From the holdings of the Department of Special Collections , Princeton University Library, Princeton, NJ

These documents are either in the public domain, or can only be used for educational and research purposes (Fair use) as per U.S. Copyright law (text below). By accessing this file, all users agree that their use falls in the public domain or within fair use as defined by the copyrightlaw. We do not charge any permission or usage fees for the publication of images of material inour collections, including those provided by Princeton University Library, via our website, catalog, or directly from staff.For more information regarding copyright, credit, and citation please review the information onthe Copyright, Credit and Citation Guidelines page of our website.
https://library.princeton.edu/special-collections/policies/copyright-credit-and-citation-guidelines. If you have any questions, including how to order higher-quality images from the collection, contact Special Collections Public Services via our Ask Us! Form. https://library.princeton.edu/special-collections/ask-us

---

Master's Theses, 2022-2026

Computer Science, 2023

2023

Learning Language through Interactions with the Digital World

Yang, John Boda

Princeton, NJ : Princeton University

---

http://arks.princeton.edu/ark:/88435/dsp01ns064933x

U.S. Copyright law: The copyright law of the United States (Title 17, §108, United States Code) governs the making of photocopies or other reproductions of copyrighted material. Under certain conditions specified in the law, libraries and archives are authorized to furnish a photocopy or other reproduction. One of these specified conditions is that the photocopy or reproduction is not to be used for any purpose other than private study, scholarship, or research. If a user makes a request for, or later uses, a photocopy or reproduction for purposes in excess of fair use that user may be liable for copyright infringement. This institution reserves the right to refuse to accept a copying order if, in its judgment, fulfillment of the order would involve a violation of copyright law. https://www.copyright.gov/title17/title17.pdf

Learning Language through Interactions with the Digital World

John Boda Yang

A Master's Thesis
Presented to the Faculty
of Princeton University
in Candidacy for the Degree
of Master of Science in Engineering

Recommended for Acceptance by
the Department of
Computer Science

Adviser: Professor Karthik Narasimhan
Reader: Professor Danqi Chen

May 2023

I hereby declare that I am the sole author of this thesis.

I authorize Princeton University to lend this thesis to other institutions or individuals for the purpose of scholarly research.

I further authorize Princeton University to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.



John Boda Yang

Abstract

A noteworthy omission in the development process of common NLP models is the lack of interactive components. While common downstream applications of large language models increasingly involve interacting with humans or other agents in a shared environment, there remains a gap in infrastructures and approaches for incorporating interactive machine learning components into the training or inference paradigms of existing large language models (henceforth referred to as LLMs). The emergence of reasoning and decision-making capabilities as interesting and desirable behaviors in such LLMs presents a number of opportunities for designing more benchmarks and methodologies in the realm of *interactive* natural language processing.

In this thesis, I discuss WebShop, a benchmark for interactive natural language processing. – a simulated e-commerce website environment that presents several challenges for language grounding including understanding compositional instructions, query (re-)formulation, comprehending and acting on noisy text in webpages, and performing strategic exploration. Given a text instruction specifying a product requirement, an agent needs to navigate multiple types of webpages and issue diverse actions to find, customize, and purchase an item. WebShop includes a collection of over $1,600$ human demonstrations for the task, and training plus evaluation of a diverse range of agents are performed using reinforcement learning, imitation learning, and pre-trained image and language models. The best model achieves a task success rate of 29%, which outperforms rule-based heuristics (9.6%) but is far lower than human expert performance (59%). Analysis of agent and human trajectories along with ablations of various model components provide insights for developing future agents with stronger language understanding and decision making abilities. Finally, agents trained on WebShop exhibit non-trivial `sim-to-real` transfer when evaluated on `amazon.com` and `ebay.com`, indicating the potential value of WebShop towards developing practical web-based agents that can operate in the wild.

Acknowledgements

I'd like to extend my gratitude first and foremost to the Princeton Computer Science department. Studying at this institution has helped me redefine and rediscover my curiosity for computer science. I'm thankful for how my time here has instilled confidence in myself to put aside comfort or security in favor of directions that inspire my excitement and enthusiasm.

I am truly grateful to Professor Karthik Narasimhan for his kindness, mentorship, and the invaluable opportunity he gave me to learn and grow as a researcher under his direction. Working with his group kindled the spark and established my drive to continue participating in and contributing to the NLP academic community. I'm incredibly fortunate to have had such a wonderful mentor.

A pivotal aspect of my time here has been my fortune to be part of the Princeton Natural Language Processing group. As I move forward, I'm especially thankful for the time I've been able to spend in the Friend Center with brilliant and supportive colleagues. Thank you to Dan Friedman, Zexuan Zhong, Alexander Wettig, Jane Pan, Jens Tuyls, Vishvak Murahari, Austin Wang, Gianluca Bencomo, Xindi Wu, and Yuhan Liu in addition to the many more who have formed my fondest memories of Princeton University. Thank you to my collaborators and friends Shunyu Yao and Howard Chen for bringing me on board for the WebShop project and always being willing to provide both valuable insights and advice for my research studies. I am deeply grateful to Carlos E. Jimenez for introducing me to the wonderful world of NLP and for being an unwavering rock of support, guidance, and encouragement throughout my journey these past two years.

Finally, thank you to my loving family, Jia Chen, Binwei Yang, and Joyce Yang for always being with me in my heart and soul every step of the way in life. I hope I will continue to make you proud as your son and brother, and I'm so grateful to have my life blessed with you three by my side. This thesis is dedicated to my family.

Contents

Abstract iii

Acknowledgements iv

List of Tables vii

List of Figures ix

1 Introduction **1**

2 Related Works **4**

3 Environment Formulation **6**

3.1 Task Formulation 6

3.2 Environment Implementation 9

3.3 Research Challenges 10

4 Methods **12**

4.1 Rule Baseline 12

4.2 Imitation Learning (IL) 12

4.3 Reinforcement Learning (RL) 14

5 Experiments **16**

5.1 Setup and task verification 16

5.2 Results 16

5.3 Analysis 18

5.4 Zero-shot Sim-to-real Transfer 21

6 Extensions **23**

6.1 Environment 25

6.1.1 Reward Function Reformulation 25

6.1.2 Semantic Details 27

6.2 Scalability 27

7 Appendix **30**

7.1 Environment Details 30

7.2 Model Details 36

7.3 WebShop Experiment Details 37

7.4 Sim-to-real Details 39

7.5 Potential Societal Impacts and Limitations 41

8 Conclusions **43**

List of Tables

3.1 Actions in WebShop. 7

3.2 Item rank in search results when the instruction is directly used as search query. 7

5.1 Left: Score breakdown. Right: average, maximum, and minimum number of states visited, items checks, and searches in a trajectory. . 18

5.2 Two example trajectories (showing only actions) from the human and the IL+RL model. We omit some human actions from instruction 2 for space and truncate the item names for readability. Red denotes options and blue denotes attributes. 19

5.3 Task performance with the `Choice` oracle. *first* and *last* refer to the first and last search queries found in human demonstrations, respectively. 20

5.4 Zero-shot sim-to-real transfer to Amazon and eBay over 100 test instructions. The Score / SR (Success Rate) column indicates the overall performance. The remaining breakdown are in Score. 21

6.1 Reward function verification comparing trajectories generated by average and expert human MTurk workers. 26

7.1 Product item statistics. 30

7.2 Two examples of failed human trajectories. A common pattern is impatience: after one search (even with correct attributes like the right example) the less performant worker commits to the first selected item. Often, the item does not contain the desired options even though the item's title text seem relevant. An expert worker will recognize the need to select the correct options and go back to refine the searches, while less performant workers simply commit to the current selected item. 34
7.3 Reward Verification Statistics 35
7.4 Task performance with the `Choice` oracle. *first* and *last* refer to the first and last search queries found in human demonstrations, respectively. 38
7.5 Zero-shot sim-to-real transfer to Amazon and eBay over 100 test instructions. 38
7.6 Sampling vs. top-1 38
7.7 Image ablations. 39
7.8 An example trajectory (showing only actions) from the IL agent on the real Amazon website. We omit instructions and some human actions for instruction and trim item names for readability. Red denotes options and blue denotes attributes. 40

List of Figures

1.1 The WebShop environment. **A**: An example task trajectory in `HTML` mode, where a user can (1) search a query in a `search` page, (2) click a product item in a `results` page, (3) choose a color option in a `item` page, (4) check `item-detail` pages and go back to the item page, and (5) finally buy the product to end the episode and receive a reward $r \in [0, 1]$ (§3.2). **B**: the `results` page in `simple` mode for agent training and evaluation. The blue text indicates clickable actions and bold text indicates an action selected by the agent. **C**: The product notation used in §3.1 with corresponding examples from the product in **A**. The attributes Y_{att} are hidden from the task performer. 2

4.1 Architecture of our choice-based imitation learning (IL) model. The image I is passed to a ResNet to obtain the image representation. The instruction text u is passed to a transformer (initialized with BERT) to obtain the text representations. The concatenated bi-modal representations are fused with the action representations using the Attention Fusion Layer. The resulting fused-action representations are mean-pooled and reduced by an MLP layer to a scalar value $S(o, a)$ denoting the logit value of the action `choose[`khaki`]`. 14

5.1 Task scores and Success Rate (%) for our models on the test split of WebShop over 3 trials. LP Search uses a pre-trained BART model to generate the search query and IL w/o LP Search uses the rule-based heuristic. LP Choice uses pre-trained BERT weights to initialize the choice action model and IL w/o LP Choice trains a Transformer from scratch. 17

6.1 Performance of fine tuned T5 models of various sizes on attribution generation, reformulated as a extractive short-phrase generation task. 28

7.1 The Amazon Mechanical Turk interface for the instruction writing task. The green box shows the general instruction for the task and the grey box shows an concrete example. 32

7.2 The Amazon Mechanical Turk interface for the instruction writing task. The blue box shows the actual annotation interface. The worker is required to check the boxes and write the instructions in the text field before submission. 33

Chapter 1

Introduction

Recent advances in natural language processing (NLP) and reinforcement learning (RL) have brought about several exciting developments in agents that can perform sequential decision making while making use of linguistic context [30, 51, 59]. On the other hand, large-scale language models like GPT-3 [6] and BERT [11] excel at traditional NLP benchmarks such as text classification, information extraction and question answering. While the former set of tasks are limited in their set of linguistic concepts and prove difficult to scale up, the latter tasks usually contain static, non-interactive datasets that lack adequate grounding to extra-linguistic concepts [4].

The world wide web (WWW) is a massive, open-domain, interactive environment that inherently satisfies the first aforementioned requirement through its interconnected set of pages consisting of naturally intertwined text, images and interactive elements. By being simultaneously **scalable**, **semantic**, **interactive**, **dynamic**, and **realistic**, the web is uniquely different from existing environments for autonomous agents like games or 3D navigation. While there has been prior work on building web-based tasks, they either lack depth in the transition and action spaces, or prove difficult to scale up. Some benchmarks only contain either a single classification task [39, 47, 31] or interactions containing only a handful of different pages in each

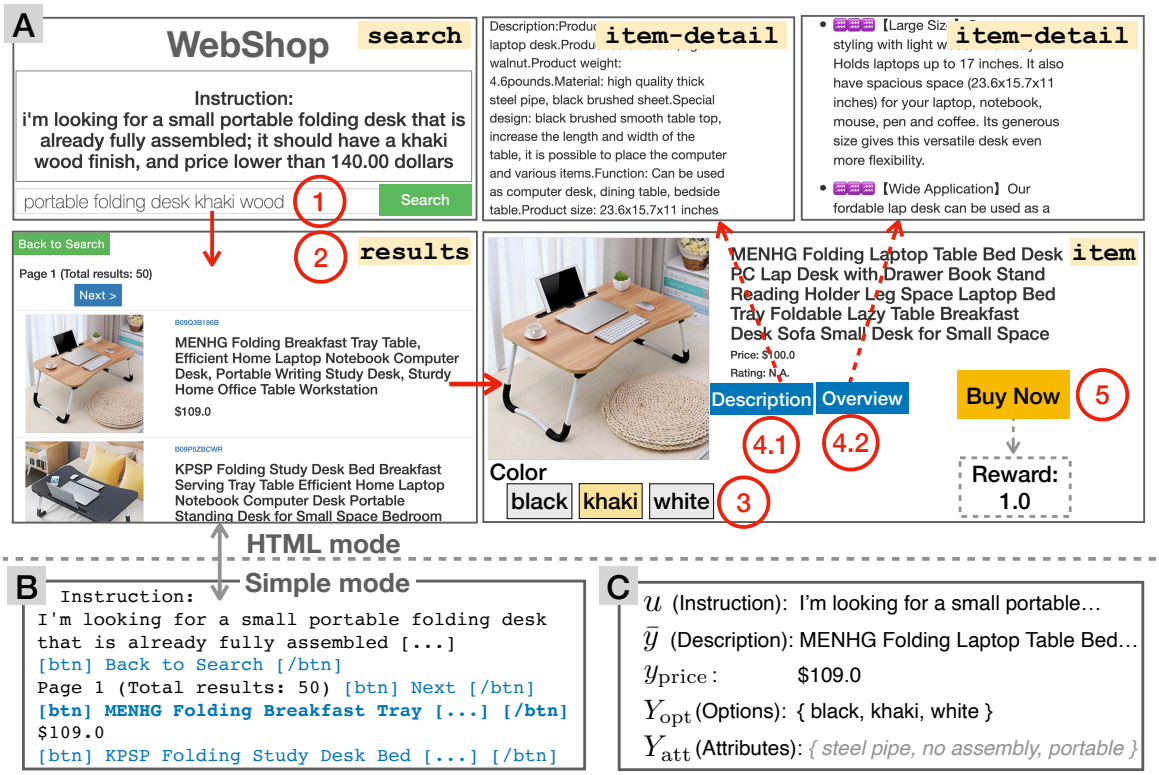


Figure 1.1: The WebShop environment. **A**: An example task trajectory in `HTML` mode, where a user can (1) search a query in a `search` page, (2) click a product item in a `results` page, (3) choose a color option in a `item` page, (4) check `item-detail` pages and go back to the item page, and (5) finally buy the product to end the episode and receive a reward $r \in [0, 1]$ (§3.2). **B**: the `results` page in `simple` mode for agent training and evaluation. The blue text indicates clickable actions and bold text indicates an action selected by the agent. **C**: The product notation used in §3.1 with corresponding examples from the product in **A**. The attributes Y_{att} are hidden from the task performer.

episode [44]. Others propose tasks with longer horizons but are either limited to following hyperlinks for web navigation [36] or require human-in-the-loop feedback due to the lack of an automated reward function [33].

In this paper, we introduce WebShop (Figure 1.1) – a large-scale interactive web-based environment for language understanding and decision making – and train autonomous agents to complete tasks on this benchmark. With the goals of being scalable and containing realistic language and visual elements, WebShop emulates the task of online shopping on an e-commerce website, where the agent's goal is to under-

stand a human-provided text instruction and *purchase* a product to match the specifications. WebShop contains over one million products scraped from *amazon.com*, over 12 thousand crowdsourced instructions, and a diverse semantic action space of searching text queries and choosing text buttons. It is packaged into a convenient OpenAI Gym [5] environment and can be rendered in two modes (`HTML` or `simple`) with parallel observation spaces that are easier for human and model gameplay respectively. Rewards are automatically computed using a combination of programmatic matching functions that consider the attributes, type, options and price of the chosen product, alleviating the need for human evaluation.

Chapter 2

Related Works

Reinforcement learning on the web

Nogueira et. al 2016 [36] introduced WikiNav as a benchmark for RL agents navigating pages, but the task is purely navigational with the actions restricted to either choosing a hyperlink to follow or deciding to stop. The World of Bits (WoB) benchmark [44] enables training of RL agents to complete tasks on webpages using pixel and Document Object Model (DOM) observations. Several follow-up papers have tackled MiniWoB using techniques like workflow-guided exploration [29], curriculum and meta-learning [15], DOM tree representation [21], adversarial environment generation [16] and large-scale behavioral cloning [20]. However, MiniWoB lacks long-range decision making across multiple different pages and does not scale easily in terms of difficulty or size due to its use of low-level mouse clicks and keystrokes as actions. In contrast, WebShop requires navigating longer paths with context-based action selection and backtracking, and it uses high-level *search* and *choose* actions that are more scalable and transferable to real settings. While not directly operating on web pages, AndroidEnv [49] and MoTIF [8] provide environments to train agents for interacting with apps and services on mobile platforms.

Non-interactive web-based tasks

Various supervised classification tasks on webpages have been proposed, including predicting web elements [39], generating API calls [47, 48, 55] and semantic parsing into concept-level navigation actions [31]. Perhaps most similar content-wise to our work is the Klarna product page dataset [19] which contains over $50,000$ product pages labeled with different element categories for supervised classification. All these works only consider supervised settings with a single decision, and may require the definition of web APIs or command templates for each domain. Our benchmark, WebShop, combines webpages with realistic text and image content with a rich and diverse interaction space for long-range sequential decision making.

Leveraging the web for traditional NLP tasks

Several papers have explored the use of the web for information extraction [34] and retrieval [1], question answering [58, 25], dialog [46], and training language models on webtext [2]. These approaches primarily use web search engines as a knowledge retriever for gathering additional evidence for the task at hand. Perhaps most similar to our work is WebGPT [33], which uses a web interface integrated with a search engine to train RL agents to navigate the web and answer questions. However, our environment has a more diverse action and observation space (including images) and does not require human-in-the-loop evaluation.

Chapter 3

Environment Formulation

We create WebShop as a large-scale web-based interactive environment with over 1.1 million real-world products scraped from amazon.com. In this environment, an agent needs to find and purchase a product according to specifications provided in a natural language instruction. WebShop is designed in a modular fashion which disentangles the website transitions from the task-specific aspects like instructions and reward, allowing for easy extension to new tasks and domains.

3.1 Task Formulation

WebShop can be formulated as a partially observable Markov decision process (POMDP) $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \mathcal{U}, \mathcal{O})$ with state space $\mathcal{S}$, action space $\mathcal{A}$, deterministic transition function $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$, reward function $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to [0, 1]$, instruction space $\mathcal{U}$, and a state observation space $\mathcal{O}$.

State and action

A state $s \in \mathcal{S}$ represents a web page, which falls into one of the four types – the *search* page that contains a search bar, the *results* page that lists a set of products returned by a search engine, the *item* page that describes a product, or the *item-detail* page that shows further information about the product (Figure 1.1A(1-4)

Type	Argument	State → Next State
`search`	[*Query*]	Search → Results
`choose`	Back to search	∗ → Search
`choose`	Prev/Next page	Results → Results
`choose`	[*Product title*]	Results → Item
`choose`	[*Option*]	Item → Item
`choose`	Desc/Overview	Item → Item-Detail
`choose`	Previous	Item-Detail → Item
`choose`	Buy	Item → Episode End

Table 3.1: Actions in WebShop.

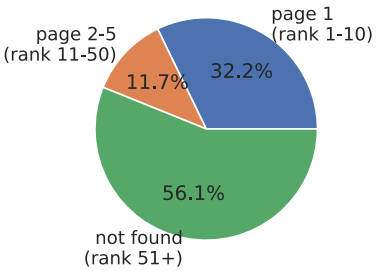


Table 3.2: Item rank in search results when the instruction is directly used as search query.

respectively). We define the following notations for a product y. We denote $\bar{y}$ to be the aggregation of the various text fields including product title, description, and overview. We denote y_{price} to be the price, Y_{opt} to be a set of buying options, and I to be a set of images, each corresponding to a specific option. Finally, each product is associated with Y_{att}, a set of attributes hidden from the agent which is extracted from the title and the *item-detail* pages. The attributes are used for the automatic reward calculation.

An action $a \in \mathcal{A}(s)$ can either be searching a text query (e.g. `search[Red shoes]`) or choosing a text button (e.g. `choose[Size 9]`). These two action types are not available simultaneously – search is only allowed when the agent is at the search page; on all other pages, click is the only action choice. The chosen action argument (button) will be clicked as a web link as opposed to the low-level mouse-click actions in previous environments such as World of Bits [44]. The transitions initiated by clicks deterministically redirect the web page to one of the four page types. The transition initiated by search is based on a deterministic search engine (§3.2).

Observation

Using Flask [42] and OpenAI Gym [5], we provide two parallel observation modes to render the state and instruction $\mathcal{S} \times \mathcal{I} \rightarrow \mathcal{O}$: (1) `HTML` mode that contains the HTML of the web page, allowing for interaction in a web browser(Figure 1.1A),

and (2) `simple` mode which strips away extraneous meta-data from raw HTML into a simpler format (Figure 1.1B). Note that while the environment allows for training reinforcement learning agents on raw pixels in `HTML` mode (like in [44]), we believe that it provides a very low-level non-semantic action space. Moreover, it is straightforward to write a translator that converts any new HTML page into `simple` format for use with trained agents, which enables sim-to-real transfer.

Instruction and reward

Each natural language instruction $u \in \mathcal{U}$ contains the following information: a non-empty set of attributes U_{att}, a set of options U_{opt}, and a price u_{price}. The instruction is generated based on a target product y^* by human annotators. The instruction collection process is lightweight and scalable (§3.2). Concretely, $U_{\text{att}} \subseteq Y^*_{\text{att}}$ is a subset of the product attributes, $U_{\text{opt}} \subseteq Y^*_{\text{opt}}$ is a subset of the product option field-value pairs, $u_{\text{price}} > y^*_{\text{price}}$ is a price set to be higher than the target product price. For example, the instruction "Can you find me a pair of *black-and-blue* sneaker that is *good in rain weather*? I want it to have *puffy soles*, and price less than 90 dollars." contains the aforementioned attributes $U_{\text{att}} = \{$"waterproof", "soft sole"$\}$ and option $U_{\text{opt}} = \{$"color": "black and blue"$\}$. In each episode, the agent receives a reward $r = \mathcal{R}(s_T, a)$ in the end at timestep T, where $a =$ `choose[buy]`, y is the product chosen by the agent in the final state s_T, and Y_{att} and Y_{opt} are its corresponding attributes and options. The reward is defined as:

$$r = r_{\text{type}} \cdot \frac{|U_{\text{att}} \cap Y_{\text{att}}| + |U_{\text{opt}} \cap Y_{\text{opt}}| + \mathbf{1}[y_{\text{price}} \leq u_{\text{price}}]}{|U_{\text{att}}| + |U_{\text{opt}}| + 1} \tag{3.1}$$

where the type reward $r_{\text{type}} = \texttt{TextMatch}(\bar{y}, \bar{y}^*)$ is based on text matching heuristics to assign low reward when y and y^* have similar attributes and options but are obviously different types of products. For example, "butter" and "plant-based meat" differ in types but may both contain attributes "cruelty-free", "non-GMO", and an

option "size: pack of 2".

Evaluation metrics

We use two evaluation metrics: (1) **Task Score**: defined as ($100 \times$ avg. reward), which captures the average reward obtained across episodes; and (2) **Success Rate (SR)** defined as the portion of instructions where $r = 1$. Note that it is possible to obtain $r = 1$ for an episode even if the final product is not y^*.

3.2 Environment Implementation

Data scraping

We use ScraperAPI [35] to scrape $1,181,436$ products from `amazon.com` across 5 categories (fashion, makeup, electronics, furniture, and food) using 113 sub-category names as queries. The product texts (title and item details) have an average length of 262.9 and a vocabulary size $224,041$ (word frequency higher than 10). In addition, the products have a total of $842,849$ unique options, reflecting the scale and complexity of the data.

Search engine

We use Pyserini [28] for the search engine, where indices are built offline using a BM25 sparse retriever with text for each product concatenated from the title, description, overview, and customization options. The search engine is deterministic, which eases imitation learning and result reproducibility.

Attribute mining and annotation

Each product is annotated with a set of hidden *attributes*, which are used to represent its latent characteristics as well as to calculate the reward as detailed in §3.1. An attribute is a short natural language phrase that describes the property of the product (see examples in Figure 1.1). We mine the attributes by calculating TF-IDF scores for all bi-grams in the concatenated titles and descriptions based on each

product category. We review the top 200 bi-grams for each category, remove the noisy ones by inspection (decide based on whether the bi-gram is human understandable), and assign them to the products. We consolidate a pool of 670 attributes.

Natural language instructions

We use Amazon Mechanical Turk (AMT) to collect natural language instructions that specify goal products with appropriate options. Specifically, an AMT worker is presented with a sampled goal product, including the product title, category, attributes, and the buying options, and asked to write a command to instruct an automatic shopping agent to find the target. Workers are instructed to avoid being too specific such as including the entire title in the instruction, but stay faithful to describing the target product. We collect a total of $12,087$ linguistically diverse instructions with an overall vocabulary size of $9,036$ words and an average length of 15.9 words.

Human demonstrations

We collect trajectories from humans performing the task in the `HTML` mode of WebShop to understand the task difficulty for humans and to analyze how humans would solve the task. We use qualification tests to train and select motivated workers to perform the task. We recruit and train a total of 13 workers for data collection, and among them we select the top 7 performing workers to be "experts" (see §7.1 for examples). We also leverage this data to perform imitation learning (described in §4.2).

3.3 Research Challenges

WebShop brings together several research challenges for autonomous systems from various subfields in NLP and RL into a single benchmark. These include: 1) generation of good search queries [22, 60] and reformulation [37, 52], 2) strategic exploration

for navigating through the website [56, 57, 29], 3) robust language understanding for textual state and action spaces [3, 7, 17, 45], and 4) long-term memory for comparing items or backtracking [54, 13, 23] (Figure 1.1). While we believe individual advances in each of these will improve agent performance, WebShop also provides an ideal testbed for the development of interdisciplinary techniques that tackle more than one of the above mentioned challenges simultaneously. For example, external memory modules may be very effective if combined with strategic exploration, or exploration could be helpful in information query reformulation. Further analysis based on human and model trajectories is in §5.3.

Chapter 4

Methods

We propose various models that combine language and image pre-training with imitation learning (IL) and reinforcement learning (RL).

4.1 Rule Baseline

A simple **rule baseline** is to search the exact instruction text, then choose and buy the first item in the results page without choosing any options. The heavy lifting of the lexical search engine makes it also a simple non-learnable information retrieval (IR) baseline, and would lead to a non-trivial attribute reward. However, simple heuristic rules cannot resolve noisy natural language options, strategically explore, or learn to generate what to search, so the total reward and task success rate should be low.

4.2 Imitation Learning (IL)

For the text generation and choice problems presented in WebShop, we propose using two pre-trained language models to separately learn how to search and choose from human demonstrations.

Imitating human search generation

We frame searching as a sequence-to-sequence text-generation problem: the agent generates a search action $a = \texttt{search[...]}$ given an instruction u without considering any other context (e.g. past searches, visited items). We use $M = 1,421$ instruction-search pairs from $1,012$ training human trajectories to construct a dataset $\mathcal{D} = \{(u, a)\}_{i=1}^{M}$ and fine-tune a BART model [26] parameterized by ϕ to perform conditional language modeling:

$$\mathcal{L}_{\text{search}} = \mathbb{E}_{u,a\sim\mathcal{D}}\left[-\log \pi_\phi(a \mid u)\right] \tag{4.1}$$

Imitating human choice. The choice-based imitation model (Figure 4.1) predicts a probability distribution over all the available click actions $\mathcal{A}(o)$ in observation o and maximizes the likelihood of the human clicked button $a^* \in \mathcal{A}(o)$. We construct a dataset $\mathcal{D}' = \{(o, \mathcal{A}(o), a^*)\}_{i=1}^{M'}$ of $M' = 9,558$ samples from the training human trajectories. We use a 12-layer pre-trained BERT model [10] parameterized by θ to encode the o into an observation representation of contextualized token embeddings, and we similarly encode each action. Each action representation is passed into a cross-attention layer with the observation representation, then mean pooled into a single vector and multiplied with a matrix W to obtain a scalar score $S(o, a)$. The policy $\pi_\theta\left(a \mid o, \mathcal{A}(o)\right)$ is the softmax distribution over action scores $S(o, a)$:

$$\mathcal{L}_{\text{choose}} = \mathbb{E}_{o,\mathcal{A}(o),a^*\sim\mathcal{D}'}\left[-\log \pi_\theta\left(a^* \mid o, \mathcal{A}(o)\right)\right] \tag{4.2}$$

$$\pi_\theta\left(a \mid o, \mathcal{A}(o)\right) \sim \exp\left(W^\top \text{mean}\big[\text{cross-attn}\big(\text{BERT}(o;\theta), \text{BERT}(a;\theta)\big)\big]\right) \tag{4.3}$$

Handling Images

To fuse in visual component of the observation in item pages, we use a pre-trained ResNet-50 [18] to pre-process images across different products and options into a 512 dimensional feature vector, which is then transformed into 768 dimensions with a

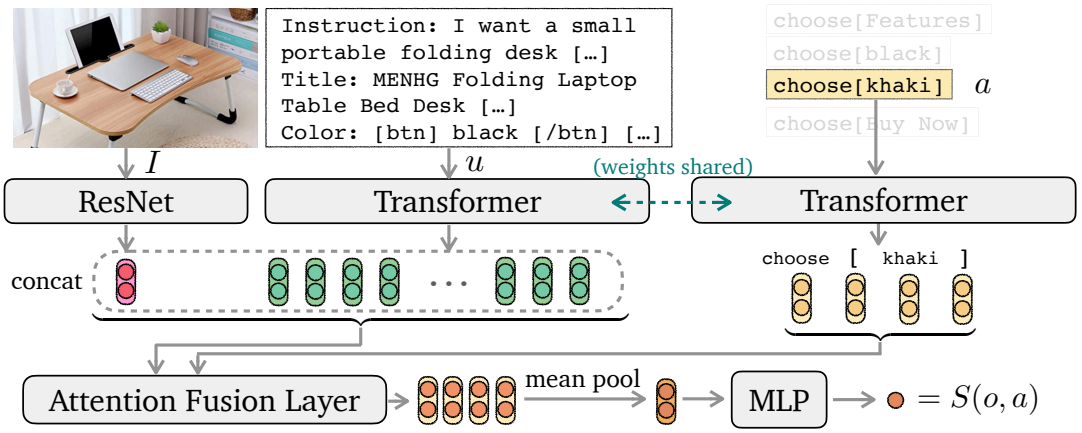


Figure 4.1: Architecture of our choice-based imitation learning (IL) model. The image I is passed to a ResNet to obtain the image representation. The instruction text u is passed to a transformer (initialized with BERT) to obtain the text representations. The concatenated bi-modal representations are fused with the action representations using the Attention Fusion Layer. The resulting fused-action representations are mean-pooled and reduced by an MLP layer to a scalar value $S(o, a)$ denoting the logit value of the action `choose[khaki]`.

learned linear layer and concatenated to BERT(o) as the observation representation.

Full pipeline

Combining the above during environment interaction, we use the BART model in the search page to generate the top-5 search queries via beam search and choose a random one. For other pages, we sample one action from $\pi_\theta\left(a \mid o, \mathcal{A}(o)\right)$ using the BERT model. We find these methods useful to encourage diverse actions. In contrast, an ineffective strategy that uses only the top generated search query or the button with the highest probability might lead to limited product candidates or being stuck (e.g. bouncing back and forth between pages).

4.3 Reinforcement Learning (RL)

We also fine-tune the choice-based IL model with online RL (i.e. IL+RL). Prior work suggests that directly fine-tuning text generation via RL might lead to language drifting [24] and deteriorated performance. Therefore, we freeze the BART model

to provide the top-10 search generations as a refined action space for the choice-based IL model to learn to pick – an inspiration borrowed from previous work in text games [56] and referential games [24]. We use the policy gradient method [32] with return-to-go $R_t = \mathbb{E}_\pi[r_t + \gamma R_{t+1}]$ and a learned value baseline $V(o) = W_v^\top \text{BERT}(o; \theta)$ parameterized by $\{W_v, \theta\}$ (the BERT weights are tied with the policy):

$$\mathcal{L}_{\text{PG}} = \mathbb{E}_\pi \left[- \left(R_t - V(o_t) \right) \log \pi \left(a_t \mid o_t, \mathcal{A}(o_t) \right) \right] \tag{4.4}$$

The value $V(o)$ is learned with an L2 loss $\mathcal{L}_{\text{value}} = (R_t - V(o_t))^2$. We also add an entropy loss $\mathcal{L}_{\text{entropy}} = \sum_{a \in \mathcal{A}(o_t)} \pi_\theta\big(a_t \mid o_t, \mathcal{A}(o_t)\big) \log \pi_\theta\big(a_t \mid o_t, \mathcal{A}(o_t)\big)$ to prevent premature convergence. Our full RL model minimizes the total loss $\mathcal{L}_{\text{RL}} = \mathcal{L}_{\text{PG}} + \mathcal{L}_{\text{value}} + \mathcal{L}_{\text{entropy}}$.

Chapter 5

Experiments

5.1 Setup and task verification

We split a total of 12,087 instructions into an i.i.d. distributed train / development / test split of 10,587 / 1,000 / 500 instances for all models. While future work can investigate splits with more generalization gaps (e.g. split by product category), we will show the i.i.d. split is already challenging for current models. We randomly sample a subset of the 10,587 training instructions, then collect 1,012 human demonstrations for task verification and imitation learning (IL) and a further 54 demonstrations from instances in the development set for IL hyperparameter tuning and checkpoint selection. We also collect human trajectories for all 500 test instructions and report human and model performances averaged across these 500 instructions.

5.2 Results

Task performance

From Figure 5.1, we observe that the rule baseline obtains a low score of 45.6 and a very low success rate of 10% since it cannot resolve options specified in language or explore more products, empirically demonstrating the non-trivial nature of the task.

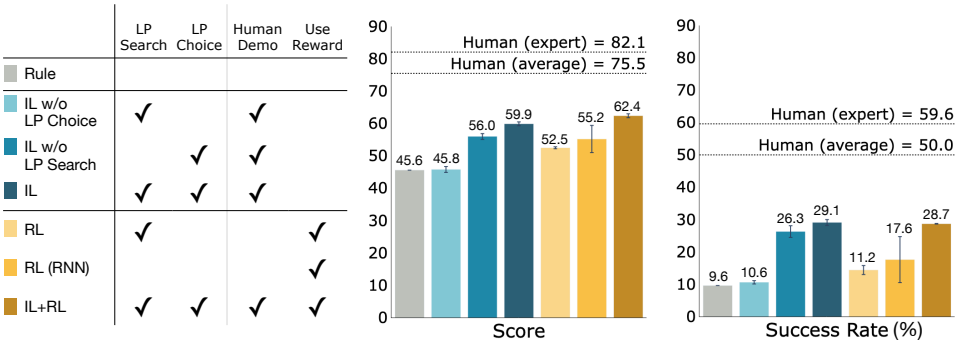


Figure 5.1: Task scores and Success Rate (%) for our models on the test split of WebShop over 3 trials. LP Search uses a pre-trained BART model to generate the search query and IL w/o LP Search uses the rule-based heuristic. LP Choice uses pre-trained BERT weights to initialize the choice action model and IL w/o LP Choice trains a Transformer from scratch.

The IL model significantly outperforms the rule baseline on both metrics, achieving a score of 59.9. Further RL finetuning improves the score to 62.4 while slightly hurting the success rate (29.1% $\rightarrow$ 28.7%) (analyzed further in §5.3). We also observe a significant gap between models and humans – our best model's success rate (29.1%) is less than half of expert humans (59.6%) and only 60% of the average human (50%). This indicates a great room for model improvement by tackling reseach challenges in WebShop.

IL ablations

Figure 5.1 also contains several ablations that confirm important design choices for models. When the choice action model for the IL agent is randomly initialized (**IL (w/o LP Choice)**; LP = language-pretraining), the success rate drops by nearly two-thirds, indicating the importance of language pre-training for our task. When the search query generator in the IL agent is replaced by a simple rule, which always uses the instruction text (**IL (w/o LP Search)**), both reward and success rate drop by around 3 points. This suggests the importance to explore by expanding the search space for exploration, but it is not as critical as learning to choose the right options. We experiment with incorporating history of one past observation and the last five actions into the model and find a slight degradation in the score from 59.9 to 57.3,

	Score					Count		
	All	Att	Opt	Type	Price	State	Item	Search
Rule	45.6	66.6	0.0	80.5	86.0	3.0 (3 / 3)	1.0 (1 / 1)	1.0 (1 / 1)
IL	59.9	69.3	**45.2**	86.4	84.0	9.4 (90 / 3)	1.6 (11 / 1)	1.3 (17 / 1)
IL+RL	**62.4**	**74.0**	38.9	**89.7**	**88.7**	4.5 (5 / 1)	1.0 (1 / 1)	1.0 (1 / 1)
Human Expert	82.1	81.8	73.9	94.4	97.7	11.3 (114 / 4)	1.9 (16 / 1)	1.4 (16 / 1)

Table 5.1: Left: Score breakdown. Right: average, maximum, and minimum number of states visited, items checks, and searches in a trajectory.

suggesting more advanced techniques are needed to leverage past information.

5.3 Analysis

To better understand the differences between the agents and human experts, we perform several fine-grained analyses. We first break down the overall score into its four sub-parts according to Eq. (6.1): 1) attribute score ($|U_{\text{att}} \cap Y_{\text{att}}|/|U_{\text{att}}|$), 2) option score ($|U_{\text{opt}} \cap Y_{\text{opt}}|/|U_{\text{opt}}|$), 3) price score ($\mathbf{1}[y_{\text{price}} \leq u_{\text{price}}]$), and 4) type score ($r_{\text{type}}$). We report trajectory statistics such as the average number of states, unique items visited, and number of searches per episode in Table 5.1 and provide qualitative examples of the trajectories in Table 5.2.

Human expert vs. agents

Human experts outperform the agents on all score sub-parts (Table 5.1), but the most significant boost comes from the option score (a 28% gap), revealing that agents have trouble selecting the correct product options. Humans also have longer trajectories, explore more items and perform more searches than the agents, with a higher variance, demonstrating their flexibility. Table 5.2 provides some samples trajectories. In the first example, the human decides to search again after removing 'inches', 'width', 'height', and 'white' from the query since product texts often contain abbreviated symbols for these terms like '"', 'w', and 'h'. Thus, **search generation** is challenging for models since it involves reasoning and adapting to grounded environ-

Instruction 1	**Instruction 2**
I want to find white blackout shades that are 66 inches in width and 66 inches in height. They need to be easy to install [...]	I need a gingko light and 20"x20" pillow cover that is hand painted [...]

Human Actions ($r = 1.0$, length $= 8$)
`search[` 66 inches in width and 66 inches in height white shades `]`
`choose[` item : CALYX... `]`
`choose[` Back to Search `]`
`search[` 66 x 66 blackout shades `]`
`choose[` item : Milin... `]`
`choose[` opt : 66"w x 66"h `]`
`choose[` opt : cordless bottom up-blackout-white `]`
`choose[` Buy `]`
	Human Actions ($r = 1.0$, length $= 17$)
	`search[` gingko light 20"x20" pillow cover hand painted `]`
	`choose[` item : Maison... `]` [...]
	`choose[` Description `]`
	`choose[` < Previous `]`
	`choose[` Overview `]`
	`choose[` < Previous `]` [...]
	`choose[` item : Maison... `]`
	`choose[` opt : 20"x20" `]`
	`choose[` opt : nudes (gingko light) `]`
	`choose[` Buy `]`

IL+RL Actions ($r = 0.2$, length $= 3$)
`search[` white blackout shades 65 inches in width and 66 inches in height `]`
`choose[` item : Window... `]`
`choose[` Buy `]`
	IL+RL Actions ($r = 0.25$, length $= 3$)
	`search[` gingko light and 20x20 pillow cover hand painted `]`
	`choose[` item : UPOOS... `]`
	`choose[` Buy `]`

Table 5.2: Two example trajectories (showing only actions) from the human and the IL+RL model. We omit some human actions from instruction 2 for space and truncate the item names for readability. Red denotes options and blue denotes attributes.

ments, and ideas from query reformulation [37, 1] could help alleviate this. Agents also struggle to perform robust **semantic matching**, which is important in choosing options that contain noisy paraphrases of instruction spans. In the second example, the human explores several products first, and decides to return to the first explored product, demonstrating long-term **memory** that is lacking in the IL+RL model.

Effect of RL fine-tuning after IL

Table 5.1 also shows that RL fine-tuning adapts the IL model to become more 'greedy' and less 'exploratory', as the average trajectory length drops from 9.4 to 4.8, and the model explores fewer items and search queries. As a result, the attribute, type, and price scores all increase, but option score drops from 45.2 to 38.9. This points to the need for a better balance exploration with exploitation during RL, e.g. by using intrinsic bonuses.

	Instr. text	IL BART	Human expert (first)	Human expert (last)
Score	94.9	94.5	94.5	**95.5**
Success Rate	85.4%	84.2%	85.6%	**87.8%**

Table 5.3: Task performance with the `Choice` oracle. *first* and *last* refer to the first and last search queries found in human demonstrations, respectively.

Results with at `Choice` oracle

To disentangle the effects of learning to search from choosing the right actions, we construct a `Choice` oracle that has access to the hidden reward function as well as hidden attributes and options underlying each product and instruction. A similar search oracle is also possible but harder to design since the search space is infinite. One possible oracle is to search for the underlying product name for each instruction, but that makes choice trivial as the underlying product is then almost always the first search result. Given a search query, the `Choice` oracle will perform an exhaustive search over every result item, try out all combinations of options and finally choose the best item with options that maximize the reward — meaning each episode will take hundreds or thousands of steps, as opposed to 4.5 and 11.3 steps on average for the IL+RL model and human experts (Table 5.1). We use 500 test instructions and consider four types of search queries: the instruction text (used by rule baseline), top IL BART generated query (used by all learning models), and the first and last queries from human experts in each test trajectory (74.8% of the time there is only one query in the trajectory). `Choice` oracle improves the success rate of rule heuristics from 9.6% to 85.4%, and even the human expert success rate from 59.6% to 87.8% (Table 7.4), confirming that choosing the right actions is indeed a major bottleneck for current models with great room for improvement. However, using a better search query is still important even with such a strong `Choice` oracle, as the last human search query still outperforms other search queries. This also suggests human experts improve search query qualities over reformulations.

5.4 Zero-shot Sim-to-real Transfer

Finally, we conduct a '*sim-to-real*' transfer experiment where our models trained on WebShop are tested on the real-world Amazon (`amazon.com`) and eBay (`ebay.com`) shopping websites without any fine-tuning. We sample 100 test instructions and deploy 3 WebShop models (rule, IL, IL+RL) to interact with Amazon and eBay, and manually score each episode based on Eq. (6.1). As shown in Table 5.4, model performances on the two website are similar to WebShop performances in Figure 5.1, except for the rule baseline, likely due to the better search engine of Amazon than WebShop.

	Amazon					eBay				
	Score / SR	Att	Opt	Type	Price	Score / SR	Att	Opt	Type	Price
Rule	45.8 / 19%	45.6	38.0	66.2	90.0	31.7 / 7%	62.3	25.9	49.0	67.0
IL	61.5 / 27%	60.7	**53.7**	85.6	96.0	58.2 / **21%**	60.2	**52.3**	85.1	96.9
IL+RL	**65.9** / **25%**	**71.6**	47.0	**87.8**	**100.0**	**62.3** / **21%**	**69.1**	39.5	**91.7**	**97.0**
Human	88.2 / 65%	86.2	76.3	99.0	100.0	79.7 / 40%	80.3	70.1	99.5	100.0

Table 5.4: Zero-shot sim-to-real transfer to Amazon and eBay over 100 test instructions. The Score / SR (Success Rate) column indicates the overall performance. The remaining breakdown are in Score.

On `amazon.com`, IL+RL achieves a Score of 65.9 and SR of 25%, outperforming the Rule baseline's Score of 45.8 and SR of 19% by large margin. Similarly, on `ebay.com`, IL+RL achieves a Score of 62.3 and SR of 21%, widely outperforming the Rule baseline's Score of 31.7 and SR of 7%. These results confirm positive sim-to-real values of trained agents for real-world web tasks despite domain shifts in data (products) and dynamics (search engine). We also obtain a human average score of 88.0 / 79.7 and success rate of 65% / 40% by asking turkers (§3.2) to find the instructed product on the Amazon and eBay websites respectively. While humans perform much better than agents, their web interactions are much slower — taking on average 815 seconds per episode as opposed to < 8 seconds per episode for our IL and IL+RL models on Amazon. This sim-to-real transfer only requires two minor coding

additions, suggesting that environments like WebShop are suitable for developing *practical* grounded agents to reduce human effort on real-world web tasks. We provide additional performance and in-depth analysis in §7.4.

Chapter 6

Extensions

A significant aspect of WebShop's utility towards model training is its ability to simulate real world web domains. This suggests that the WebShop environment should be realistic, scalable, and faithful to human perceptions towards this task. In this paper, we identify three key aspects where WebShop falls short on these claims, ultimately limiting its serviceability as a truly automatic environment. First, the WebShop environment does not include semantic information that heavily influences how humans perform the WebShop shopping task. Second, WebShop's original reward function consistently over-penalizes a chosen product due to its faulty exact matching criterion, compromising its faithfulness to human evaluation. Third, while WebShop's product dataset is collected in a scalable fashion via web scraping, generating corresponding instructions relies entirely on human crowd-sourcing; WebShop has 1.18 million real products, but of these, only $12,087$ have corresponding text instructions. This reliance on human generation does not scale and bottlenecks WebShop's model training efficacy.

We put forth improvements to address these three points, demonstrating how such adjustments collectively make for a semantically richer environment that better reflects real world platforms and offer a scalable way to generate more instructions

for model training. First, we solicit and incorporate feedback from an audience of 75 random individuals regarding information missing from WebShop that would be useful to completing the shopping task. Ensuring that WebShop captures key semantic components is fundamental to its main deliverable of constructing agents that can transfer to real-life settings. Second, we rewrite the automatic reward function's matching criteria to look for lexically similar and synonymous tokens when calculating the *attributes* and *options* score components. Our `V2` reward function coheres to human evaluation much more precisely (Original 81.5%, V2 87.7%, Human 89.9%). Lastly, we train and evaluate several attribute extraction models from a product's description. Our `t5-3b` model [41] fine-tuned on $2,000$ training points of `[X=product information, Y=attributes]` pairs achieves an accuracy of 72.22%, demonstrating the potential for high performance at an affordable cost in terms of human data collection. We then briefly discuss future plans to automate the instruction generation process. Eliminating the need for human participation in the instruction generation process is vital to WebShop's extendibility. As real world platforms evolve, WebShop's long term viability for model training hinges on how efficiently the environment, dataset, and instructions can be updated. Without such automation, WebShop's instructions and relevance will wither with time.

We believe that the collection of changes presented in this paper greatly advances WebShop's usability as an environment for designing language instructed agents with imminent real world applications, and our primary goal with this work is to make WebShop a worthwhile platform for developing web agents to the greater grounded language research community.

6.1 Environment

We address areas of improvement for WebShop's semantic richness and faithfulness to human evaluation. The collective goal of these changes is to close the gap on WebShop's claims and create a training environment that more accurately reflects real world counterparts.

6.1.1 Reward Function Reformulation

WebShop's original reward function generates a composite score from calculating the similarity strictly between two products' attributes, type, options, and price, with a custom programmatic matching function per category. Exact matching is used to score attributes and options. To quantify the faithfulness of the original reward function, we randomly re-score 100 samples, selected from a pool of trajectories generated by average and expert Amazon Mechanical Turk (AMT) workers, against a human criteria. This criteria follows the original reward function with two main modifications. Instead of exact matching, points are awarded if (1) the picked product's attributes, options or type are lexically similar or synonymous with the goal's product information and (2) the desired goal value is not found verbatim anywhere in the picked product's descriptions.

The matching criteria consistently overpenalizes a picked product due to its failure to account for lexical similarities and synonyms that humans would otherwise award. For instance, given a goal token *lightweight*, the existing reward function would award neither *light_weight* (semantically similar) nor *easy to carry* (synonym). In addition, the original approach does not reward a goal attribute or option that (1) does *not* appear in the picked product's corresponding category, but (2) does appear elsewhere in the product's description. For example, given *organic* as a desired option, a human scorer would award points if the picked product contains *organic* in its title even if *or-*

MTurk Type	Reward	Attribute	Options	Overall
Average	Original	71.7	50.5	72.4
	V2	74.1	55.0	74.9
	Human	75.3	57.0	76.3
Expert	Original	78.1	56.1	81.5
	V2	85.2	64.9	87.7
	Human	88.2	66.8	89.9

Table 6.1: Reward function verification comparing trajectories generated by average and expert human MTurk workers.

ganic is not presented as an option. The consistent disparity in the *attribute*, *options*, and *overall* scores between the *Original* and *Human* reward functions, as shown in Figure 6.1, highlights the over-penalization that manifests from these discrepancies.

We implement a modified reward function that applies lexical and synonym matching for scoring attributes and options along with a comprehensive search of product information. The new proposed reward function is defined in its entirety as Equation 6.1.

$$r = r_{\text{type}} \cdot \frac{match_{\text{attrs}}(U_{\text{att}}, Y_{\text{att}}) + match_{\text{opts}}(U_{\text{opt}}, Y_{\text{opt}}) + \mathbf{1}[y_{\text{price}} \leq u_{\text{price}}]}{|U_{\text{att}}| + |U_{\text{opt}}| + 1} \tag{6.1}$$

To determine the faithfulness of the new reward function to human rewarding, we repeat the aforementioned verification procedure with the new reward function defined in Equation 6.1 and list the average scores per category in Figure 6.1. We also re-run imitation learning models discussed in the original WebShop paper. For both average and expert MTurk worker trajectories, the *Attribute*, *Options*, and *Overall* scores generated by the V2 reward function are all greater than the `Original` reward function scores, but do not exceed the Human benchmarks.

Figure 6.1 reflects our observation that the V2 implementation of automatic scoring reduces over-penalization and is much more faithful to human evaluation. From

manual checks of 20 trajectories chosen randomly from the pool of 200 scored trajectories, the improvements in these scores can be directly attributed to the lexical and synonym matching cases. Across all 200 trajectories, there were no instances where the V2 reward function assigned a score that was greater than the corresponding Human reward function's score. The remaining gap between the V2 and Human reward functions can mainly be attributed to lexical versus numeric representations of numbers (i.e. "three" and "3") or a lack of contextualization when querying for synonyms (i.e. is "blue" used as a color or an emotion).

6.1.2 Semantic Details

We surveyed an audience of 75 individuals, each of whom were asked to (1) complete a single round of the WebShop shopping task, then (2) discuss if there was information useful for completing a shopping task that was not found in WebShop. The three most frequent responses were *customer ratings and reviews* (53 mentions), *similar products* (41 mentions), and *frequently asked questions* (37 mentions). We then implemented a *Reviews* tab on the WebShop environment that appears on a product's `item` page.

6.2 Scalability

WebShop's attribute tagging and instruction generation pipelines require human annotators. For the attribute tagging task, given a product and a pool of attributes, a human worker is tasked with assigning relevant attributes to the product. For the instruction generation task, given a product, including its title, product category, attributes, and options, a human worker is tasked with constructing a natural language query. This human-in-the-loop system is time-consuming, expensive, and also introduces potential human biases (i.e. varying degrees of knowledge across product categories). Furthermore, this methodology lacks robustness to changes in the

WebShop environment and product dataset. For instance, if new semantic signals are added to products (i.e. reviews), collecting new instructions that incorporate additional details carries a cost that must be paid every time for any future iteration. Yet, such adaptability would be crucial to WebShop's long term viability.

To automate the attribute generation task, we fine tune an out-of-box T5 model [41] to predict attributes from the product information. We train the model at different sizes on pairs of `[X=product information, Y=attributes]` drawn from WebShop's dataset of products annotated with attributes by MTurk workers. The product information consists of the title, description, and features. The corresponding label consists of a list of five attributes. We test T5 models of sizes `['small', 'base', 'large', '3b']` with training sets of size `[50, 200, 500, 1000, 2000, 3000, 4000, 5000, 6500, 8000]`. The validation and test data sets each contain $1,000$ data points. To evaluate the model's performance, we calculate accuracy as the intersection of the predictions and ground truth labels. Figure 6.1 plots each model's accuracy at each training set size.

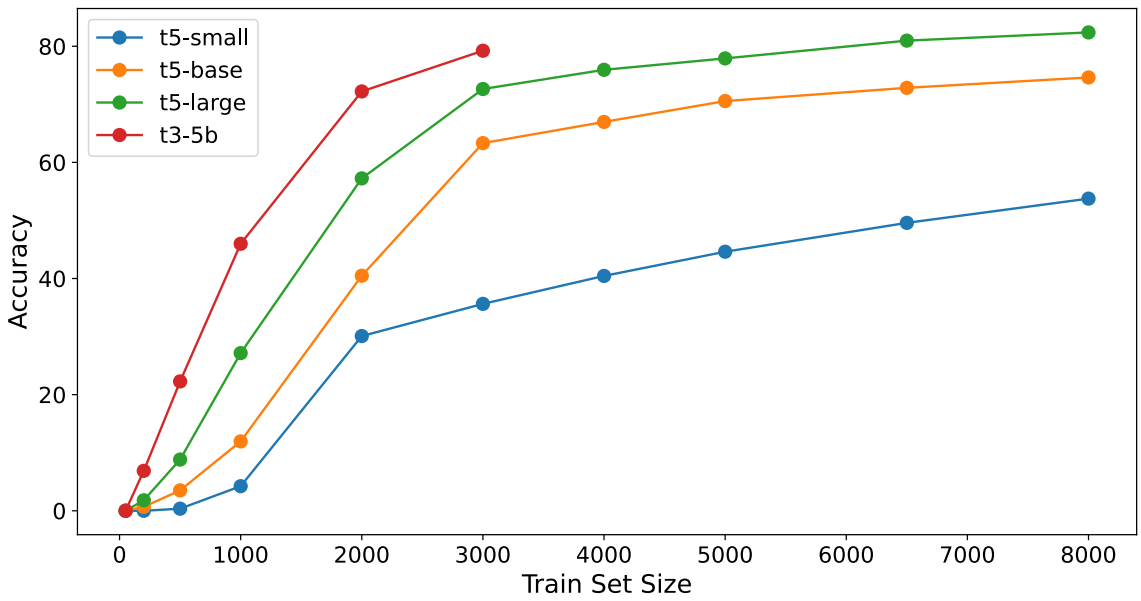


Figure 6.1: Performance of fine tuned T5 models of various sizes on attribution generation, reformulated as a extractive short-phrase generation task.

With $2,000$ training points, the `t5-3b` model achieves an accuracy of 72.22%. Larger models like `t5-large` and `t5-3b` produce structurally and syntactically sound predictions at 1000 training points. At $2,000$ training points, `t5-3b` consistently generates a correctly structured output consisting of five unique attributes. At the same training set size, as the model size increases, accuracy increases. If this trend persists, larger models such as `t5-11b` may offer greater accuracy at an affordable cost. This reformulation demonstrates promise as an efficient and faithful replacement for human generation.

The performance of the model on attribute generation is encouraging for future work towards automating instruction generation. This model could be supplied with a product's information, attributes, options, and price, then asked to output a natural language query. However, such a model might lean towards learning more extractive practices, which in turn could confine the diversity of the outputted instructions to a finite set of learned templates. On the other hand, a text generation model with a similar set of inputs and outputs could potentially devise richer queries at the cost of requiring more human-produced training data.

Chapter 7

Appendix

7.1 Environment Details

Product Scraping

We use ScraperAPI [35] to extract publicly available product information from `amazon.com`. We use five categories (beauty, food, fashion, furniture, electronics) and 313 associated sub-category names appeared in `amazon.com` (e.g. "Women's Loafers & Slip-Ons" in fashion, "Pendants and Chandeliers" in furniture) to scrape $1,181,436$ products. We filter products with duplicate titles or product IDs, but do not perform extra filtering in order to avoid selection bias. Specifically, as `amazon.com` has its own content screening process, we did not find any personally identifiable information or offensive content during random sampling checks.

Products	Unique Attributes	Avg Attributes	Unique Options	Avg Options
1,181,436	670	3.1	842,849	0.67

Table 7.1: Product item statistics.

Product Attribute Mining

We use `TfidfVectorizer` from `scikit-learn` to extract probable bi-grams as attributes from product title and descriptions for further annotation. We manually

inspect these attributes to keep only the *specific* and *human-readable* ones and filter out the rest. An attribute should be suitable in at least one of the following use: 1) `IsGoodFor`, 2) `HasA` (contains), 3) `WhichIs`, and 4) `IsA`. For example, attributes such as "oz ml" and "men women" will be filtered out since it's unparsable. On the other hand, "hair color" will also be filtered since it is not specific enough to fit in the above 4 categories. Attributes such as "dry skin" can fit the `IsGoodFor` in the context of a make-up product being good for dry skin.

Search Engine

Each time the agent performs a search, the top 50 items are retrieved and displayed across five search result pages, where each page contains 10 items and the agent can use actions `choose`[Prev/Next page] to navigate across result pages. Figure 3.2 shows that when searching directly with the instruction text, the corresponding item appears in the first search page (rank 1-10) nearly 1/3 of the time, but it cannot be found in any search pages (rank 50+) more than half of the time. This indicates that while the search engine can decently retrieve items based on lexical matching, directly searching the instruction is not enough for solving the task, and good query (re)formulation based on the instruction is important.

Instruction Collection

We collect human written instructions by providing the workers a product including the title, product category, and its set of attributes and options (Figure 7.1, 7.2). We conduct qualification task by having each participating workers to work on $2-5$ examples. We inspect and assign qualification to 213 workers to perform the instruction writing task. We pay for each example 0.15 dollars. We do not anticipate any potential participant risk.

Reward Calculation

The type reward r_{type} consists of 3 elements: 1) course-grain product category match ($c = 1$ if matched), 2) fine-grain category match ($f = 1$ if matched), and

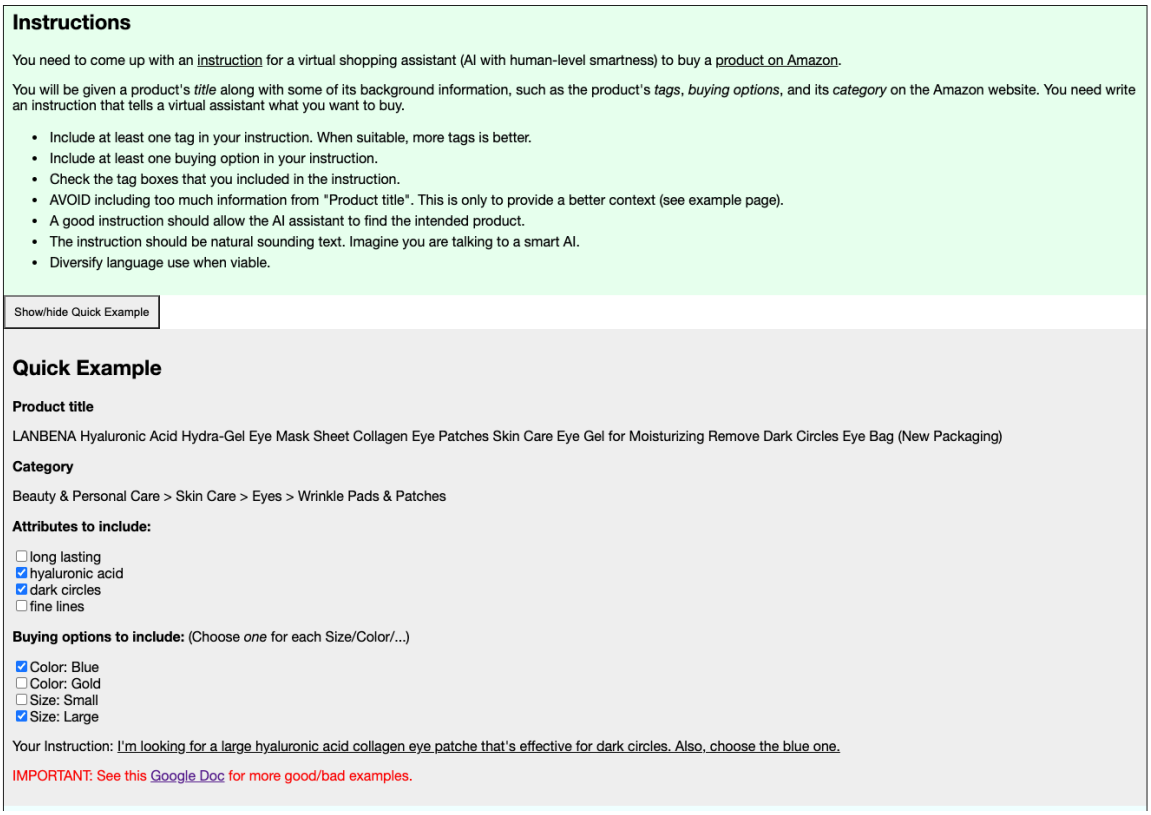


Figure 7.1: The Amazon Mechanical Turk interface for the instruction writing task. The green box shows the general instruction for the task and the grey box shows an concrete example.

3) product title match. Course-grain product category refers to the 5 categories described in §3.2. Fine-grain category is the chain of categories that the product is under on the Amazon website. For example, and eye mask sheet would be under the *Beauty & Personal Care* > *Skin Care* > *Eyes* > *Wrinkle Pads & Patches* fine-grain category. The product title refers to $\bar{y}$ described in §3.1.

$$r_{\text{type}} = \begin{cases} 0, & \text{if } \texttt{TextMatch}(\bar{y}, \bar{y}^*) = 0 \\ 0.1, & \text{if } \texttt{TextMatch}(\bar{y}, \bar{y}^*) < 0.1 \\ 0.5, & \text{if } \texttt{TextMatch}(\bar{y}, \bar{y}^*) > 0.2 \text{ and } c = 1 \text{ and } f = 1, \\ 1, & \text{otherwise} \end{cases} \tag{7.1}$$

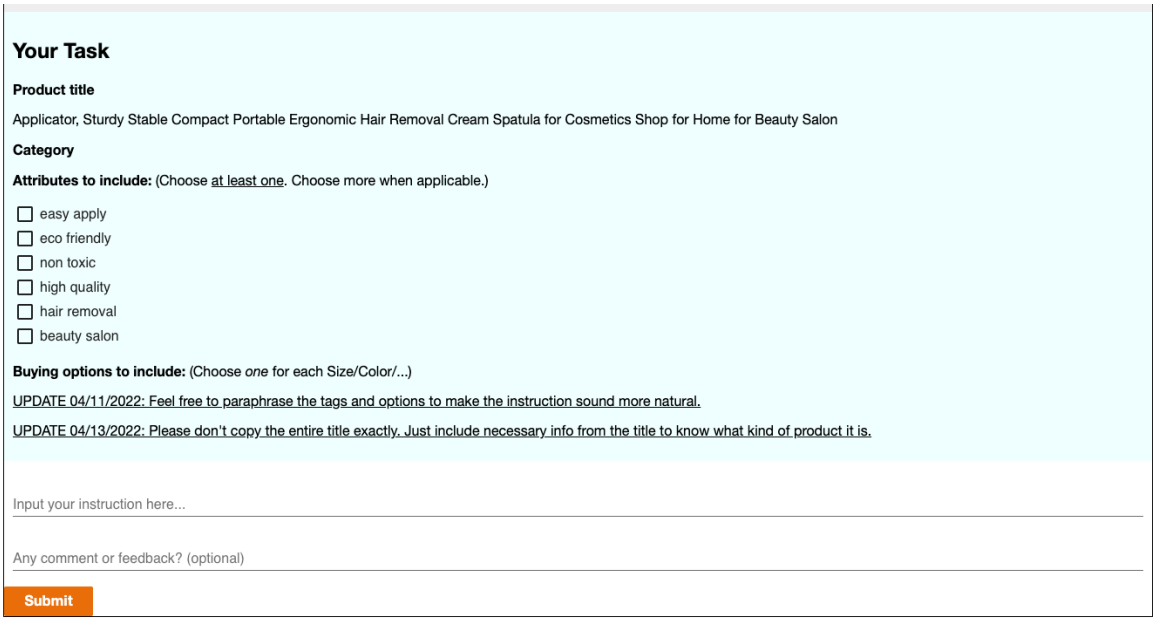


Figure 7.2: The Amazon Mechanical Turk interface for the instruction writing task. The blue box shows the actual annotation interface. The worker is required to check the boxes and write the instructions in the text field before submission.

Here, $\texttt{TextMatch}(\bar{y}, \bar{y}^*)$ is a simple string match between the selected product title text and the goal product title text. We use only the words tagged with PNOUN, NOUN, and PROPN tags parsed by the SpaCy parser in the title text.

Human Trajectory Collection

We use the `HTML` environment in Figure 1.1 to collect human trajectories. We select a pool of 13 workers using qualification tasks where each workers complete 5 examples. The workers that achieve an average reward more than 0.75 are qualified. The task instruction is shown at the end of Appendix. We observe a pronounced performance gap between the very high performing workers and average workers. We use the top 50% of these qualified workers as experts (7 workers in total). We pay for each completed trajectory 0.7 dollars. In human evaluation, 8 out of the 13 workers participated and 5 among them are in the aformentioned expert pool. The 8 participants achieve an overall score of 75.5 and a success rate of 50.0% We observe non-negligible variance even within the experts—the best performer achieves a score

Instruction 1: I would like a stained glass wall lamp with a bronze finish, and price lower than 190 dollars.	**Instruction 2** I would like a lead free bracelet birthday cake jar candle, and price lower than 50.00 dollars.
Human Actions ($r = 0.33$, length = 4) search[stained glass wall lamp] click[item-QCLU Tiffany Style Lamp Sunflower...] click[wall lamp 3 - 12 inch] click[buy]	**Human Actions** ($r = 0.03$, len = 4) search[lead free bracelet birthday cake jar candle] click[item-Happy Birthday Candle...] click[8 ounce round tin] click[buy]

Table 7.2: Two examples of failed human trajectories. A common pattern is impatience: after one search (even with correct attributes like the right example) the less performant worker commits to the first selected item. Often, the item does not contain the desired options even though the item's title text seem relevant. An expert worker will recognize the need to select the correct options and go back to refine the searches, while less performant workers simply commit to the current selected item.

of 87.4 and success rate of 69.5%, while the lowest performing worker achieves a score of 45.8 and success rate of 10%. The best performing worker also shows better consistency—drawing at a standard deviation of 2.3 in score, contrasting the lowest performing counterpart at 3.1. We provide examples of common human failure cases such as not matching the option/attribute due to impatience (Table 7.2), cautioning some caveats of the task with human workers.

Reward Verification.

We randomly select 100 samples each from the pools of trajectories generated by average and expert MTurk workers. Each trajectory is then manually re-scored against a human criteria; the purpose of this is to determine how representative the reward function is of a human's judgment towards whether the chosen product satisfies the given instructions. The human score calculation procedure exactly follows the formula laid out in Section 7.1 – the attribute, option, price, and type scores are individually determined, then aggregated to calculate the overall score – except for one main modification. Instead of the exact matching approach, points are awarded if (1) the picked product's attributes, options, or type are lexically similar or synonymous with the goal's product information and (2) the desired value is not found verbatim anywhere in the picked product's descriptions. For instance, if the value *lightweight*

is specified as a desired attribute for an instruction, but the value *easy carry* is found instead in the picked product's description, then the attribute score for the picked product is increased to reflect that the *lightweight* value was found. On the other hand, if *cyan* is desired as an option for a goal product, but the user picks *blue* even though *cyan* is available as a choice, then no points are awarded. To ensure the score is calculated without bias, the original rewards for each trajectory were not compared with the human evaluation scores until the human evaluation scoring was completed.

For the average trajectories, the automatic task score was 74.9 and our manual score was 76.3 with a Pearson correlation of 0.856. For expert trajectories, the respective scores were 81.5 and 89.9 with a Pearson correlation of 0.773. Therefore, the automatic reward seems to provide a reasonably close lower bound to the actual task performance. We find that for average workers, 87.0% of automatic scores are within a 10% of the manual score, with the main source of error being synonyms or lexically similar words that don't get matched correctly in the automatic reward function.

MTurk Type	Reward Function	Price	Type	Attribute	Result	Overall
Average	WebShop	95.0	92.9	71.7	50.5	74.9
	Human	95.0	93.8	75.3	57.0	76.3
Expert	WebShop	100.0	100.0	78.1	56.1	81.5
	Human	100.0	100.0	88.2	66.8	89.9

Table 7.3: Reward Verification Statistics

Table 7.3 reflects our observation that our reward function is similar to a human's score, with a consistent tendency to over-penalize the picked product. For every trajectory's product, the human score across all categories (e.g. attributes, options) is always greater than or equal to the original score. This under-scoring is a result of our reward function's exact matching criterion. In future work, we hope to improve our matching functionality such that, within the context of a single product with respect to the goal instructions, it can identify synonyms and decide whether to award additional points.

7.2 Model Details

Cross Attention Layer

Our cross attention layer follows Seo et al. [43]. Denote the i-th contextualized token embedding from the observation and action to be $\mathbf{o}_i$ and $\mathbf{a}_i$ respectively. The attention between $\mathbf{o}_i$ and $\mathbf{a}_j$ is defined as

$$\alpha_{ij} = \mathbf{w}_1 \cdot \mathbf{o}_i + \mathbf{w}_2 \cdot \mathbf{a}_j + \mathbf{w}_3 \cdot (\mathbf{o}_i \otimes \mathbf{a}_j) \tag{7.2}$$

where $\otimes$ denotes element-wise product and $\mathbf{w}_1, \mathbf{w}_2, \mathbf{w}_3$ are learnable vectors. The observation-contextualized vector for j-th action token is then

$$\mathbf{ca}_j = \mathbf{w}_5 \cdot \text{leakyRELU}(\mathbf{w}_4 \cdot [\mathbf{a}_j, \mathbf{c}_j, \mathbf{a}_j \otimes \mathbf{c}_j, \mathbf{q} \otimes \mathbf{c}_j]) \tag{7.3}$$

$$\mathbf{c}_j = \frac{\sum_i \exp(\alpha_{ij}) \cdot \mathbf{o}_i}{\sum_i \exp(\alpha_{ij})}, \quad \mathbf{q} = \frac{\sum_{j'} \exp(\max_i \alpha_{ij'}) \mathbf{a}_{j'}}{\sum_{j'} \exp(\max_i \alpha_{ij'})} \tag{7.4}$$

We then average pool all $\mathbf{ca}_j$ to derive the action score $S(o, a)$:

$$S(o, a) = \mathbf{w}_6 \cdot \frac{1}{n_a} \sum_{j \leq n_a} \mathbf{ca}_j \in \mathbb{R} \tag{7.5}$$

where n_a is the number of tokens for action a.

RNN Baseline

Our RNN baseline is inspired by Guo et al. [14], where we use the same attention layer as described above, but replace the Transformer text encoder with one-layer bi-directional Gated Recurrent Units (GRU) [9] of hidden dimension 512. Another difference is that we also add an cross attention between the instruction and action input word embeddings, as we hypothesize it might help option text matching.

7.3 WebShop Experiment Details

IL Training Details

The training code for our IL models is adapted from Huggingface's glue training example, whose repository is licensed under Apache License 2.0. We use a training batch size of 1 with 32 gradient accumulation steps, a learning rate of 2×10^{-5}, and 10 training epochs. The training takes around 2 hours on one RTX 2080 GPU with a GPU memory of around 10GB.

RL Training Details

We train the RL models using 4 parallel environments for $100,000$ training steps. The backprogation through time (BPTT) is taken at every 8 steps. We use an Adam optimizer with a learning rate of 10^{-5} (for Transformer models) or 5×10^{-4} (for RNN models).

For RL models with the Transformer (BERT) architecture, it takes around 27 hours on one RTX 3090 GPU with a GPU memory of around 20GB. For RL models with the GRU architecture, it takes around 20 hours on one RTX 2080 GPU with a GPU memory of around 10GB.

To disentangle the effects of learning to search from choosing the right actions, we construct a `Choice` oracle that has access to the hidden reward function as well as hidden attributes and options underlying each product and instruction. Given a search query, the `Choice` oracle will perform an exhaustive search over every result item, try out all available options and finally choose the best item with options that maximize the reward — meaning each episode will take more than a hundred steps, as opposed to 4.5 and 11.3 steps on average for the IL+RL model and human experts (Table 5.1). We use 500 test instructions and consider four types of search queries: the instruction text (used by rule baseline), top IL BART generated query (used by all learning models), and the first and last queries from human experts in each test trajectory. `Choice` improves the success rate of rule heuristics from 9.6% to 52.6%,

	Instr. text	IL BART	Human expert (first)	Human expert (last)
Score	79.7	83.0	82.1	**84.4**
SR	52.6%	57.6%	57.9%	**61.0%**

Table 7.4: Task performance with the `Choice` oracle. *first* and *last* refer to the first and last search queries found in human demonstrations, respectively.

		Rule	IL	IL+RL	Human
Amazon	Score	45.8	61.5	65.9	**88.0**
	SR	19%	27%	25%	**65%**
eBay	Score	31.7	58.2	62.3	**79.7**
	SR	7%	21%	21%	**40%**

Table 7.5: Zero-shot sim-to-real transfer to Amazon and eBay over 100 test instructions.

	Score	SR
IL	60.56 (1.94)	29.00 (2.42)
IL (top-1 search)	61.96 (0.47)	30.80 (0.72)
IL (top-1 choice)	45.10 (3.50)	24.93 (3.14)

Table 7.6: Sampling vs. top-1

and the IL model from 29.1% to 57.6% (Table 7.4), confirming that choosing the right actions is indeed a major bottleneck for current models with great room for improvement. However, it does not impact human performance much since they are likely good at making good choices.

Sampling vs. Top-1

We show comparisons between using beam search vs. top-1 for both the search model and the choice model in Table 7.6. During testing, the search model uses beam search to generate top-5 search queries. We randomly and uniformly sample from the top-5 queries to increase search diversity in case of multiple searches. We conduct experiments to instead always use the top-1 search, which shows slight performance improvement (see table below), and we will include the result in the paper. The choice model has a fixed set of action candidates at each step (e.g. all available buttons), and we sample from the choice policy what action to take, as always taking the top action will lead to significantly detorior performances.

Image Ablation

We train 3 trials with different random seeds for both the IL model and the ablated IL model without images, with performances over 500 test cases (7.7). Removing im-

	Score	SR
IL	60.6 (1.94)	29.0 (2.42)
IL (w/o image)	60.3 (0.47)	28.4 (0.87)

Table 7.7: Image ablations.

age only slightly hurts the overall performance, but significantly reduces the variance. This is reasonable as our current instruction and reward setups only use textual information, and we believe future efforts to incorporate visual information into the task setup will better challenge models' visual understanding, and make pre-trained vision-language models such as CLIP more useful.

7.4 Sim-to-real Details

Sim-to-real Transfer Details

To test how well our IL agent trained in WebShop performs on `amazon.com` (`ebay.com` similarly), we wrote a series of scripts that generally achieve two steps - translate a real Amazon URL into our IL model's input (text observation, set of valid actions) and map the model's output back to a real Amazon URL. The following procedure is repeated until the IL model generates a "buy now" action:

- Amazon URL → Amazon HTML → Amazon Page Information: Using ScraperAPI [35], we first get the HTML source code for a given Amazon page, then extract information relevant to rendering the equivalent page in the WebShop environment (e.g. title, price, options).
- Amazon Page Information → WebShop HTML → Text Observation: Given the scraped information, we generate the corresponding WebShop page in `HTML` mode, then transform it into a `simple` mode text observation.
- Amazon Page Information → Valid Action Set: From the scraped information, we determine what valid actions the model can take (i.e. `search[`Red shoes`]`,

`choose[Size 9]`). This logic is captured as a mapping of page type to permissible actions.

- Text Observation, Valid Action Set → IL Model → Amazon URL: Given the text observation and allowed of valid actions, the IL model produces an action. This action is then used to construct a corresponding Amazon URL via a set of mapping rules, and the loop is repeated. This continues until the model generates a "buy now" action, terminating the loop.

Sim-to-real Transfer Results

The resulting numbers in Table 5.4 closely cohere to the reported numbers of WebShop found in Figure 5.1, suggesting that the WebShop has promise for developing grounded agents that can operate on real web environments. Between the two websites, transfer to Amazon is better than eBay as we note that (i) eBay has a larger product gap from WebShop, e.g. some item categories like food are disallowed in eBay. (ii) the eBay search engine seems weaker, and would sometimes display no results for lengthy instructions. The following Table 7.8 is an example of a trajectory generated by the IL agent searching on the real Amazon website.

Instruction: I want to find white blackout shades that are 66 inches in width and 66 inches in height. They need to be easy to install..
search[white blackout shades 66 inches in width and 66 inches height, easy to install] click[item - Easy Up & Down 100% Blackout Pleated Window Shades Temporary Window Blinds 36in x 64in (Fits Window Width 18"-36") 2pcs-Pack Operating with Pull Cord Easy Trimming & Installing] click[features] click[back to search] search[white blackout shades that are 66 inches in width and 66 inches height] click[item - Redi Shade Inc 1617201 Original Blackout Pleated Paper Shade Black 36" x 72" 6-Pack] click[¡ prev] click[Shade + Strips, White] click[buy]

Table 7.8: An example trajectory (showing only actions) from the IL agent on the real Amazon website. We omit instructions and some human actions for instruction and trim item names for readability. Red denotes options and blue denotes attributes.

It is evident that the exploratory behavior and patterns learned and exhibited by the agent within the WebShop environment is not lost in this transfer. These

results point to the opportunity for sim-to-real trained agents to transfer to other real-world web tasks despite the domain shift in both data (products) and dynamics (search engine) With that said, the gap between human and model performance also encourage us to look into expanding on the current limitations in our work regarding both the model and the WebShop environment.

7.5 Potential Societal Impacts and Limitations

WebShop is designed to minimize human efforts in data collection and processing, but there are still potential concerns regarding diversity, fairness, and representation. Developing RL agents that interact with the web also bear safety concerns, especially when transferring from simulation to real-world websites. We also discuss other limitations regarding the semantics of current task (instruction/reward).

Diversity and representation in data collection

We chose five common categories from `amazon.com` and scrape all products using all subcategories to minimize bias. However, our data is still biased toward the website country (USA) and website language (English), and may only represent a subset of all possible products that users potentially want to buy. Having this limitation in mind, the design of WebShop allows the product data to be easily updated for different representations of real-world usage.

Bias in data processing

Currently our attribute labeling is manually done and may be biased by the labeler's own experience (e.g. more knowledge toward product attributes like sports rather than makeup). An more automatic alternative would be to employ trained NLP models (e.g. relation extraction) to extract product attributes, which might be less biased than one labeller. Our reward design is general and could be updated to weight more toward attributes, options, price, etc.

Safety for developing web agents

Unlike recent work [33] that directly employs agents on the World Wide Web (WWW), WebShop aims to provide a realistic simulation environment to train agents in a controllable and safe manner. In our preliminary sim-to-real experiments, the agent could only update the current webpage's url in two fixed and safe ways (i.e. search for results, open an item), and any form sending action (e.g. click options or buy) is held within the sim-to-real interface for later reward calculation. As a result, only navigation is done on the real-world website. For future deployment to real-world websites with more advanced functions, we believe a good specification of possible model behaviors is key to avoid harmful actions.

Limitations in the current task

Our current instructions are still limited by the attributes and options used. While attributes are simple and sometimes too generic (e.g. "easy to use"), the options might get too specific (e.g. "d17(dedicated right, back)"). Therefore, an agent might sometimes use a special option as cues to find the product, while ignoring other parts of the instruction. To better leverage images and texts (including reviews written by human users, which are not used in current work) of products for more semantic and challenging instructions is an important future direction from WebShop.

Chapter 8

Conclusions

We have developed WebShop, a new web-based benchmark for sequential decision making and language grounding, modeled on interaction with an e-commerce website. We performed an empirical evaluation of autonomous agents trained using imitation and reinforcement learning, and demonstrated promising results on sim-to-real transfer to real-world shopping websites. The qualitative and quantitative analysis of model and human trajectories (§5.3) identified several research challenges in WebShop and provided insights for future model development by incorporating multidisciplinary techniques. For example, pre-training with multi-modal data [27, 53], web hypertext [2], or web instruction-action mapping [38] could help agents better understand and leverage rich semantics of webpage content, actions, and instructions. Ideas from query (re)formulation [22, 60, 37, 52] may help agents expand the range of search exploration, and improved action exploration [40, 12, 50] and memory [54, 13, 23] mechanisms could help agents make better decisions over the long horizon and large action space. The modular design of WebShop also allows for new web tasks and domains to be easily incorporated, which we hope will help shape future research into grounded language agents with stronger capabilities for real-world web interaction.

Bibliography

[1] L. Adolphs, B. Boerschinger, C. Buck, M. C. Huebscher, M. Ciaramita, L. Espeholt, T. Hofmann, and Y. Kilcher. Boosting Search Engines with Interactive Agents. *arXiv preprint arXiv:2109.00527*, 2021.

[2] A. Aghajanyan, D. Okhonko, M. Lewis, M. Joshi, H. Xu, G. Ghosh, and L. Zettlemoyer. Htlm: Hyper-text pre-training and prompting of language models. *ArXiv*, abs/2107.06955, 2021.

[3] J. Andreas, J. Bufe, D. Burkett, C. Chen, J. Clausman, J. Crawford, K. Crim, J. DeLoach, L. Dorner, J. Eisner, et al. Task-oriented dialogue as dataflow synthesis. *Transactions of the Association for Computational Linguistics*, 8:556–571, 2020.

[4] E. M. Bender and A. Koller. Climbing towards NLU: On Meaning, Form, and Understanding in the Age of Data. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5185–5198, 2020.

[5] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. OpenAI Gym. *arXiv preprint arXiv:1606.01540*, 2016.

[6] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[7] P. Budzianowski, T.-H. Wen, B.-H. Tseng, I. Casanueva, S. Ultes, O. Ramadan, and M. Gašić. Multiwoz–a large-scale multi-domain wizard-of-oz dataset for task-oriented dialogue modelling. *arXiv preprint arXiv:1810.00278*, 2018.

[8] A. Burns, D. Arsan, S. Agrawal, R. Kumar, K. Saenko, and B. A. Plummer. Interactive Mobile App Navigation with Uncertain or Under-specified Natural Language Commands. *arXiv preprint arXiv:2202.02312*, 2022.

[9] J. Chung, Çaglar Gülçehre, K. Cho, and Y. Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *ArXiv*, abs/1412.3555, 2014.

[10] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *ArXiv*, abs/1810.04805, 2019.

[11] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT (1)*, 2019.

[12] A. Ecoffet, J. Huizinga, J. Lehman, K. O. Stanley, and J. Clune. Go-explore: a new approach for hard-exploration problems. *arXiv preprint arXiv:1901.10995*, 2019.

[13] M. Fortunato, M. Tan, R. Faulkner, S. Hansen, A. Puigdomènech Badia, G. Buttimore, C. Deck, J. Z. Leibo, and C. Blundell. Generalization of reinforcement learners with working and episodic memory. *Advances in neural information processing systems*, 32, 2019.

[14] X. Guo, M. Yu, Y. Gao, C. Gan, M. Campbell, and S. Chang. Interactive fiction game playing as multi-paragraph reading comprehension with reinforcement learning. *arXiv preprint arXiv:2010.02386*, 2020.

[15] I. Gur, U. Rueckert, A. Faust, and D. Hakkani-Tur. Learning to Navigate the Web. *arXiv preprint arXiv:1812.09195*, 2018.

[16] I. Gur, N. Jaques, K. Malta, M. Tiwari, H. Lee, and A. Faust. Adversarial Environment Generation for Learning to Navigate the Web. *arXiv preprint arXiv:2103.01991*, 2021.

[17] M. Hausknecht, P. Ammanabrolu, M.-A. Côté, and X. Yuan. Interactive fiction games: A colossal adventure. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 7903–7910, 2020.

[18] K. He, X. Zhang, S. Ren, and J. Sun. Deep Residual Learning for Image Recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[19] A. Hotti, R. S. Risuleo, S. Magureanu, A. Moradi, and J. Lagergren. The Klarna Product Page Dataset: A RealisticBenchmark for Web Representation Learning. *arXiv preprint arXiv:2111.02168*, 2021.

[20] P. C. Humphreys, D. Raposo, T. Pohlen, G. Thornton, R. Chhaparia, A. Muldal, J. Abramson, P. Georgiev, A. Goldin, A. Santoro, et al. A data-driven approach for learning to control computers. *arXiv preprint arXiv:2202.08137*, 2022.

[21] S. Jia, J. Kiros, and J. Ba. Dom-q-net: Grounded RL on Structured Language. *arXiv preprint arXiv:1902.07257*, 2019.

[22] M. Komeili, K. Shuster, and J. Weston. Internet-augmented dialogue generation. *arXiv preprint arXiv:2107.07566*, 2021.

[23] A. Lampinen, S. Chan, A. Banino, and F. Hill. Towards mental time travel: a hierarchical memory for reinforcement learning agents. *Advances in Neural Information Processing Systems*, 34:28182–28195, 2021.

[24] A. Lazaridou, A. Potapenko, and O. Tieleman. Multi-agent Communication meets Natural Language: Synergies between Functional and Structural Language Learning. In *ACL*, 2020.

[25] A. Lazaridou, E. Gribovskaya, W. Stokowiec, and N. Grigorev. Internet-augmented language models through few-shot prompting for open-domain question answering. *ArXiv*, abs/2203.05115, 2022.

[26] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. *arXiv preprint arXiv:1910.13461*, 2019.

[27] X. Li, X. Yin, C. Li, P. Zhang, X. Hu, L. Zhang, L. Wang, H. Hu, L. Dong, F. Wei, et al. Oscar: Object-semantics aligned pre-training for vision-language tasks. In *European Conference on Computer Vision*, pages 121–137. Springer, 2020.

[28] J. Lin, X. Ma, S.-C. Lin, J.-H. Yang, R. Pradeep, and R. Nogueira. Pyserini: An Easy-to-Use Python Toolkit to Support Replicable IR Research with Sparse and Dense Representationss. *arXiv preprint arXiv:2102.10073*, 2021.

[29] E. Z. Liu, K. Guu, P. Pasupat, T. Shi, and P. Liang. Reinforcement Learning on Web Interfaces using Workflow-Guided Exploration. *arXiv preprint arXiv:1802.08802*, 2018.

[30] J. Luketina, N. Nardelli, G. Farquhar, J. N. Foerster, J. Andreas, E. Grefenstette, S. Whiteson, and T. Rocktäschel. A survey of reinforcement learning informed by natural language. In *IJCAI*, 2019.

[31] S. Mazumder and O. Riva. FLIN: A Flexible Natural Language Interface for Web Navigation. *arXiv preprint arXiv:2010.12844*, 2020.

[32] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937. PMLR, 2016.

[33] R. Nakano, J. Hilton, S. Balaji, J. Wu, L. Ouyang, C. Kim, C. Hesse, S. Jain, V. Kosaraju, W. Saunders, et al. WebGPT: Browser-Assisted Question-Answering with Human Feedback. *arXiv preprint arXiv:2112.09332*, 2021.

[34] K. Narasimhan, A. Yala, and R. Barzilay. Improving Information Extraction by Acquiring External Evidence with Reinforcement Learning. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2355–2365, 2016.

[35] D. Ni. ScraperAPI, 2015. URL `https://www.scraperapi.com/`.

[36] R. Nogueira and K. Cho. End-to-End Goal-Driven Web Navigation. *Advances in Neural Information Processing Systems*, 29, 2016.

[37] R. Nogueira and K. Cho. Task-Oriented Query Reformulation with Reinforcement Learning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 574–583, 2017.

[38] P. Pasupat, T.-S. Jiang, E. Z. Liu, K. Guu, and P. Liang. Mapping natural language commands to web elements. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2018.

[39] P. Pasupat, T.-S. Jiang, E. Z. Liu, K. Guu, and P. Liang. Mapping natural language commands to web elements. In *EMNLP*, 2018.

[40] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell. Curiosity-driven exploration by self-supervised prediction. In *International conference on machine learning*, pages 2778–2787. PMLR, 2017.

[41] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer, 2019. URL `https://arxiv.org/abs/1910.10683`.

[42] A. Ronacher. Flask API, 2010. URL `https://flask.palletsprojects.com/en/2.1.x/`.

[43] M. Seo, A. Kembhavi, A. Farhadi, and H. Hajishirzi. Bidirectional attention flow for machine comprehension. *arXiv preprint arXiv:1611.01603*, 2016.

[44] T. Shi, A. Karpathy, L. Fan, J. Hernandez, and P. Liang. World of Bits: An Open-Domain platform for web-based agents. In *International Conference on Machine Learning*, pages 3135–3144. PMLR, 2017.

[45] M. Shridhar, J. Thomason, D. Gordon, Y. Bisk, W. Han, R. Mottaghi, L. Zettlemoyer, and D. Fox. 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, 2020.

[46] K. Shuster, M. Komeili, L. Adolphs, S. Roller, A. D. Szlam, and J. Weston. Language models that seek for knowledge: Modular search & generation for dialogue and prompt completion. *ArXiv*, abs/2203.13224, 2022.

[47] Y. Su, A. H. Awadallah, M. Khabsa, P. Pantel, M. Gamon, and M. Encarnacion. Building Natural Language Interfaces to Web APIs. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 177–186, 2017.

[48] Y. Su, A. Hassan Awadallah, M. Wang, and R. W. White. Natural Language Interfaces with Fine-Grained User Interaction: A Case Study on Web APIs. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 855–864, 2018.

[49] D. Toyama, P. Hamel, A. Gergely, G. Comanici, A. Glaese, Z. Ahmed, T. Jackson, S. Mourad, and D. Precup. AndroidEnv: A Reinforcement Learning Platform for Android. *arXiv preprint arXiv:2105.13231*, 2021.

[50] J. Tuyls, S. Yao, S. Kakade, and K. Narasimhan. Multi-stage episodic control for strategic exploration in text games. *arXiv preprint arXiv:2201.01251*, 2022.

[51] V. Uc-Cetina, N. Navarro-Guerrero, A. Martin-Gonzalez, C. Weber, and S. Wermter. Survey on reinforcement learning for language processing. *arXiv preprint arXiv:2104.05565*, 2021.

[52] X. Wang, C. Macdonald, and I. Ounis. Deep reinforced query reformulation for information retrieval. *arXiv preprint arXiv:2007.07987*, 2020.

[53] Z. Wang, J. Yu, A. W. Yu, Z. Dai, Y. Tsvetkov, and Y. Cao. Simvlm: Simple visual language model pretraining with weak supervision. *arXiv preprint arXiv:2108.10904*, 2021.

[54] G. Wayne, C.-C. Hung, D. Amos, M. Mirza, A. Ahuja, A. Grabska-Barwinska, J. Rae, P. Mirowski, J. Z. Leibo, A. Santoro, et al. Unsupervised predictive memory in a goal-directed agent. *arXiv preprint arXiv:1803.10760*, 2018.

[55] K. Williams, S. H. Hashemi, and I. Zitouni. Automatic Task Completion Flows from Web APIs. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1009–1012, 2019.

[56] S. Yao, R. Rao, M. J. Hausknecht, and K. Narasimhan. Keep CALM and Explore: Language Models for Action Generation in Text-based Games. *ArXiv*, abs/2010.02903, 2020.

[57] S. Yao, K. Narasimhan, and M. Hausknecht. Reading and acting while blindfolded: The need for semantics in text game agents. *arXiv preprint arXiv:2103.13552*, 2021.

[58] X. Yuan, J. Fu, M.-A. Côté, Y. Tay, C. J. Pal, and A. Trischler. Interactive machine comprehension with information seeking agents. In *ACL*, 2020.

[59] V. Zhong, A. W. Hanjie, S. Wang, K. Narasimhan, and L. Zettlemoyer. Silg: The multi-domain symbolic interactive language grounding benchmark. *Advances in Neural Information Processing Systems*, 34:21505–21519, 2021.

[60] S. Zhuang, H. Ren, L. Shou, J. Pei, M. Gong, G. Zuccon, and D. Jiang. Bridging the gap between indexing and retrieval for differentiable search index with query generation. *arXiv preprint arXiv:2206.10128*, 2022.