Do BERTs Learn to Use Browser User Interface? Exploring Multi-Step Tasks with Unified Vision-and-Language BERTs

Anonymous ACL submission

Abstract

Unifying models by reducing task-specific structures have been studied to facilitate the transfer of learned knowledge. A text-to-text framework has pushed the unification of the model. However, the framework remains limited because it does not allow contents with a layout for input and has a basic assumption that the task can be solved in a single step. To address these limitations, in this paper, we explore a new framework in which a model performs a task by manipulating displayed web pages in multiple steps. We develop two types of task web pages with different levels of difficulty and propose a BERT extension for the framework. We trained the BERT extension with those task pages jointly, and the following observations were made. (1) The model maintains its performance greater than 80% of that of the original BERT separately fine-tuned in a single-step framework in five out of six tasks. (2) The model learned to solve both tasks of difficulty level. (3) The model did not generalize effectively on unseen tasks. These results suggest that although room for improvement exists, we can transfer BERTs to multi-step tasks, such as using graphical user interfaces.

1 Introduction

Prior studies have attempted to unify models for processing natural language to facilitate the transfer of learned knowledge by reducing task-specific structures. For example, Radford et al. (2018); Devlin et al. (2019) suggest that language models with a generic structure, Transformer (Vaswani et al., 2017), are effective. Raffel et al. (2020) proposed a text-to-text framework which converts tasks into a problem where a model receives and generates text. Cho et al. (2021) extended the input of the text-to-text framework to accommodate images.

However, existing research on unified models remains limited. First, the models proposed by Cho et al. (2021) use a linear sequence of text and several images as input. However, they are not designed to handle input with a layout. Second, existing unified models assume single-step tasks. Task-specific design still must be completed when applying these models to compound tasks, such as reading a single document and subsequently searching for missing information. The latter challenge is more difficult to address because methods for using a transformer in multiple-step tasks, have not yet been fully established. Although transformer-based models have been successful in many language-related tasks such as language understanding (Wang et al., 2019), question answering (Rajpurkar et al., 2016), visual question answering (Antol et al., 2015), and referring expression comprehension (Kazemzadeh et al., 2014), nevertheless, these are single-step tasks.

In this study, we investigate the following research question to address this limitation: *Can models complete tasks using user interfaces (UIs) that are integrated with visual input content?* We pro-

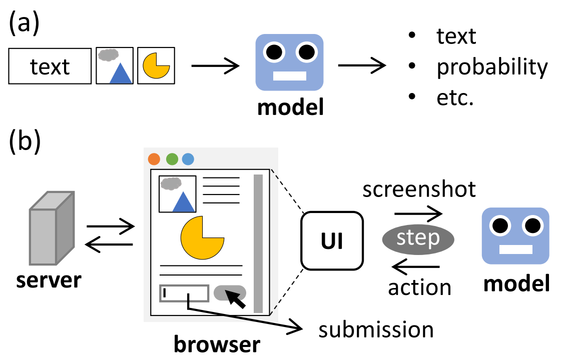


Figure 1: Comparison of task frameworks. (a) Conventional frameworks assume single-step tasks in which a model takes a sequence of text and images to generate an output. (b) In our framework, we make a task as web pages, which allow structured contents, hyperlinks and scripts. The page design decides how to *submit* an answer (e.g., choose a button or input text). A model completes a task in *multiple steps* using the browser user interface (BUI). The model take a screenshot to output an action for each step. (e.g., click or keystroke).

pose a task framework in which tasks are written as web pages, and models complete those tasks via browser UI (Figure 1). The essence of our study is the model that uses graphical UIs. The reason we chose a browser instead of other options such as an operating systems is that web pages are easier to create than native software and browsers are connectable to real services.

We formulate the interaction between a browser and a model (§ 3), and create task pages based on the existing datasets, including GLUE (Wang et al., 2019), SQuAD (Rajpurkar et al., 2016) and VQA (Antol et al., 2015) (§ 4). Our formulation ensures that the model actions are general and primitive to enable further extension. Our tasks include not only single-page but also multi-page tasks that require page jumps to diversify the goal of actions. We introduce a BERT (Devlin et al., 2019) extension with a simple memory mechanism and pre-training for actions (§ 5). In our experiments, we train our model in a multi-task setting. We validate whether our model can learn in the framework and compare it with the models in other framework based on the same BERT. We show that our pre-training and memory mechanisms are effective and analyze the models' ability to solve unseen tasks (§ 6). Code will be available online[1].

Our contributions:

- We propose a framework, in which unified models perform tasks with a browser UI, that enables the study of models involving multi-step tasks.
- By designing multi-page tasks that require page transition, we demonstrate how the proposed framework can expand the task landscape.
- We introduced a BERT extension and demonstrate its ability to learn diverse tasks (GLUE, SQuAD, VQA, and multi-page tasks) jointly.

2 Related Work

2.1 Execution Style of Unified Models

Unified Models aim to reduce task-specific structures to promote learning different tasks jointly such that learned knowledge can be shared between tasks[2]. After the success of transformer-based language models (LMs) (Devlin et al., 2019; Radford et al., 2019) and their visual extensions (Lu et al., 2019; Li et al., 2019a; Tan and Bansal, 2019; Chen et al., 2020; Su et al., 2020), unified models with transformers have received significant attention.

We can categorize unified transformers in terms of task execution: *task-specific head* and *text generation* styles. The **task-specific head** style shares most model weights between tasks and provides a head for each task. ViLBERT-MT (Lu et al., 2020) and UniT (Hu and Singh, 2021) use this style. The **text generation** style employs text generation to bridge the differences in output between tasks. GPT-2 (Radford et al., 2019) and GPT-3 (Brown et al., 2020) show that large pre-trained models can multitask in the text region by changing the prompt. T5 (Raffel et al., 2020) and VL-T5 (Cho et al., 2021), which extends T5 to vision, also employ this style. The text generation style can be applied to, in principle, all tasks that can be expressed in a text-to-text format. Our framework is in this line of study. A model manipulates web pages that define a task via general actions. As a result, it extends the tasks to what can be rendered in a browser screen while keeping the model structure. We refer to this style as the **BUI action** style.

2.2 Vision-and-Language Tasks

Document AI is a technique for automatically reading, understanding, and analyzing documents (Cui et al., 2021). Our work relates to studies on HTML documents. Tanaka et al. (2021); Chen et al. (2021) proposed reading comprehension datasets on web pages. Wu et al. (2021); Li et al. (2021a) proposed pretrained models for HTML documents. Although documents are processed differently (such as using screenshots or incorporating hierarchy of the elements), prior studies were concerned with a visually rich layout. Our focus is on the interaction between models and the documents.

UI modeling is an emerging topic, and Bai et al. (2021); He et al. (2021) have pre-trained UI models for mobile devices to obtain better representations for the UI in terms of understanding tasks, such as predicting the application type or retrieving similar UI components. Li et al. (2021b) proposed a multi-task UI model that can answer questions about the UI. While the questions include commands e.g., 'Go to the next screen', they are limited to single-step commands. By contrast, our models use UIs by recurrently generating actions.

Vision-and-language navigation (VLN) (Anderson et al., 2018; Das et al., 2018; Shridhar et al., 2020) studies models that follow instructions in

[1] https://url.will.be.replaced/

[2] While it is a kind of multi-task learning (Caruana, 1997; Ruder, 2017), it often does not have the central tasks.

a physical space, such as room. VLN tasks have progressed in action generation with V&L models. Recent studies used pre-trained LMs to encode instructions (Li et al., 2019b; Majumdar et al., 2020; Hong et al., 2021; Qi et al., 2021). However, the visual input rarely contains long text because the target is a physical space. Combination of views with a long text and actions remains a challenge.

3 Task Formulation with Browser UI

In this study, the term browser refers to software for accessing web pages. A browser renders web pages, navigates to a new page when a hyperlink is clicked, and executes the scripts on a page internally.

Our formulation focuses on browsers that run on personal computers[3]. We assume that the browser input devices are a mouse and keyboard, and that the browser provides a screenshot. At each step, the model partially observes the state of web pages from a screenshot to output an action. The cursor position is drawn as a dot in the screenshot. We apply the action to the browser and waited for a period of time ($\sim$ 500ms)[4] for the browser to complete internal computation (e.g., rendering, navigating). Subsequently, we take the next screenshot. In conclusion, suppose a screenshot of the visible area of a page s_i and model's action a_i at step i, then the model predicts a_i from s_i:

$$a_i = \mathrm{Model}(s_i),\ s_{i+1} = \mathrm{Browser}(a_i).$$

Note that the current framework does not support video or audio contents that progress independently of the model's actions owing to this formulation.

Fixed-size screenshot. In lieu of inputting a whole page by using a screenshot with variable size or scale, we use fixed-size screenshots and give the models actions to move their visible area. Such actions are suitable for pages that dynamically load additional parts and avoid unexpected long inputs.

Actions. Table 1 presents the actions defined. The actions cover using a mouse, keystrokes and moving the visible area. The unit of keystrokes is the model's vocabulary. A model selects one action for each step. Thus, if a task requires inputting a sentence to a text box, the model will move the cursor to a text box (MOVETO), click it (CLICK), and enter tokens (TOKEN).

scope	action name	description
mouse	MOVETO(x, y)	move the cursor to (x, y)
mouse	CLICK	right click.
key	TOKEN(word)	type characters in a word
key	SPACE	type space key
key	BACKSPACE	type backspace key
key	ENTER	type enter key
view	LEFT	move the view to the left
view	RIGHT	move the view to the right
view	UP	move the view to the up
view	DOWN	move the view to the down

Table 1: Defined actions. MOVETO and TOKEN take the arguments specified in the parentheses.

4 Task Pages

In the BUI framework, tasks are written as web pages. Although there are no restrictions on the layout of the pages, we used layouts that have an instruction, a main content, and an answer form for simplicity. We assumed that a task example has a single answer. Figure 2 summarizes task pages we made. This section describes the types of task page and how to obtain gold actions for training.

4.1 Types of Task Page

(a) Pre-Training for Actions (PTA). Prior knowledge of interface usage, such as the use of clickable buttons, could assist more efficient learning of tasks in the BUI by avoiding situations where models learn such knowledge and reasoning (e.g., reading comprehension) simultaneously. We introduced pre-training for actions: a set of small tasks that focus on moving the *cursor*, clicking a *button*, inputting *text*, and moving the visible *area*. As shown in Figure 2, in PTA tasks, the instructions are written at the top of the screen, and the model succeeds if it follows the instruction. We generated task instances using templates (in Appendix C.2).

(b) Single-page tasks. To evaluate to what extent models can solve traditional tasks in BUI, we created tasks of this type based on existing datasets. We used GLUE (Wang et al., 2019) and SQuAD (Rajpurkar et al., 2016, 2018) for natural language understanding and VQA (Antol et al., 2015) for visual grounding. Task pages of this type involve scrolling pages and submitting answers. We chose answer forms that matched the format of those datasets. We used buttons for GLUE (classification), and a text box for SQuAD and VQA (question answering). The condition for success is to submit the correct answer of the original datasets.

[3] Firefox (`https://www.mozilla.org/en-US/firefox/`) was adopted as the browser and Selenium (`https://www.selenium.dev/`), which is an automation tool for browser operations, was used to apply the model's actions to the browser.

[4] An internal server was used. Accessing external servers could require additional time.

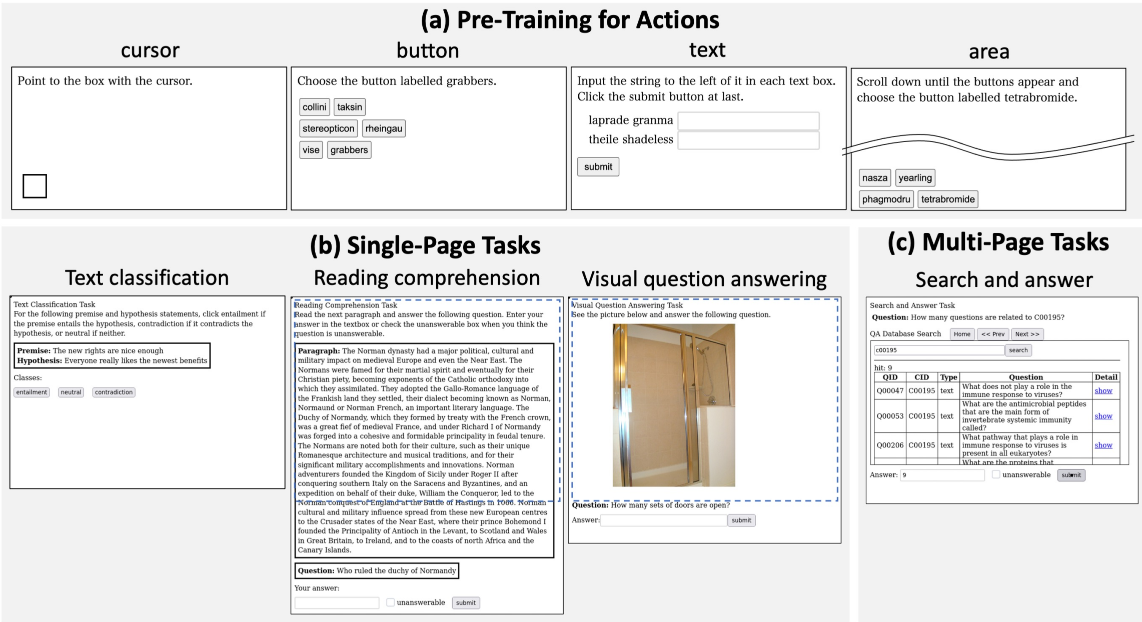


Figure 2: Three types of task page and the examples. (a) Pre-training for actions. In the *area* task, a blank space exists between the instructions and the buttons so that a model needs to scroll until the buttons are visible. (b) Single-page tasks. The rectangles outlined by the blue dotted lines represent the initial visible area. (c) Multi-page tasks. Models can make page transitions within the child frames embedded in the outer page.

(c) Multi-page tasks. This type introduces page transitions to focus more on procedural tasks that BUI enables. We designed Search and Answer (SA) task (Figure 3). For the task, we made databases on question answering tasks by sampling the contexts (paragraphs or images) and questions from SQuAD and VQA. We assigned unique ids to the contexts and questions. Task pages of SA are linked to one of those databases that can be queried with the search UI. The goal of the tasks is to answer a question about the database using the search UI. We prepared four groups to verify whether the models can handle different questions:

- **SA-H**: *How many questions are related to CID?* requires querying a given Context ID (CID) and answering the number of Hits.
- **SA-Q**: *What is the question of QID?* requires identification of the question corresponding to a given Question ID (QID).
- **SA-QID**: *What is the QID of QUESTION?* requires identification of the QID corresponding to a given question.
- **SA-A**: *Answer the question of QID.* requires answering the question corresponding to a given QID.

While SA-H, -Q, and -QID can be answered directly from the search results, SA-A requires models to display a detailed page to produce the answers. Appendix C.3 provides further detail.

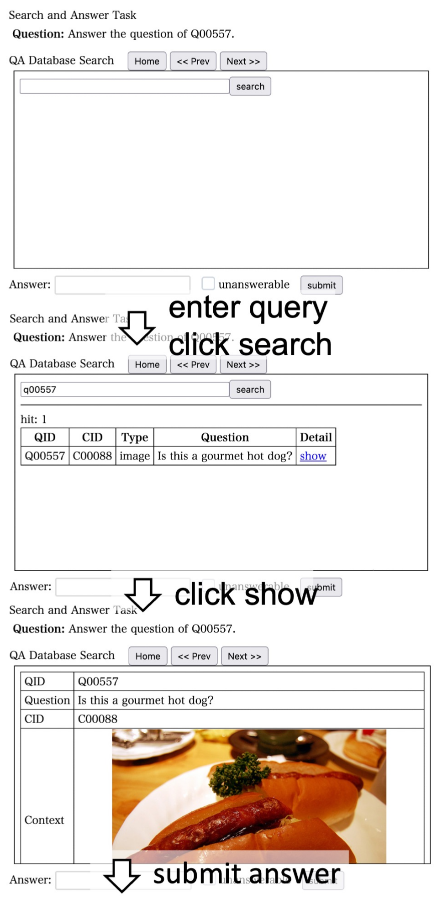


Figure 3: Example pages for the SA-A task. All the SA tasks share the page design. The task pages include an initial page, search result, and detail page. The result changes depending on the query. Models are required to jump between those pages to answer the question.

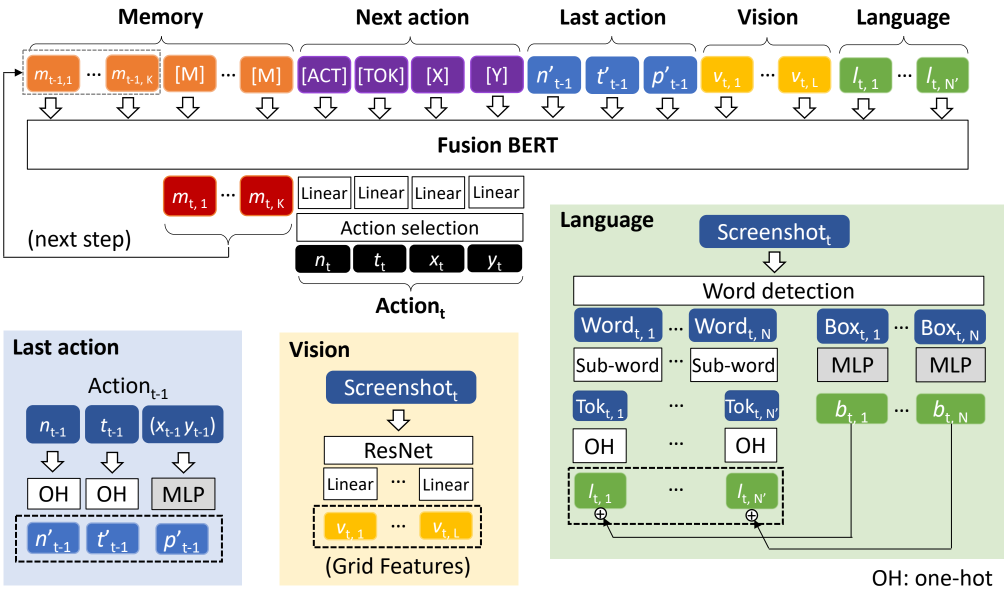


Figure 4: Overview of BUI-BERT. It consists of a pre-trained BERT (Fusion BERT), which takes vision, language, memory, and some auxiliary tokens to output the next action. The model adds position and segment embeddings to each token in the same way as the original BERT (omitted in the view).

4.2 Creating Gold Sequences

Supervised learning was used to train the proposed BUI model because the probability of completing a task by randomly acting on a page appears small. To record gold sequences of action-screenshot pairs, we manually created rules for each task and manipulated task pages loaded in a browser following the rules. We designed the rules to identify the contents on a task page once and to take actions to submit answers.

Individual rules are listed as follows. Note that each rule breaks down further into the actions.

- **GLUE.** Scroll down until the buttons appear (to view all contents). Click the correct button.
- **SQuAD and VQA.** Scroll down until the submit form appears. If the question is unanswerable, check unanswerable. Otherwise, type the answer in the text box. Click the submit button.
- **SA-H.** Query the CID in an instruction. Submit the number of hit records.
- **SA-Q.** Query the QID in an instruction. Scroll down until the record related to the query appears. Submit the corresponding question.
- **SA-QID.** We swapped the question and QID in the SA-Q rule. We used only the first three tokens of the question to reduce the number of actions.
- **SA-A.** Query QID in an instruction. Click the 'show' link to display the context. View the entire context to produce the answer for the question corresponding to the QID. Submit the answer.

5 BUI-BERT

This section describes how to extend the pre-trained $\text{BERT}_{\text{small}}$ to manipulate the browser UI. A small language model was used instead of pre-trained professional models with standard size (e.g., LayoutLM) owing to the multiple long sequences required by the BUI setup. As illustrated in Figure 4, vision, language, memory, and some auxiliary tokens are fused with fusion BERT to obtain the next action. We initialized the weight of fusion BERT based on the weight of the pre-trained BERT and pre-trained the model using the PTA tasks.

5.1 Vision Input

We used grid features from a pre-trained convolutional neural network similar to Huang et al. (2020), considering the speed and amount of data. We encoded the screenshot at each step with a frozen pre-trained ResNet (He et al., 2016)[5] followed by a trainable fully connected layer.

5.2 Language Input

To treat words as a separate modality, we detected words from a screenshot and broke down the words into sub-words using the BERT tokenizer. To avoid the necessity to consider detection errors, a word-based OCR was emulated by inserting span tags between words in the HTML pages. While this emulation works both in text content and labels on

[5] Pre-trained ResNet18 bundled with PyTorch Vision.

the buttons, it does not capture text in text boxes. A detection example can be found in Appendix D.1.

Location embedding. We added location embedding to each sub-word embeddings to indicate the location on a screenshot. The word bounding box[6] was encoded by a trainable MLP. Sub-words in a word share the location embedding of the word.

5.3 Memory Mechanism

Completing a task over several steps requires memory. Our memory mechanism used $2 \times K$ embeddings. The first half K was copied from the previous memory output, and the second half was filled with the [M] embedding, a trainable one-hot vector. After fusing inputs, we retained the K encoded embeddings corresponding to the second half for the next step. During training, we inputted the memory embeddings recurrently while the number of steps did not exceed the maximum (50 in our study).

5.4 Auxiliary Inputs

Last action. The last action is represented with the embeddings of the action name, the cursor position, and the sub-word [7]. We used trainable one-hot vectors for the action name and the sub-word embeddings. We encoded the cursor position using the same MLP as the word location [8].

Next action. We appended trainable one-hot vectors for [ACT], [TOK], [X], and [Y] tokens and inputted these tokens to predict the next action.

5.5 Next Action Prediction and Loss

We predicted the next action from the embeddings that the fusion BERT encoded the [ACT], [TOK], [X], and [Y] tokens. Suppose the encoded embedigns are e_{act}, e_{Tok}, e_{x}, and e_{y}. We first classified the action name from e_{act}. We then classified the token id in Fusion BERT's vocabulary from e_{tok} for TOKEN and the pixel coordinate[9] from e_{x} and e_{y} for MOVETO. All embeddings are projected to the class distributions with trainable linear layers. During training, we used the Softmax cross-entropy loss for the action name, token, x, and y. These were evenly added in a mini-batch:

$$L_{\text{mb}} = \langle L_{\text{name}} \rangle + \langle L_{\text{token}} \rangle + \langle L_{\text{x}} \rangle + \langle L_{\text{y}} \rangle,$$

where $\langle \cdot \rangle$ denotes average for non-pad labels.

[6](center x, center y, width, and height). All elements were normalized by the width or height of a screenshot.

[7]For actions unrelated to the cursor position or sub-word, their embeddings were filled with zeros.

[8]Width and height were set to zero

[9]$x \in \{1, ..., \text{screen width}\}$ and $y \in \{1, ..., \text{screen height}\}$.

model	cursor	button	text	area
BUI-BERT$_{\text{small}}$	**1.00**	**0.89**	**0.77**	**0.54**
BUI-BERT$_{\text{medium}}$	**1.00**	0.63	0.66	0.50
chance level	-	0.43	-	0.42

Table 2: Exact match accuracy of BUI-BERTs on Pre-Training for Actions. Models were trained on the four tasks jointly. Chance levels of the button and area tasks were calculated as reciprocals of the number of buttons.

6 Experiments[10]

First, we trained our BUI models on the PTA tasks to pre-train the models. Second, we trained the BUI models in the multi-task setting; thereafter, we compared the BUI models to the models with different task styles. Finally, we analyzed the models.

6.1 Pre-Traing for Actions

We trained small and medium sized BUI-BERTs[11] on PTA tasks jointly with 60k training examples.

Setup. The memory length of both models was 64. We set the screen at 640px×448px and resized the screenshots by half before inputting to ResNet18. The maximum epoch was 50. We tracked the validation loss at the end of each epoch and used the model with the smallest validation loss for the evaluation with the actual browser. During evaluation, the trial was stopped and considered a failure if a model did not submit an answer within 1.5 times the number of steps in the gold sequences. We used the ADAM optimizer (Kingma and Ba, 2014) with a fixed learning rate of 5e-5 and accumulated the gradient such that the mini-batch size was 128.

Results. Table 2 presents the results of the PTA tasks. Our models performed well in the cursor, button, and text tasks. The accuracy on the area task was above the chance level, but it was lower than the button task. The reason could be that the area task requires the models to remember the label in the instruction. This result suggests that room for improvement exists in the memory mechanism.

6.2 Main Tasks

We trained the pre-trained BUI-BERTs on CoLA (Warstadt et al., 2019), STS-B (Cer et al., 2017), MNLI-matched (Williams et al., 2018), SQuADv2, VQAv2, and our SA tasks jointly. The number of training examples were 8.6 k, 5.7 k, 393 k, 130 k, 444 k, and 50 k, respectively. The first three tasks are from the GLUE benchmark.

[10]Note that our results are based on a single run.

[11]Initialized with the pre-trained BERTs from https://github.com/google-research/bert

model	base LM	#params	exec. style	architecture
$\text{BERT}_{\text{small}}$ / +V		31M / 42M	task-spec. head	BERT (Devlin et al., 2019)
$\text{BERT}_{\text{small}}$-s2s+V	$\text{BERT}_{\text{small}}$	74M	text gen.	Enc-dec from Pr. LMs (Rothe et al., 2020)
BUI-$\text{BERT}_{\text{small}}$		42M	BUI action	BUI-BERT (our base BUI model)
BUI-$\text{BERT}_{\text{medium}}$	$\text{BERT}_{\text{medium}}$	54M	BUI action	BUI-BERT (our BUI model)
T5-small+V	T5-small	72M	text gen.	T5 (Raffel et al., 2020)

Table 3: Models to be compared. Model with +V use an image input obtained from a frozen pre-trained ResNet18. Of the #params, ResNet18 and its related layers account for approximately 11M.

model	multi-task	CoLA M	STS-B P	MNLI-m macro f1	VQAv2 acc.	SQuADv2 exact.	SA acc.
$\text{BERT}_{\text{small}}$/+V	*no*	31.3	81.2	75.8	42.9 / 51.4	56.8	-
$\text{BERT}_{\text{small}}$-s2s+V	w/o SA	0.0	82.5	75.5	51.4	47.0	-
BUI-$\text{BERT}_{\text{small}}$	all	-1.0	72.6	70.5	48.1	49.1	63.4
BUI-$\text{BERT}_{\text{medium}}$	all	-2.0	78.2	75.7	48.8	52.2	65.1
T5-small+V	w/o SA	9.5	86.9	81.8	52.4	70.3	-

Table 4: Overall scores on the validation splits. M and P denote Matthews' and Pearson's correlation, respectively.

Compared models. Table 3 shows the summary. **$\text{BERT}_{\text{small}}$/+V**: To estimate the upper bound of performance, we fine-tuned $\text{BERT}_{\text{small}}$ to each task independently, except SA, with task-specific heads. **$\text{BERT}_{\text{small}}$-s2s+V, T5-small+V**: For comparison with text generation models, we prepared an encoder-decoder model whose encoder and decoder weights were initialized based on the weights of $\text{BERT}_{\text{small}}$, and T5-small[12]. We trained those models on all the tasks except for SA jointly. The input sequences were generated such that they provided the most complete information required to solve a task, for example, task description, question, and class labels, using templates (in Appendix B.1).

The models with the suffix +V use an image input for VQA. We obtained the grid features using ResNet18 in a manner similar to BUI-BERTs. We inserted the features into the head of the input embeddings. Appendix B.2 provides further details.

Setup. We trained all models in 10 epochs. We tracked the validation loss at the end of each epoch to select the best model. The other conditions for BUI-BERTs were the same as those for the PTA training. We optimized the hyper-parameters for the compared models (see Appendix B.3).

Results. Table 4 summarizes the results. A comparison between $\text{BERT}_{\text{small}}$/+V and BUI-$\text{BERT}_{\text{small}}$ shows that the performance of BUI-$\text{BERT}_{\text{small}}$ was 80-90% of that of the original $\text{BERT}_{\text{small}}$ fine-tuned on each task separately with classification heads. BUI-$\text{BERT}_{\text{small}}$ obtained lower scores than $\text{BERT}_{\text{small}}$-s2s+V. This indicates that the BUI action style is more challenging than the language generation style; however, the models can learn in the BUI framework. The improvement of BUI-$\text{BERT}_{\text{medium}}$ suggests that LMs with higher performance will give larger gains. T5 can be a candidate owing to the highest scores. We leave for future work the BUI model based on the encoder-decoder LM as our base BERT is an encoder-only LM. Note that the small CoLA scores in the multi-task setting can be due to the relatively small training data. Applying dynamic sampling of examples (Lu et al., 2020) might mitigate the inequality.

6.3 Analysis

Ablation study. We added two models to validate PTA and the memory mechanism. For BUI-$\text{BERT}_{\text{small}}$ w/o PTA, we initialized its weight with $\text{BERT}_{\text{small}}$ and directly trained it on the multi-task training. For BUI-$\text{BERT}_{\text{small}}$ w/o mem, we omitted the memory sequence. This model was re-initialized and trained on PTA. Table 5 shows the results. Ablated models were lower than BUI-$\text{BERT}_{\text{small}}$ on almost all the tasks. This shows that PTA and the memory mechanism is effective. Especially, BUI-$\text{BERT}_{\text{small}}$ w/o mem largely degraded on the SA tasks. This suggests that memory plays important role in the interactive tasks.

SA tasks. BUI-$\text{BERT}_{\text{small/medium}}$ achieved high accuracy on the SA-QID, -Q, and -H as presented in Table 5. By contrast, the accuracy of the SA-A for those models is low. BUI-$\text{BERT}_{\text{small/medium}}$ failed to submit an answer in half/one-third of the cases. This indicates those models didn't fully learn to click on the 'show' hyperlinks. Although BUI-BERTs can learn the procedures that consist of querying text, reading a table, and inputting an answer, this contrast suggests those models need many examples to learn how to use the UI elements.

Unseen tasks. We used three GLUE tasks:

[12]Weights from https://huggingface.co/t5-small

		CoLA	STS-B	MNLI-m	VQAv2	SQuADv2	Search and Answer (SA)				
		-all	-all	-all	-all	-all	-QID	-Q	-H	-A	-all
	#steps	2	2	2.0	5.8	9.2	23.0	30.8	13.1	19.4	24.0
	#cases	1043	1500	9815	214354	11873	1623	1634	182	1561	5000
	metric	M	P	macro f1	acc.	exact	acc.	acc.	acc.	acc.	acc.
BUI-BERT$_{\text{sml.}}$	#sub	1042	1500	9815	213294	10444	1547	1624	172	753	4096
	score	-1.0	**72.6**	**70.5**	**48.1**	**49.1**	**86.1**	**82.7**	**94.0**	**16.1**	**63.4**
w/o PTA	#sub	1040	1500	9813	213206	10254	1359	1622	131	755	3867
	score	-2.0	11.1	68.0	46.0	44.9	49.3	75.5	70.9	5.0	46.0
w/o mem	#sub	1043	1500	9815	212406	8423	679	1210	55	106	1264
	score	**0.0**	50.1	69.0	46.9	32.3	22.1	1.0	10.4	0.4	8.0
BUI-BERT$_{\text{med.}}$	#sub	1041	1500	9814	213562	10335	1541	1627	171	1060	4399
	score	-1.5	78.2	75.8	48.8	52.2	90.9	93.0	72.0	8.1	65.1

Table 5: Ablation study with the validation splits. #steps : the averaged numbers of steps in the gold sequences. #cases : the number of cases evaluated. #sub : the number of cases where the model made a submission. We counted the cases with no submission as failure cases. M and P represent Matthews' and Pearson's correlation, respectively.

(#cor / #sub)	WNLI	MRPC	SST-2
#cases	71	408	872
T5-small+V	0 / 0	0 / 0	124 / 155
BERT$_{\text{small}}$-s2s+V	8 / 14	0 / 0	1 / 2
BUI-BERT$_{\text{small}}$	20 / 35	169 / 238	160 / 359
BUI-BERT$_{\text{medium}}$	9 / 18	11 /31	27 / 57

Table 6: Unseen task evaluation on the validation splits. #sub (#cor) : the number of cases where the model was successful in submitting an answer (a correct answer). #cases : the number of cases evaluated.

WNLI (Levesque et al., 2012), MRPC (Dolan and Brockett, 2005), and SST-2 (Socher et al., 2013). Those were two-choice tasks, and their similarity to the learned tasks was differed. WNLI and MNLI were textual entailment tasks. MRPC and STS-B were equivalence and similarity tasks. SST-2 is a sentiment prediction, which was new to the models. Table 6 presents the results. Nudged by the answer form of buttons, the BUI-BERTs can submit across the tasks. However, the number of times submitted and correct answers was low in all of those task.

7 Discussion

Thus far, we constructed the BUI framework to test whether it can serve as a foundation for unified models. Experiments demonstrated that BUI-BERTs can learn different tasks in a single model using general inputs and outputs and an objective function. Our tasks include multi-step procedures, indicating that BUI models can go beyond the single-step assumption. In particular, the BUI framework could be suitable for the dynamic grounding study (Chandu et al., 2021).

Generalization performance is the key to a unified model that is more valuable than a single model for multiple tasks. As shown in our analysis, the ability of BUI models to complete unseen tasks remained limited. The generalization of BUI models involves both reasoning and procedure. Using larger LMs will be effective if sufficient computational resources are available. Such LMs will improve linguistic reasoning and the understanding of instructions that explain procedures. However, LMs pre-trained on text distribution are not trained to perform procedures and thus need a large amount of training examples to learn procedures. We could obtain the examples by perturbing the task pages we made (e.g., changing the font size and contents position) and converting existing datasets.

Finally, we point out the problem structure behind SA tasks, in which a model transfers some processing to an external program (database search in this case). Our results suggest that transformer LMs with an interactive framework may address this structure. A hierarchical system could be considered with specialized programs and a model that *uses* such program to achieve both performance and generality. Now might be a time to ask the question: To what extent should unified models unify task-related processing in their weights?

8 Conclusion

In this work, we demonstrated that BERT can be applied to a task framework that requires multiple actions to use a browser UI. In multi-task training, our BERT extension with a memory mechanism learned to solve six tasks according to the UI, including hyperlinks, provided by the task pages. Simultaneously, we observed the low ability to solve unseen tasks. It is worth noting that the proposed solution could be limited by the small model size and a lack of diversity of task pages. In future work, we aim to create and evaluate larger models using memory-efficient methods. We hope this study will inspire the future design of unified models.

References

Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian Reid, Stephen Gould, and Anton Van Den Hengel. 2018. Vision-and-Language Navigation: Interpreting Visually-Grounded Navigation Instructions in Real Environments. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3674–3683.

Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C. Lawrence Zitnick, and Devi Parikh. 2015. VQA: Visual Question Answering. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2425–2433.

Chongyang Bai, Xiaoxue Zang, Ying Xu, Srinivas Sunkara, Abhinav Rastogi, Jindong Chen, and Blaise Agüera y Arcas. 2021. UIBert: Learning Generic Multimodal Representations for UI Understanding. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence*, pages 1705–1712.

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language Models are Few-Shot Learners. *arXiv preprint arXiv:2005.14165*.

Rich Caruana. 1997. Multitask Learning. *Machine Learning*, 28:pages 41–75.

Daniel Cer, Mona Diab, Eneko Agirre, Iñigo Lopez-Gazpio, and Lucia Specia. 2017. SemEval-2017 Task 1: Semantic Textual Similarity Multilingual and Crosslingual Focused Evaluation. In *Proceedings of the 11th International Workshop on Semantic Evaluation*, pages 1–14.

Khyathi Raghavi Chandu, Yonatan Bisk, and Alan W Black. 2021. Grounding 'Grounding' in NLP. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 4283–4305.

Xingyu Chen, Zihan Zhao, Lu Chen, JiaBao Ji, Danyang Zhang, Ao Luo, Yuxuan Xiong, and Kai Yu. 2021. WebSRC: A Dataset for Web-Based Structural Reading Comprehension. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 4173–4185.

Yen-Chun Chen, Linjie Li, Licheng Yu, Ahmed El Kholy, Faisal Ahmed, Zhe Gan, Yu Cheng, and Jingjing Liu. 2020. UNITER: Learning UNiversal Image-TExt Representations. In *The 2020 European Conference on Computer Vision*, pages 104–120.

Jaemin Cho, Jie Lei, Hao Tan, and Mohit Bansal. 2021. Unifying Vision-and-Language Tasks via Text Generation. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139, pages 1931–1942.

Lei Cui, Yiheng Xu, Tengchao Lv, and Furu Wei. 2021. Document AI: Benchmarks, Models and Applications. *arXiv preprint arXiv:2111.08609*.

Abhishek Das, Samyak Datta, Georgia Gkioxari, Stefan Lee, Devi Parikh, and Dhruv Batra. 2018. Embodied Question Answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–10.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. 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*, pages 4171–4186.

William B. Dolan and Chris Brockett. 2005. Automatically Constructing a Corpus of Sentential Paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Zecheng He, Srinivas Sunkara, Xiaoxue Zang, Ying Xu, Lijuan Liu, Nevan Wichers, Gabriel Schubiner, Ruby Lee, and Jindong Chen. 2021. ActionBert: Leveraging User Actions for Semantic Understanding of User Interfaces. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 5931–5938.

Yicong Hong, Qi Wu, Yuankai Qi, Cristian Rodriguez-Opazo, and Stephen Gould. 2021. VLN BERT: A Recurrent Vision-and-Language BERT for Navigation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1643–1653.

Ronghang Hu and Amanpreet Singh. 2021. UniT: Multimodal Multitask Learning With a Unified Transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 1439–1449.

Zhicheng Huang, Zhaoyang Zeng, Bei Liu, Dongmei Fu, and Jianlong Fu. 2020. Pixel-BERT: Aligning Image Pixels with Text by Deep Multi-Modal Transformers. *arXiv preprint arXiv:2004.00849*.

Sahar Kazemzadeh, Vicente Ordonez, Mark Matten, and Tamara Berg. 2014. Referitgame: Referring to objects in photographs of natural scenes. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 787–798.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *arXiv preprint arXiv:1412.6980*.

Hector Levesque, Ernest Davis, and Leora Morgenstern. 2012. The Winograd Schema Challenge. In *Thirteenth International Conference on the Principles of Knowledge Representation and Reasoning*.

Junlong Li, Yiheng Xu, Lei Cui, and Furu Wei. 2021a. MarkupLM: Pre-training of Text and Markup Language for Visually-rich Document Understanding. *arXiv preprint arXiv:2110.08518*.

Liunian Harold Li, Mark Yatskar, Da Yin, Cho-Jui Hsieh, and Kai-Wei Chang. 2019a. VisualBERT: A simple and performant baseline for vision and language. *arXiv preprint arXiv:1908.03557*.

Xiujun Li, Chunyuan Li, Qiaolin Xia, Yonatan Bisk, Asli Celikyilmaz, Jianfeng Gao, Noah A Smith, and Yejin Choi. 2019b. Robust Navigation with Language Pretraining and Stochastic Sampling. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1494–1499.

Yang Li, Gang Li, Xin Zhou, Mostafa Dehghani, and Alexey Gritsenko. 2021b. VUT: Versatile UI Transformer for Multi-Modal Multi-Task User Interface Modeling. *arXiv preprint arXiv:2112.05692*.

Jiasen Lu, Dhruv Batra, Devi Parikh, and Stefan Lee. 2019. ViLBERT: Pretraining Task-Agnostic Visiolinguistic Representations for Vision-and-Language Tasks. In *Advances in Neural Information Processing Systems*, volume 32.

Jiasen Lu, Vedanuj Goswami, Marcus Rohrbach, Devi Parikh, and Stefan Lee. 2020. 12-in-1: Multi-Task Vision and Language Representation Learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10437–10446.

Arjun Majumdar, Ayush Shrivastava, Stefan Lee, Peter Anderson, Devi Parikh, and Dhruv Batra. 2020. Improving Vision-and-Language Navigation with Image-Text Pairs from the Web. In *European Conference on Computer Vision*, pages 259–274.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*.

Yuankai Qi, Zizheng Pan, Yicong Hong, Ming-Hsuan Yang, Anton van den Hengel, and Qi Wu. 2021. The Road To Know-Where: An Object-and-Room Informed Sequential BERT for Indoor Vision-Language Navigation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 1655–1664.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. *OpenAI Blog*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language Models are Unsupervised Multitask Learners. *OpenAI blog*, 1(8).

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *Journal of Machine Learning Research*, 21(140):pages 1–67.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know What You Don't Know: Unanswerable Questions for SQuAD. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics Volume 2*, pages 784–789.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ Questions for Machine Comprehension of Text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392.

Sascha Rothe, Shashi Narayan, and Aliaksei Severyn. 2020. Leveraging Pre-Trained Checkpoints for Sequence Generation Tasks. *Transactions of the Association for Computational Linguistics*, 8:pages 264–280.

Sebastian Ruder. 2017. An Overview of Multi-Task Learning in Deep Neural Networks. *arXiv preprint arXiv:1706.05098*.

Mohit Shridhar, Jesse Thomason, Daniel Gordon, Yonatan Bisk, Winson Han, Roozbeh Mottaghi, Luke Zettlemoyer, and Dieter Fox. 2020. ALFRED: A Benchmark for Interpreting Grounded Instructions for Everyday Tasks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10740–10749.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642.

Weijie Su, Xizhou Zhu, Yue Cao, Bin Li, Lewei Lu, Furu Wei, and Jifeng Dai. 2020. VL-BERT: Pre-training of Generic Visual-Linguistic Representations. In *International Conference on Learning Representations*.

Hao Tan and Mohit Bansal. 2019. LXMERT: Learning Cross-Modality Encoder Representations from Transformers. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 5100–5111.

Ryota Tanaka, Kyosuke Nishida, and Sen Yoshida. 2021. VisualMRC: Machine Reading Comprehension on

Document Images. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 13878–13888.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. In *Advances in neural information processing systems*, pages 5998–6008.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. In *International Conference on Learning Representations*.

Alex Warstadt, Amanpreet Singh, and Samuel R. Bowman. 2019. Neural Network Acceptability Judgments. *Transactions of the Association for Computational Linguistics*, 7:pages 625–641.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A Broad-Coverage Challenge Corpus for Sentence Understanding through Inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1*, pages 1112–1122.

Te-Lin Wu, Cheng Li, Mingyang Zhang, Tao Chen, Spurthi Amba Hombaiah, and Michael Bendersky. 2021. LAMPRET: Layout-Aware Multimodal PreTraining for Document Understanding. *arXiv preprint arXiv:2104.08405*.

A Environment Detail

Browser. We used the following environment to render and execute task pages:

- OS: Ubuntu 18.04, 20.04
- Browser: Firefox version 87.0
- Browser driver: geckodriver 0.29.0
- Selenium: version 3.141.0
- Default font (main text): Dejavu Serif, 16px
- Default font (text, button): 13px

Packages and libraries. We used Python 3.6 and PyTorch (1.10) to implement our BUI models. For sequence to sequence models, we used the Transformers library (4.12). To evaluate SQuAD and VQA, we used the public scripts[13] .

Training. We used a NVIDIA V100 GPU with 32 GB VRAM or a NVIDIA RTX 3090 GPU with 24 GM VRAM in each training run.

A.1 Required Time

We report the required time for our experiments in the Table 7.

B Additional Details on Compared Models

B.1 Input Templates

We used templates to make the text inputs for the seq2seq models. Table 8 shows the templates and the rule to fill the sentences in the template for each dataset. We made the templates so that they provided task_type, instruction, content names and content values. In addition, image embeddings are added to the head of sequence for VQA. We made TASK_TYPE and INSTRUCTION with reference to the contents of the data set.

B.2 Image Input

First, we resize an given image 320px$\times$ 224px. If the aspect ratio does not match, we center the image and fill in the missing pixels with black pixel. Second, we input the image to the frozen pre-trained ResNet18 model to obtain the last feature map (C4; 10x7). We flat the feature map in one dimension and input each feature to one fully-connected linear layer, which is trainable, to align the dimension with the hidden dimension of the LM. Finally, we concatenate those features and language embeddings before adding positional and segment type embeddings.

B.3 Hyper-parameters

We used the ADAM optimizer without scheduling of learning rate (LR), and enabled Automatic Mixed Precision (AMP). Every training was 10 epoch. In a preliminary experiment, we observed that optimization of pre-trained ResNet18 had little impact on the VQA performance, so we only used the frozen setting above.

BERT$_{\text{small}}$ / +V. We fixed the max token length for the GLUE tasks and VQA 300, and for SQuAd 512. We tried six hyper-parameter combination: the mini-batch size from {64, 128, 256}, the LR from {1e-4, 5e-5}. We adopted the hyper-parameter set whose smallest validation loss was the smallest.

[13]SQuAD : https://rajpurkar.github.io/SQuAD-explorer/ and VQA : https://github.com/GT-Vision-Lab/VQA.

process	time	remarks
record gold seqs for PTA	6h	62k examples.
record gold seqs for the others	∼4d	∼1.3M examples.
Train small / medium on PTA in 50 ep.	∼2d / ∼4d	60k examples. with a GPU.
Train small / medium on the multi-task training in 10 ep.	∼6d / ∼12d	∼1.0M examples. with a GPU.
Predict with a model on val. split of PTA	20min	2k examples. with a GPU.
Predict with a model on val. split of the others	∼2d	∼230k examples. with a GPU.

Table 7: Required time. Since we used several servers with the different configurations, those values are approximations. Gold sequences are reusable if the screensize, tokenization and actions of the models are identical. We saved the screenshots and actions of a single example in a single json file, and read it from disks each time we used it. We used float32 for the training. We also tried float16 with automated mix precision. Although it reduced the training time by about 30% (we doubled the batchsize using the reduced memory space), it sometimes caused NaNs and stop the training. Therefore, we did not use it this time.

T5-small+V and BERT$_{\text{small}}$-s2s+V. We fixed the base mini-batch size 32, the max token length for text-only tasks 512 and for text-and-image tasks 432 (+70 image embeddings). We tried six hyper-parameter combination: gradient accumulation from {1, 4} and LR from {1e-4, 5e-5, 1e-5}. We adopted the hyper-parameter set whose smallest validation loss was the smallest. In the T5-small+V training with the LR 1e-4 and the AMP enabled, we sometimes saw NaNs in training losses after several epochs. We ignored such losses and continued the training. (Unlike the case of BUI-BERT, NaNs did not occur in the parameters of the model.) The best hyper-parameters were (1, 1e-4) for T5-small+V, and (4, 5e-5) for BERT$_{\text{small}}$-s2s+V

B.4 Classification with Seq2Seq models

For classification tasks, we considered the model failed to submit an answer when the generated text did not exactly match any class labels specified in the instruction.

C Tasks in the BUI setup

C.1 Instructions for Answer Forms

Here, we shows the instructions and answer forms as images. Figure 5 shows the SQuAD and VQA pages. Figure 6 shows the task pages for the GLUE tasks. Figure 7 and Figure 8 show the task pages for the SA tasks. Instructions are basically the same as the counter parts for the seq2seq models shown in Table 8 except for that word choices are changed so that they fit to the screen.

C.2 Templates for Pre-Training for Actions

Vocabulary. We made a vocabulary from the training split of the Wikitext103 (Merity et al., 2016) corpus. We kept the words that consist of only alphabets and numbers. We lower-cased the words.

Sets of words in the instructions are expanded to make the variation. We sampled one uniformly from the instructions for a task instance.

C.2.1 Cursor

Instructions:

- Move the cursor in the box.
- Point to the box with the cursor.

The coordinates of the box was sampled from a window uniformly.

C.2.2 Button

Instructions:

- {Click, Push, Press, Choose, Select} the button labelled WORD.
- {Click, Push, Press, Choose, Select} the WORD button.

WORD was sampled from the vocabulary.

C.2.3 Text

Instructions:

- {Type, Enter, Input} the string to the left of it in each text box. Click the submit button at last.

Each string was made by jointing two words, sampled from the vocabulary, with a space.

C.2.4 Area

Instructions:

- Scroll down until the buttons appear and click the button labelled WORD.
- Scroll down until the buttons appear and click the WORD button.

WORD was sampled from the vocabulary.

#contents	template
1	[TASK_TYPE] : [INSTRUCTION] [VALUE_1]
2	[TASK_TYPE] : [INSTRUCTION] [KEY_1] = [VALUE_1] [KEY_2] = [VALUE_2]

	TASK_TYPE	INSTRUCTION	KEY_1	VALUE_1	KEY_2	VALUE_2
VQA	visual question answering	See the picture and answer the following question.	question	(question)		
SQuAD	question and answering	Read the next paragraph and answer the following question. answer an empty string when you think the question is unanswerable.	Paragraph	(paragraph)	Question	(question)
CoLA	single choice classification	If the following sentence is acceptable as an English sentence, answer acceptable; if not, answer unacceptable.	sentence	(sentence)		
SST-2	single choice classification	Predict the emotion of the sentence (positive / negative).	sentence	(sentence)		
STS-B	single choice classification	Rate how similar the following two sentences are on a scale from 0 to 5 (0 being the least similar and being the most similar 5).	sentence1	(sentence1)	sentence2	(sentence2)
MRPC	single choice classification	Answer whether the following pairs of sentences are semantically equivalent. If they are equivalent, answer equivalent; if not, answer not equivalent.	sentence1	(#1 String)	sentence2	(#2 String)
MNLI	single choice classification	For the following premise and hypothesis statements, answer entailment if the premise entails the hypothesis, contradiction if it contradicts the hypothesis, or neutral if neither.	Premise	(sentence1)	Hypothesis	(sentence2)
WNLI	single choice classification	Read the following two sentences and answer their relationship: enntailment or not entailment.	sentence1	(sentence1)	sentence2	(sentence2)

Table 8: (top) the templates for text input, and (bottom) the rules to fill the sentences to the templates for seq2seq models. We used data from the datasets for the value fields.

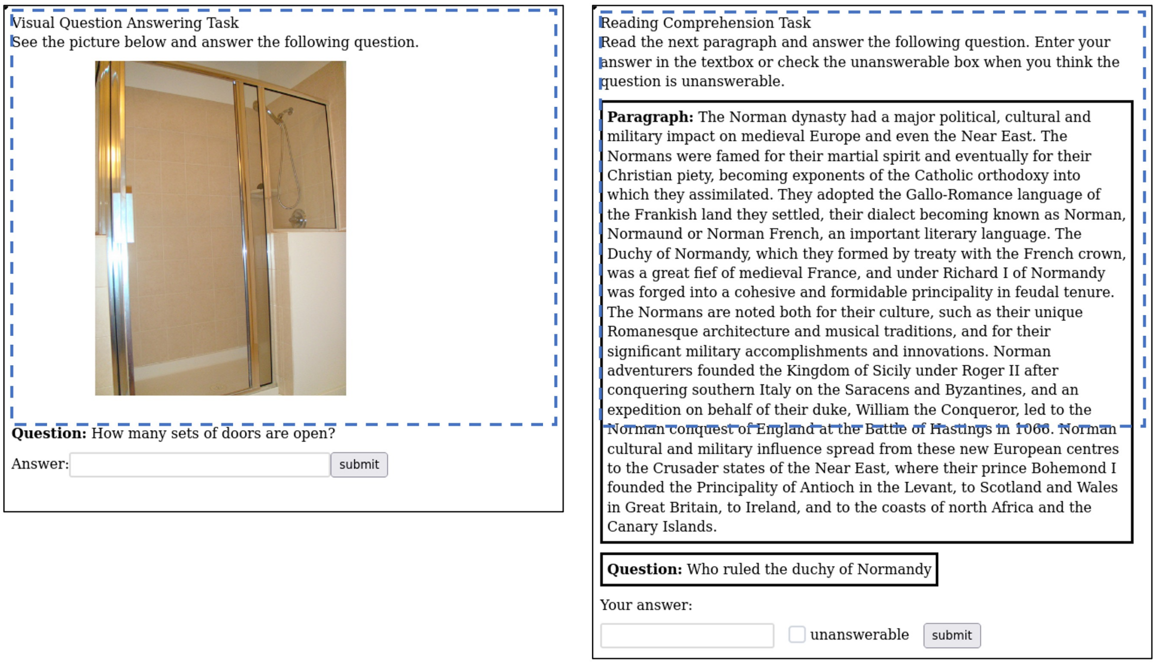


Figure 5: Screen examples form the BUI version of VQAv2 (left) and that of SQuADv2 (right). Those are screenshot that the BUI models receive. The blue dash rectangles show the initial visible area for the models. The instructions and answer forms are common for the all examples.

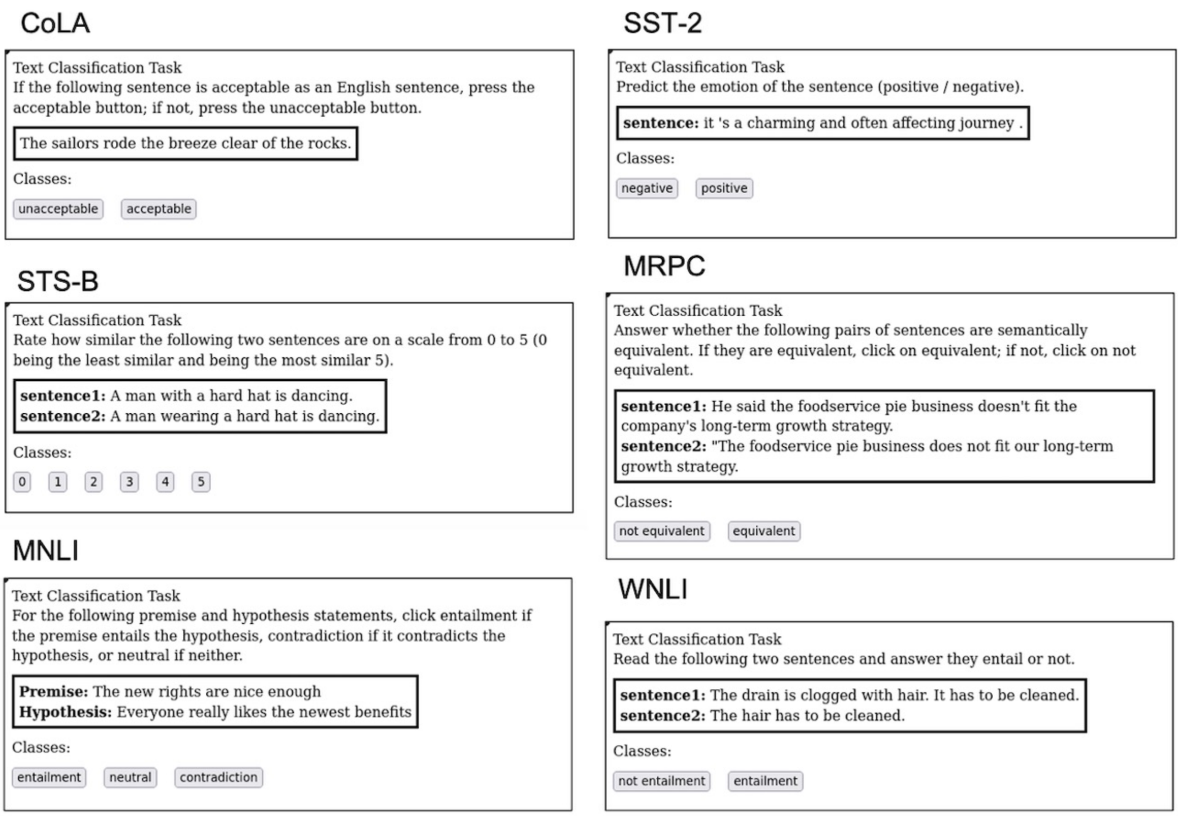


Figure 6: Screen examples from the BUI version of the GLUE benchmark. The bottom margins are omitted. While the contents in the bold solid boxes change depend on the examples, the instruction and the label buttons are common. Note that tasks we did not used (QNLI, QQP, and RTE) are not presented.

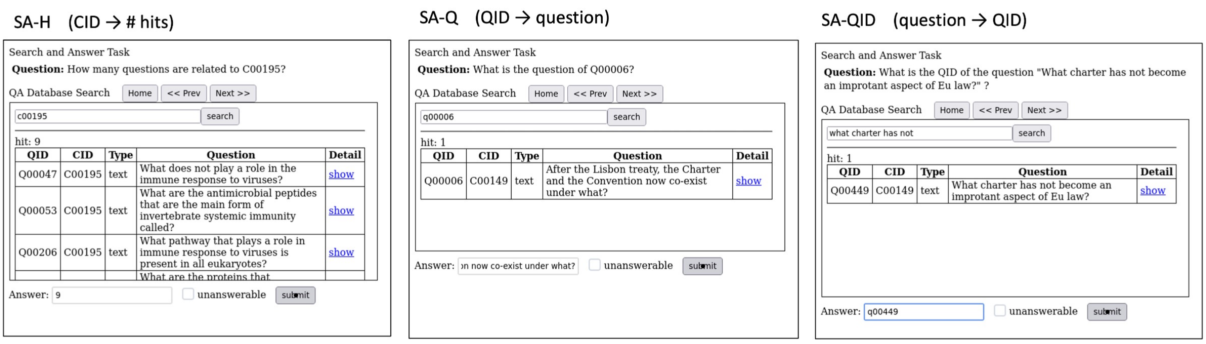


Figure 7: Screen examples of SA-QID, -Q and -H. The screenshots show the last step of tasks. These tasks are expected to be solved by (1) extracting a key phrase from a given instruction, (2) querying the key phrase, (3) finding an answer segment, and (4) entering the segment.

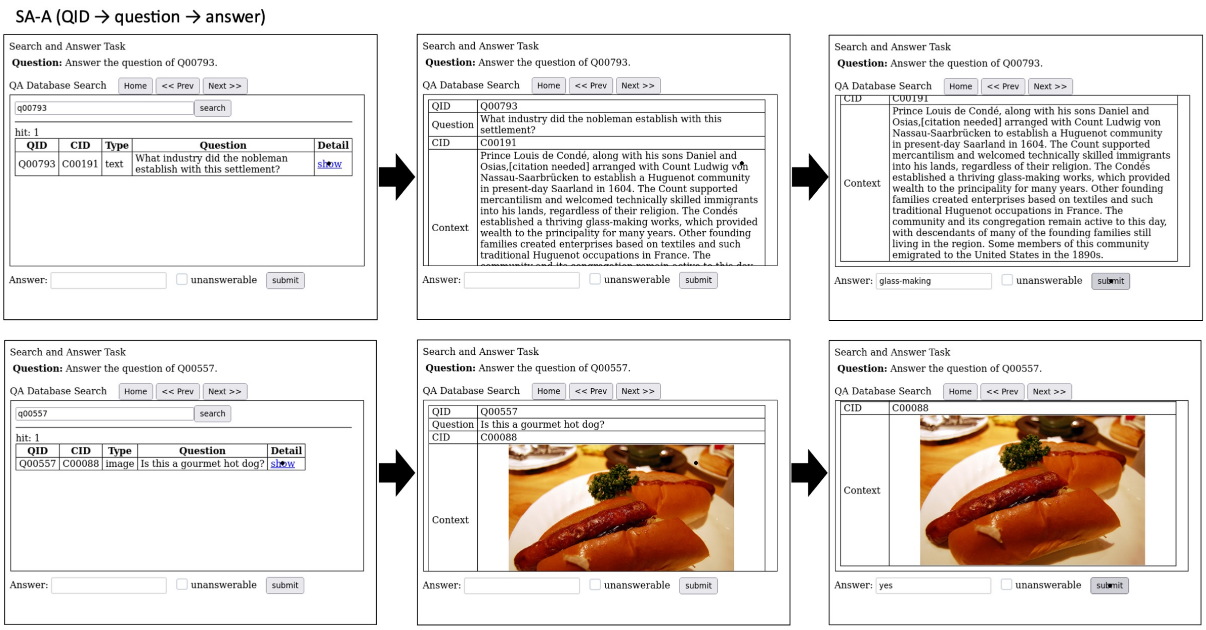


Figure 8: Screen examples of SA-A. These tasks are expected to be solved by (1) extracting a key phrase from a given instruction, (2) querying the key phrase, (3) showing the detail, (4) reading the question, (5) finding the answer in the context, and (6) entering the answer.

C.3 Detail of Search and Answer Tasks

For Search and Answer Tasks, we sampled 100 contexts (paragraphs or images) from each of SQuAD and VQA to create a database. The database contains ∼2k questions because each context has approximately 10 questions. We chose this database size to make it difficult to enter the whole data into the model. We assigned unique labels to each context and question in the database, CID, and QID, and created four tasks. A database yields 200 SA-H tasks and ∼2k SA-QID, -Q, -A tasks. Finally, we sampled 500 tasks from those generated tasks.

In total, we created 100 databases (50000 tasks) for the training split, and 10 databases (5000 tasks) for the validation split. The contexts do not overlap between databases.

The search UI uses partial matching on the entries

C.4 Distribution of the Gold Sequence Length

Figure 9 shows the distributions of the length of gold action sequences. Almost all of the examples fall within the upper limit of 50 steps that we set during our training. Tasks that require entering answers into text boxes tend to have a longer number of steps.

D Additional Details on BUI-BERTs

D.1 OCR Emulation

We used OCR emulation, where we surrounds each word in HTML sources using span tags, instead of real OCR in this work. Figure 10 shows an example. The Emulation do not capture the text in text boxes owing to technical reason. Words are sorted in a top faster and left faster manner. Sorting preserves natural orders basically, but it sometimes breaks the order as shown in the figure.

D.2 Mini-Batching Strategy

Figure 11 illustrates mini-batching we used for training. We packed multiple trajectories in a line of mini-batches to increase the filling rate. We input memory and last actions recurrently for a trajectory and reset them at each head of trajectories.

D.3 Learning Curves of BUI-BERTs

Figure 12 shows the learning curves of the BUI models. In the PTA training, three models were roughly converged. In the multi-task training, all models except BUI-BERT$_{\text{small}}$ w/o PTA were roughly converged in 10 epoch. However, the loss

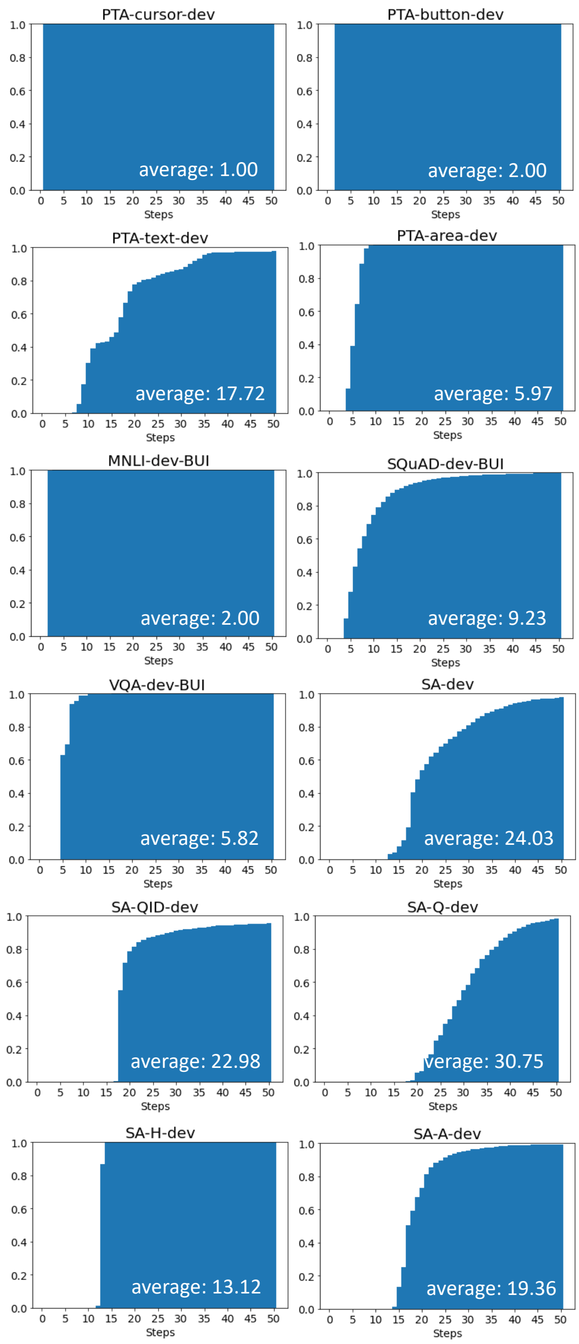


Figure 9: Distributions of the length of gold action sequences on the dev splits. We show cumulative values. Since the number of actions in the document classification task is basically two, we showed MNLI as a typical example.

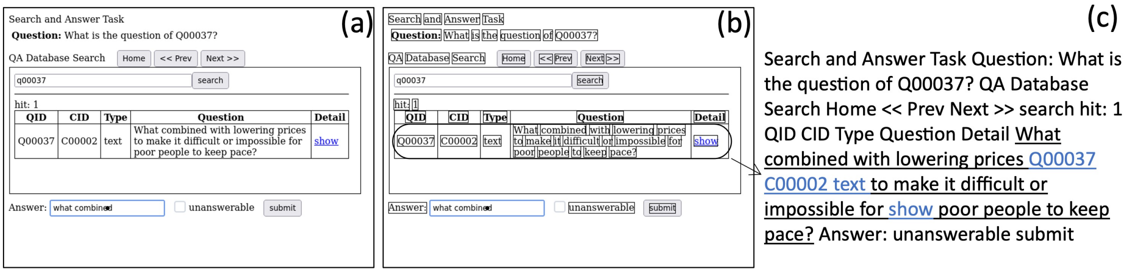


Figure 10: Example of our OCR emulation. (a) Example screen. (b) Detected words. detected words are surrounded by solid boxes. (c) Obtained text sequence. Parts with the broken order are underlined.

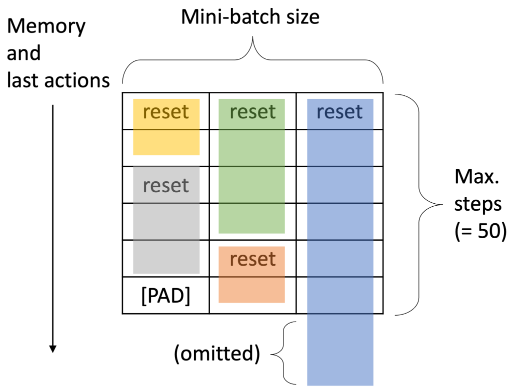


Figure 11: Mini-batching for multi-step training.

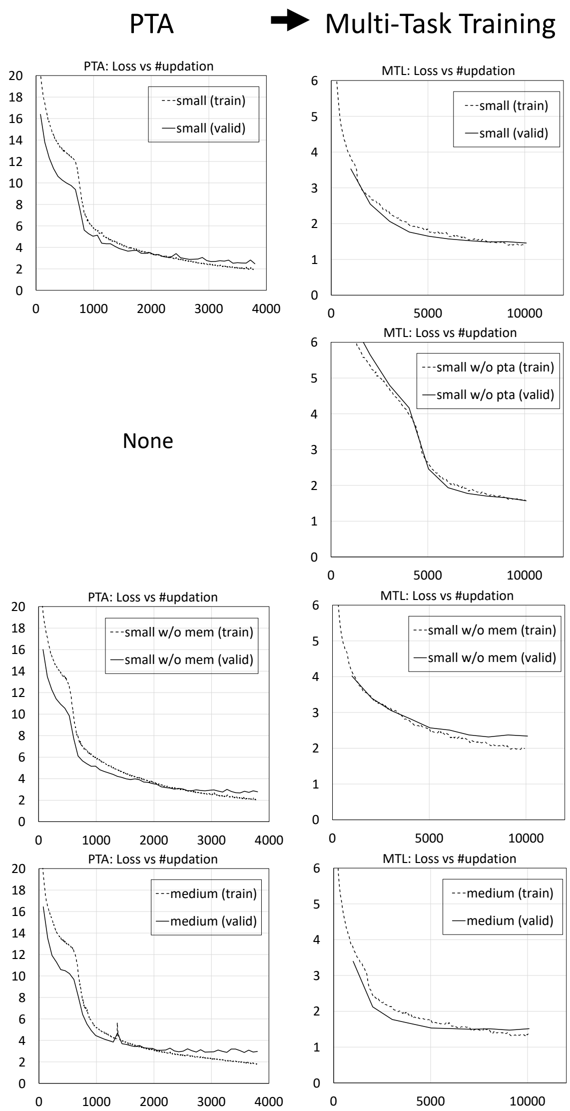


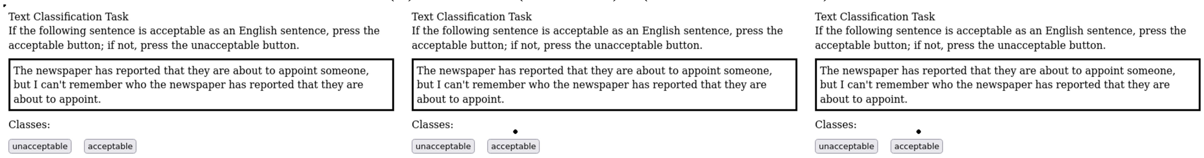
Figure 12: Learning curves of the BUI models.

of BUI-BERT$_{\text{small}}$ w/o PTA began to reduce drastically around 5k update and it could become smaller after 10 epoch. This indicate that PTA speeds up the convergence of the loss at least, but it may not affect the final performance achieved after longer time.

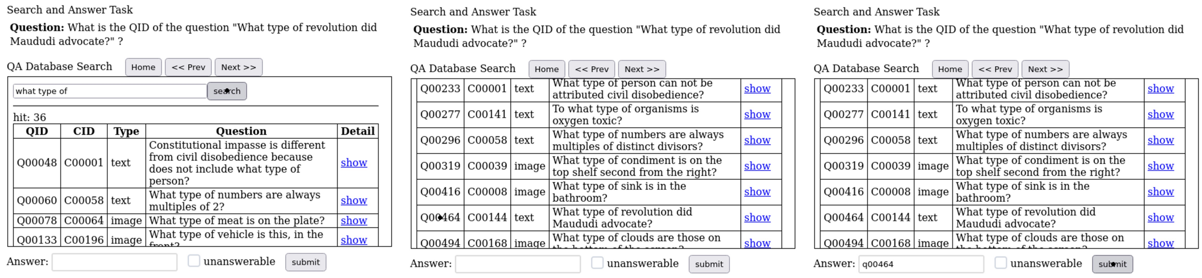
D.4 Cases of Task Execution

We show the cases of task execution using BUI-BERT$_{\text{small}}$ in Figure 13 as an aid to understanding.

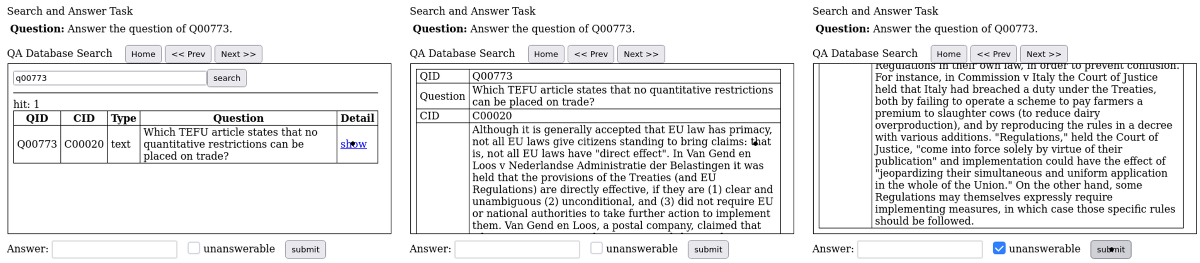
(a) Failure (timeout). (CoLA val. 554)



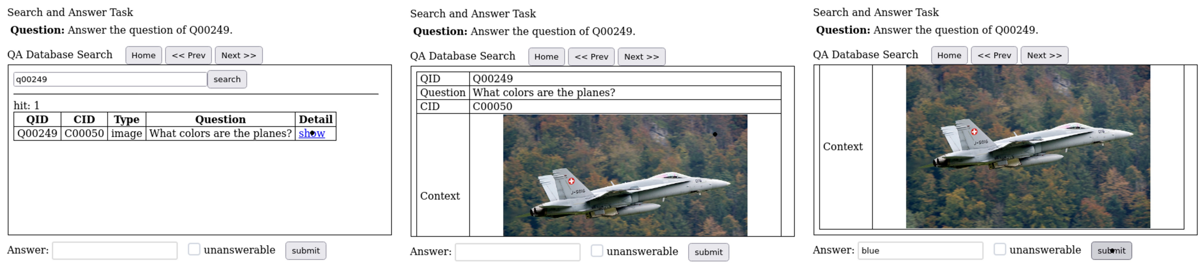
(b) Success. (SA val. 34)



(c) Failure. Gold answer : article 30, model : unanswerable (SA val. 42)



(d) Failure. Gold answer : gray, model : blue. (SA val. 46)



(e) Success. Gold answer : third, model : third-most abundant element. (SA val. 67)

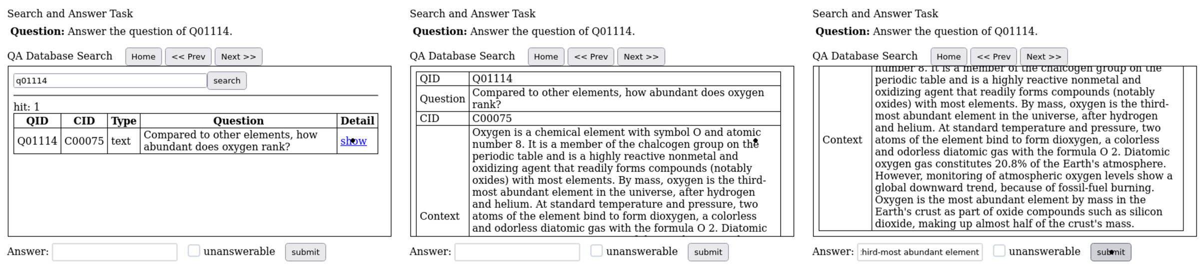


Figure 13: Case studies. (a) Model repeated move_to (172, 200), click, token ("unacceptable"), move_to (172, 178), click, token ("unacceptable"), move_to (172, 200), . . . (b) Model queried the first three words, which is the same strategy as the gold sequence, and obtained a list. It scrolled down until the question appeared and then extracted the QID successfully. (c) Model went to the detail and read all the context. However, it chose the unanswerable check box to an answerable question. (d) Model went to the detail to see the picture. The answer type was correct, but the answer was different to the gold answer. (e) Model went to the detail and read all the context to answer correctly.