neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 567–573. Association for Computational Linguistics, 2017.

Bosheng Ding, Linlin Liu, Lidong Bing, Canasai Kruengkrai, Thien Hai Nguyen,

Shafiq Joty, Luo Si, and Chunyan Miao. DAGA: Data augmentation with a generation approach for low-resource tagging tasks. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6045–6057. Association for Computational Linguistics, 2020.

Xiang Dai and Heike Adel. An analysis of simple data augmentation for named entity recognition. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 3861–3867. International Committee on Computational Linguistics, 2020.

Yilun Zhou, Julie Shah, and Steven Schockaert. Learning household task knowledge from WikiHow descriptions. In *Proceedings of the 5th Workshop on Semantic Deep Learning (SemDeep-5)*, pages 50–56. Association for Computational Linguistics, 2019.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4791–4800. Association for Computational Linguistics, 2019b.

Shuyan Zhou, Li Zhang, Yue Yang, Qing Lyu, Pengcheng Yin, Chris Callison-Burch, and Graham Neubig. Show me more details: Discovering hierarchies of procedures from semi-structured web data. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2998–3012. Association for Computational Linguistics, 2022.

Xudong Lin, Fabio Petroni, Gedas Bertasius, Marcus Rohrbach, Shih-Fu Chang, and Lorenzo Torresani. Learning to recognize procedural activities with distant supervision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13853–13863, June 2022.

Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D. Manning. Stanza: A python natural language processing toolkit for many human languages. In *Proceedings of the 58th Annual Meeting of the Association for Computational*

Linguistics: System Demonstrations, pages 101–108. Association for Computational Linguistics, 2020.

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*, pages 3570–3577. International Committee on Computational Linguistics, 2022.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. Deberta: decoding-enhanced bert with disentangled attention. In *9th International Conference on Learning Representations*, 2021.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics, 2019.

Timothy Dozat and Christopher D. Manning. Simpler but more accurate semantic dependency parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 484–490. Association for Computational Linguistics, 2018.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

Iz Beltagy, Matthew E Peters, and Arman Cohan. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*, 2020.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations. In *Proceedings of the 8th International Conference on Learning Representations*, 2020.

Dim P Papadopoulos, Enrique Mora, Nadiia Chepurko, Kuan Wei Huang, Ferda Ofli, and Antonio Torralba. Learning program representations for food images and cooking recipes. *arXiv preprint arXiv:2203.16071*, 2022.

Chaitanya Kulkarni, Wei Xu, Alan Ritter, and Raghu Machiraju. An annotated corpus for machine reading of instructions in wet lab protocols. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 97–106. Association for Computational Linguistics, 2018.

Ronen Tamari, Fan Bai, Alan Ritter, and Gabriel Stanovsky. Process-level representation of scientific protocols with interactive annotation. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 2190–2202. Association for Computational Linguistics, 2021.

Sheshera Mysore, Zachary Jensen, Edward Kim, Kevin Huang, Haw-Shiuan Chang, Emma Strubell, Jeffrey Flanigan, Andrew McCallum, and Elsa Olivetti. The materials science procedural text corpus: Annotating materials synthesis procedures with shallow semantic structures. In *Proceedings of the 13th Linguistic Annotation Workshop*, pages 56–64. Association for Computational Linguistics, 2019.

Fusataka Kuniyoshi, Kohei Makino, Jun Ozawa, and Makoto Miwa. Annotating and extracting synthesis process of all-solid-state batteries from scientific literature. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 1941–1950. European Language Resources Association, 2020.

Li Zhang, Qing Lyu, and Chris Callison-Burch. Intent detection with WikiHow. In *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing*, pages 328–333. Association for Computational Linguistics, 2020b.

Honglu Zhou, Roberto Martín-Martín, Mubbasir Kapadia, Silvio Savarese, and

Juan Carlos Niebles. Procedure-aware pretraining for instructional video understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10727–10738, June 2023b.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Proceedings of the 2020 Advances in Neural Information Processing Systems*, volume 33, pages 6840–6851. Curran Associates, Inc., 2020.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10684–10695, June 2022.

Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. In *Proceedings of the 2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 3836–3847, October 2023.

Tim Brooks, Aleksander Holynski, and Alexei A. Efros. Instructpix2pix: Learning to follow image editing instructions. In *Proceedings of the 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 18392–18402, June 2023.

Authored Works

Chapter 2

1. Keisuke Shirai, Atsushi Hashimoto, Taichi Nishimura, Hirotaka Kameko, Shuhei Kurita, Yoshitaka Ushiku, and Shinsuke Mori (2022). "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 (COLING)*, pp. 3570–3577.

2. Keisuke Shirai, Atushi Hashimoto, Taichi Nishimura, Hirotaka Kameko, Shuhei Kurita, and Shinsuke Mori (2023). "Constructing and Evaluating the Visual Recipe Flow Dataset for Predicting Visual Observations After Cooking Actions." In *Journal of Natural Language Processing*, 2023, Volume 30, Issue 3, pp. 1042-1060.

Chapter 3

1. Keisuke Shirai, Cristian C. Beltran-Hernandez, Masashi Hamaya, Atsushi Hashimoto, Shohei Tanaka, Kento Kawaharazuka, Kazutoshi Tanaka, Yoshitaka Ushiku, and Shinsuke Mori (2024). "Vision-Language Interpreter for Robot Task Planning." To appear at *the 2024 IEEE International Conference on Robotics and Automation (ICRA)*.

Chapter 4

1. Keisuke Shirai, Hirotaka Kameko, and Shinsuke Mori (2023). "Towards Flow Graph Prediction of Open-Domain Procedural Texts." In *Proceedings of the 8th Workshop on Representation Learning for NLP (RepL4NLP 2023)*, pp. 87–96.

2. Keisuke Shirai, Hirotaka Kameko, and Shinsuke Mori (2024). "Flow Graph Prediction of Open-Domain Procedural Texts." Under review at *Journal of Natural Language Processing*.

Others

1. Keisuke Shirai, Kazuma Hashimoto, Akiko Eriguchi, Takashi Ninomiya, and Shinsuke Mori (2020). "Neural Text Generation with Artificial Negative Examples." In *arXiv preprint, arXiv:2012.14124*.

2. Keisuke Shirai, Kazuma Hashimoto, Akiko Eriguchi, Takashi Ninomiya, and Shinsuke Mori (2021). "Neural Text Generation with Artificial Negative Examples to Address Repeating and Dropping Errors." In *Journal of Natural Language Processing*, 2021, Volume 28, Issue 3, pp. 751-777.

3. Keisuke Shirai, Masato Matsuzaki, Shinsuke Mori, and Makoto Goto (2022). "Knowledge Extraction from a Biographical Dictionary." In *IPSJ Journal*, 2022, Volume 63, No. 2, pp. 293-301.

4. Kento Tanaka, Taichi Nishimura, Hiroaki Nanjo, Keisuke Shirai, Hirotaka Kameko, and Masatake Dantsuji (2022). "Image Description Dataset for Language Learners." In *Proceedings of the Thirteenth Language Resources and Evaluation Conference (LREC)*, pp. 6814–6821.