WebShop: Towards Scalable Real-World Web Interaction with Grounded Language Agents

Anonymous Author(s)
Affiliation
Address
email

Abstract

Existing benchmarks for grounding language in interactive environments either lack real-world linguistic elements, or prove difficult to scale up due to substantial human involvement in the collection of data or feedback signals. To bridge this gap, we develop WebShop – a simulated e-commerce website environment with 1.18 million real-world products and $12,087$ crowd-sourced text instructions. 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 provides several challenges for language grounding including understanding compositional instructions, query (re-)formulation, comprehending and acting on noisy text in webpages, and performing strategic exploration. We collect over $1,600$ human demonstrations for the task, and train and evaluate a diverse range of agents using reinforcement learning, imitation learning, and pre-trained image and language models. Our 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%). We also analyze agent and human trajectories and ablate various model components to provide insights for developing future agents with stronger language understanding and decision making abilities. Finally, we show that 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 in developing practical web-based agents that can operate in the wild.

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 [27, 46, 53]. On the other hand, large-scale language models like GPT-3 [5] and BERT [9] are excelling 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 [3]. In order to make further progress in building **grounded** language models, we believe there is a need for **scalable** interactive environments that contain: (1) language elements that reflect rich, real-world usage and are collectible at scale, and (2) task feedback that is well-defined and automatically computable to facilitate interactive learning, without the constant need for expensive feedback from humans.

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

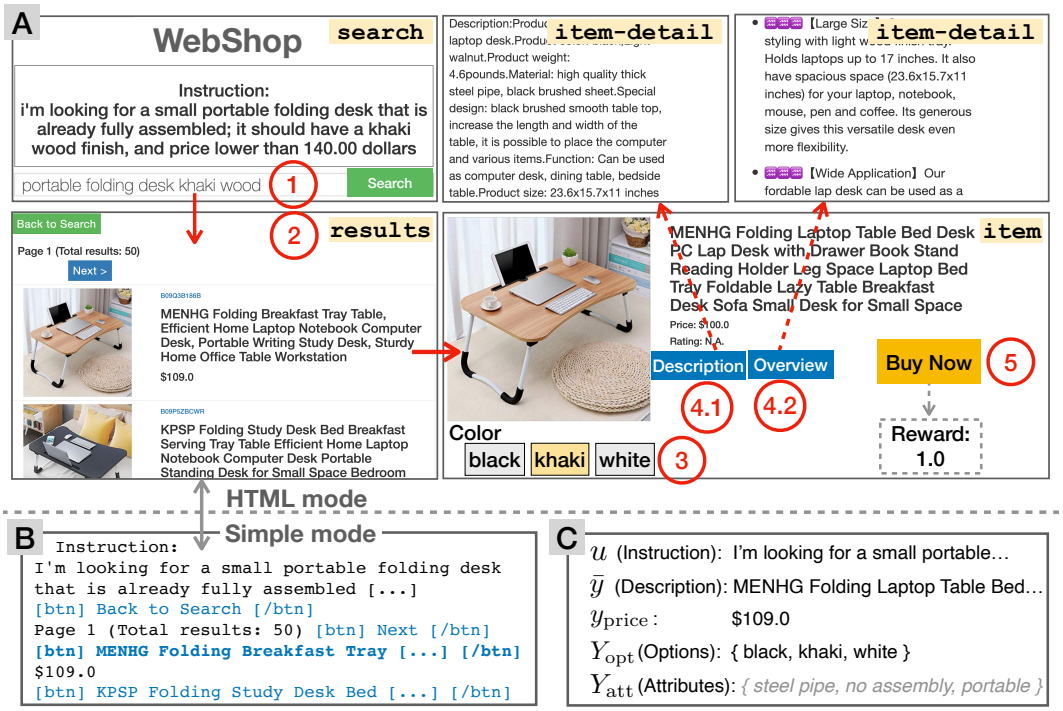


Figure 1: The WebShop environment. **A**: An example task trajectory in HTML mode, where a user can (1) search a query in a `search` page, (2) click a product item in a `results` page, (3) choose a color option in a `item` page, (4) check `item-detail` pages and go back to the item page, and (5) finally 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 with corresponding examples from the product in **A**. The attributes Y_{att} are hidden from the task performer.

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 with natural 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. Moreover, the web also provides a practical environment to deploy trained agents, with great potential for alleviating human efforts in tedious tasks (e.g. buying products, booking appointments). 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 [36, 42, 28] or interactions containing only a handful of different pages in each episode [40]. Others propose tasks with longer horizons but are either limited to following hyperlinks for web navigation [33] or require human-in-the-loop feedback due to the lack of an automated reward function [30].

In this paper, we introduce WebShop (Figure 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 understand a human-provided text instruction and *purchase* a product to match the specifications. To do so, the agent needs to query the website's search engine, choose items to explore from search results, open and read their description and details, and select the necessary options (e.g. 32 oz., red color) before clicking the 'Buy' button. In order to pick the optimal product that matches user requirements, the agent may need to view and compare various products (including backtracking between pages), and potentially perform multiple searches. 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 [4] 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 and providing a path to scaling up interactive learning.

We also develop several agents for performing this task, using both reinforcement learning (RL) and imitation learning (IL) in addition to leveraging the latest pre-trained language models [23, 9] for representing and generating text. Our modular architecture includes a factorized processing of state observations and action choices using ResNets (visual) and Transformers (text), followed by an attention fusion layer that helps the agent contextually score each action. Our best agent achieves an average score of 62.4 (out of 100) and successfully completes the task 28.7% of the time, significantly higher than a heuristic baseline that achieves 45.6 and 9.6%, respectively. While this demonstrates the potential for IL and RL, the agents are still much lower than human experts, who can achieve 82.1 and 59.6% on this task.[1] We perform several analyses and ablation studies to identify the cause of this gap and find several avenues for agent improvement in the future including more robust search generation, explicit memory modules, and better handling of noisy web text. Finally, we also demonstrate an instance of *sim-to-real* transfer by deploying agents trained with WebShop to operate on `amazon.com` and `ebay.com`, and find that they can achieve similar performances despite search engine and product differences, and consistently outperform the rule baseline of using the first result returned by the commercial search engines when directly searching the instruction texts. This demonstrates the practical potential of our work towards developing agents that can operate autonomously on the world wide web (WWW).

2 Related Work

Reinforcement learning on the web. Nogueira and Cho [33] 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 [40] 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 [26], curriculum and meta-learning [13], DOM tree representation [18], adversarial environment generation [14] and large-scale behavioral cloning [17]. 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 [44] and MoTIF [6] 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 [36], generating API calls [42, 43, 50] and semantic parsing into concept-level navigation actions [28]. Perhaps most similar content-wise to our work is the Klarna product page dataset [16] 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 [31] and retrieval [1], question answering [52, 22], dialog [41], and training language models on webtext [2]. These approaches primarily use web search engines as a knowledge

[1] In our analysis (§5.3), we observe that the task requires patience and consistency, which is lacking in some crowdsource workers, leading to lower scores. Even with this caveat, the gap between human performance and the model remains significant.

retriever for gathering additional evidence for the task at hand. Perhaps most similar to our work is WebGPT [30], 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.

3 The WebShop Environment

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 $\mathcal{T} : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$, reward function $\mathcal{R} : \mathcal{S} \times \mathcal{A} \rightarrow [0, 1]$, instruction space $\mathcal{U}$, and a state observation space $\mathcal{O}$.

Action 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 1: Actions & transitions in WebShop.

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 1A(1-4) 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 (§3.2). 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]`) as shown in Table 1. 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 [40]. The transitions initiated by clicks deterministically redirect the web page to one of the four page types (Table 1). The transition initiated by search is based on a deterministic search engine (§3.2).

Observation. Using Flask [38] and OpenAI Gym [4], 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 1A), and (2) `simple` mode which strips away extraneous meta-data from raw HTML into a simpler format (Figure 1B). The human performance scores in §4.2 are collected in the `HTML` mode, while all models are trained and evaluated in the `simple` mode. Note that while the environment allows for training reinforcement learning agents on raw pixels in `HTML` mode (like in Shi et al. [40]), 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:

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

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". The exact formula for $\texttt{TextMatch}(\cdot)$ is in the Appendix §A.5.

Evaluation metrics. We use two evaluation metrics: (1) **Task Score**: defined as $(100 \times \text{avg. reward})$, 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^* — for example, there could be many items that satisfy the goal "I want a red shirt", even if the goal is generated from a specific red shirt item.

3.2 Environment Implementation

Data scraping. We use ScraperAPI [32] 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. More details about product scraping is in the Appendix §A.1.

Search engine. We use Pyserini [25] 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. More details in A.3.

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. An attribute is a short natural language phrase that describes the property of the product (see examples in Figure 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. See more details in the Appendix §A.2.

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. We provide the detailed annotation process and interface in the Appendix §A.4.

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 Appendix §A.6 for examples). We also leverage this data to perform imitation learning (described in §4.2).

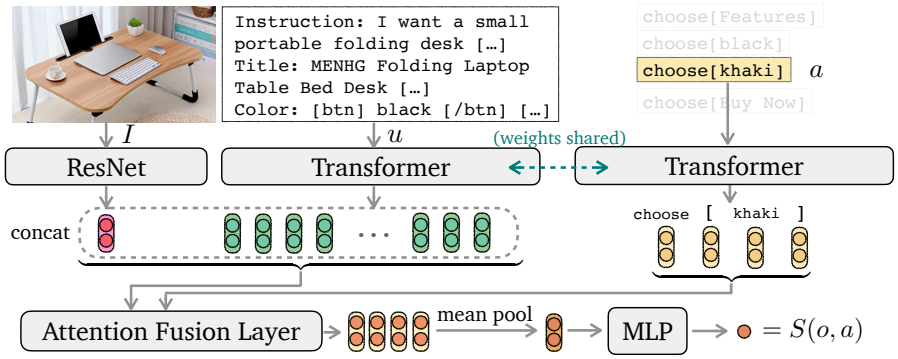


Figure 2: Architecture of our choice-based imitation learning (IL) model. The image I is passed to a ResNet to obtain the image representation. The instruction text u is passed to a transformer (initialized with BERT) to obtain the text representations. The concatenated bi-modal representations are fused 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]`.

4 Methods

We propose various models that combine language and image pre-training with imitation learning (IL) and reinforcement learning (RL). More details are provided in the Appendix §B.

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 $a =$ `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 [23] parameterized by ϕ to perform conditional language modeling:

$$\mathcal{L}_{\text{search}} = \mathbb{E}_{u,a\sim\mathcal{D}}\left[-\log \pi_\phi(a \mid u)\right] \tag{2}$$

Imitating human choice. The choice-based imitation model (Figure 2) 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 [8] 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 $S(o, a)$:

$$\mathcal{L}_{\text{choose}} = \mathbb{E}_{o,\mathcal{A}(o),a^*\sim\mathcal{D}'}\left[-\log \pi_\theta\left(a^* \mid o, \mathcal{A}(o)\right)\right] \tag{3}$$

$$\pi_\theta\left(a \mid o, \mathcal{A}(o)\right) \sim \exp\left(W^\top \text{mean}\big[\text{cross-attn}\big(\text{BERT}(o;\theta), \text{BERT}(a;\theta)\big)\big]\right) \tag{4}$$

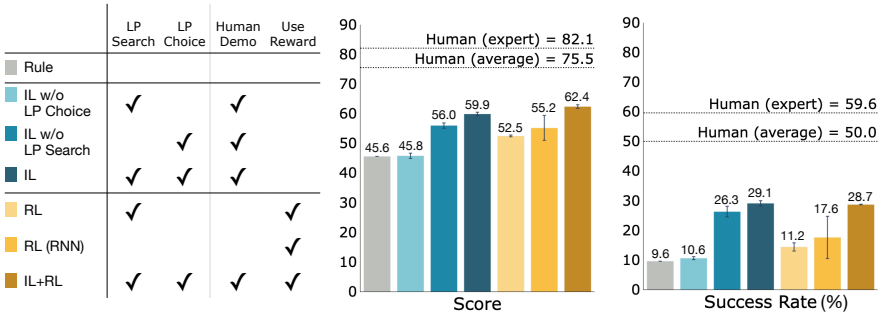


Figure 3: Task scores and Success Rate (%) for our models on the test split of WebShop over 3 trials. LP Search uses a pre-trained BART model to generate the search query and IL w/o LP Search uses the rule-based heuristic. LP Choice uses pre-trained BERT weights to initialize the choice action model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

Handling Images. We use a pre-trained ResNet-50 [15] to pre-process images across different products and options into a 512 dimensional feature vector, which is then transformed into 768 dimensions with a learned linear layer and concatenated to $\text{BERT}(o)$ 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 [21] 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 [51] and referential games [21]. We use the policy gradient method [29] 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):

$$\mathcal{L}_{\text{PG}} = \mathbb{E}_\pi\left[-\left(R_t - V(o_t)\right)\log \pi\left(a_t \mid o_t, \mathcal{A}(o_t)\right)\right] \tag{5}$$

The value $V(o)$ is learned with an L2 loss $\mathcal{L}_{\text{value}} = \left(R_t - V(o_t)\right)^2$. 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 $\mathcal{L}_{\text{RL}} = \mathcal{L}_{\text{PG}} + \mathcal{L}_{\text{value}} + \mathcal{L}_{\text{entropy}}$.

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. More setup details are in the Appendix §C.

5.2 Results

Task performance. From Figure 3, 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

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

products, empirically demonstrating the non-trivial nature of the task. 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 3 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, suggesting more advanced techniques are needed to leverage past information. More ablations in §C.

RL ablations. When we directly train an RL agent (**RL**) from pre-trained BERT parameters, the performance is even worse than the rule baseline. This suggests that IL warm-starting is critical, possibly because of the significant domain shift from traditional language tasks. We also consider a simple RL model with RNN text encoders instead of the Transformer (**RL (RNN)**), which has a success rate more than 10% worse than the IL + RL model with a much larger variance. We hypothesize that RL with a more powerful architecture could help boost and stabilize the performance if the model is initialized with better language and task priors.

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. (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 2 and provide qualitative examples of the trajectories in Table 3.

Human expert vs. agents. Human experts outperform the agents on all score sub-parts (Table 2), 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 3 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 environments, and ideas from query reformulation [34, 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.

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 black-out 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 3: 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.

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 2 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.

5.4 Zero-shot Sim-to-real Transfer

Finally, we also perform 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. (1). 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%. The 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). We provide complete performance and in-depth analysis in Appendix §D.

6 Discussion

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. Our 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 [24, 48], web hypertext [2], or web instruction-action mapping [35] could help agents better understand and leverage rich semantics of webpage content, actions, and instructions. Ideas from query (re)formulation [19, 54, 34, 47] may help agents expand the range of search exploration, and improved action exploration [37, 10, 45] and memory [49, 11, 20] 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.

References

[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] 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.

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

[5] 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.

[6] 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.

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

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

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

[14] 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.

[15] 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.

[16] 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.

[17] 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.

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

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

[20] 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.

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

[22] 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.

[23] 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.

[24] 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.

[25] 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.

[26] 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.

[27] 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.

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

[29] 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.

[30] 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.

[31] 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.

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

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

[34] 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.

[35] 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.

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

[37] 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.

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

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

[40] 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.

[41] 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.

[42] 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.

[43] 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.

[44] 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.

[45] 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.

[46] 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.

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

[48] 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.

[49] 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.

[50] 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.

[51] 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.

[52] 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.

[53] 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.

[54] 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.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See Section 6 Discussion and Appendix.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See Section 6 Discussion and Appendix.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] See supplementary materials.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] Data splits are described in the Section 5 first paragraph. Hyperparameters and training details are in the Appendix.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] Figure 3 includes error bars, Table 2 includes min/max statistics along with averages.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] In appendix training details.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] Citations include ScraperAPI, Flask, OpenAI Gym, BERT, BART, A2C.
   (b) Did you mention the license of the assets? [Yes] Discussed in appendix.
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] In the supplementary materials.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [Yes] Discussed in appendix, we only scrape publicly available data from the Internet.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [Yes] Discussed in Appendix.
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [Yes] In appendix.
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [Yes] Discussed in Appendix.

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [Yes] Discussed in Appendix.

A Environment Details

A.1 Product Scraping

We use ScraperAPI [32] 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 4: Product item statistics.

A.2 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.

A.3 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 4 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.

A.4 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 5, 6). 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.

A.5 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 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

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 5: Actions in WebShop.

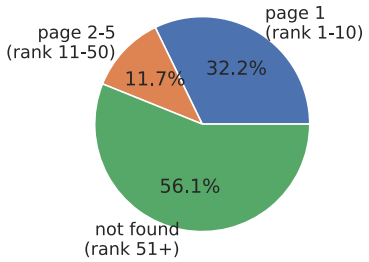


Figure 4: Item rank in search results when the instruction is directly used as search query.

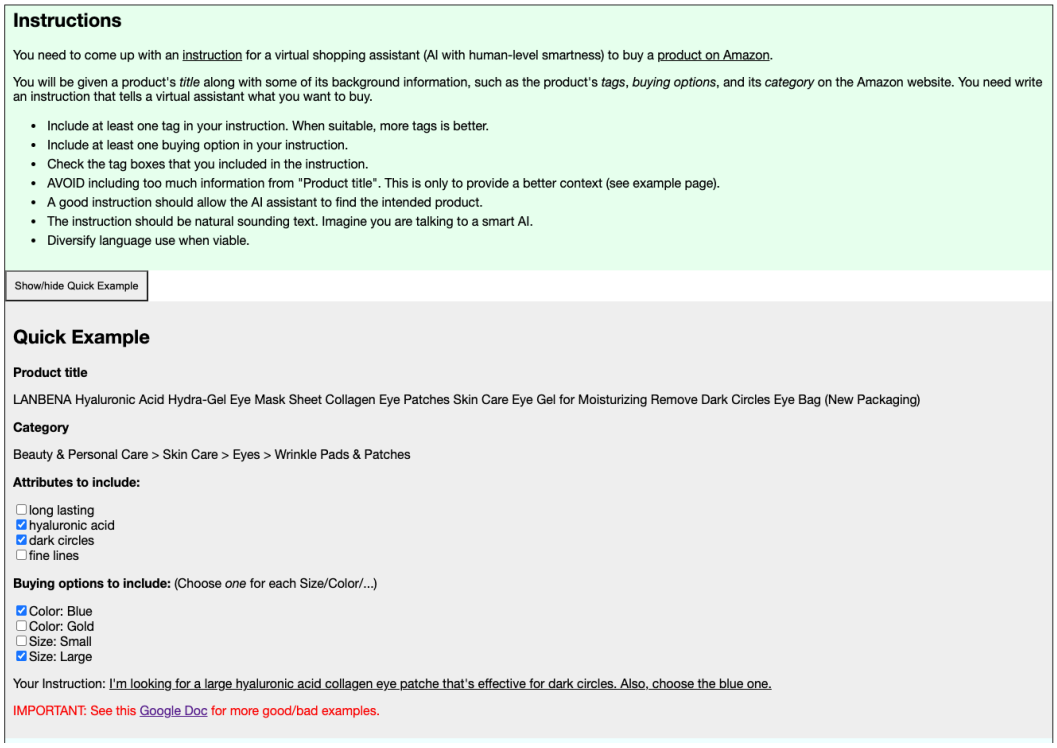


Figure 5: The Amazon Mechanical Turk interface for the instruction writing task. The green box shows the general instruction for the task and the grey box shows an concrete example.

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.

$$r_{\text{type}} = \begin{cases} 0, & \text{if } \texttt{TextMatch}(\bar{y}, \bar{y}^*) = 0 \\ 0.1, & \text{if } \texttt{TextMatch}(\bar{y}, \bar{y}^*) < 0.1 \\ 0.5, & \text{if } \texttt{TextMatch}(\bar{y}, \bar{y}^*) > 0.2 \text{ and } c = 1 \text{ and } f = 1, \\ 1, & \text{otherwise} \end{cases} \tag{6}$$

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.

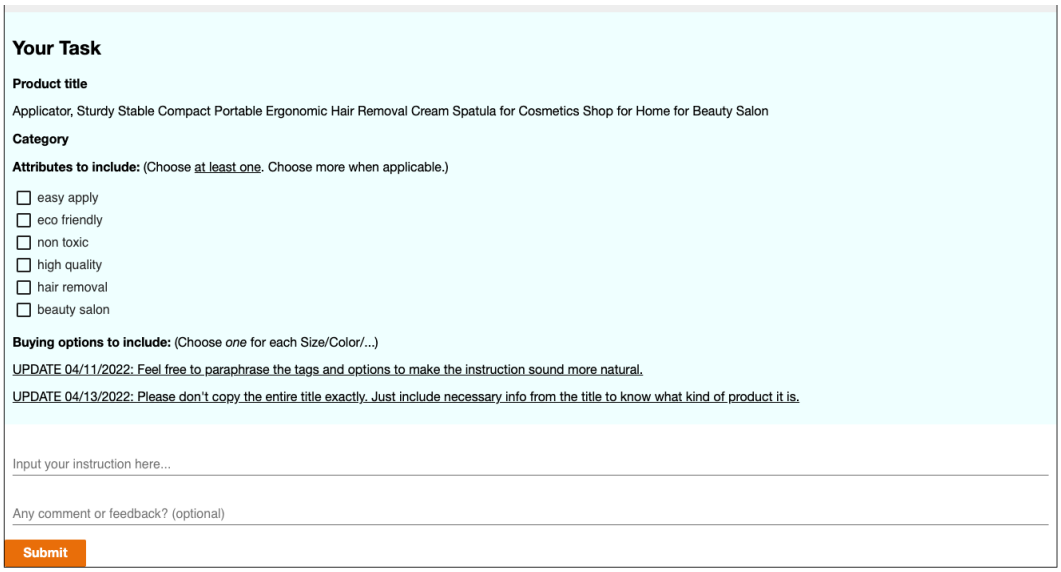


Figure 6: The Amazon Mechanical Turk interface for the instruction writing task. The blue box shows the actual annotation interface. The worker is required to check the boxes and write the instructions in the text field before submission.

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 6: 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.

A.6 Human Trajectory Collection

We use the HTML environment in Figure 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 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 6), cautioning some caveats of the task with human workers.

A.7 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 A.5 – 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: Reward Verification Statistics

Table 7 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.

B Model Details

B.1 Cross Attention Layer

Our cross attention layer follows Seo et al. [39]. 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

$$\alpha_{ij} = \mathbf{w}_1 \cdot \mathbf{o}_i + \mathbf{w}_2 \cdot \mathbf{a}_j + \mathbf{w}_3 \cdot (\mathbf{o}_i \otimes \mathbf{a}_j) \tag{7}$$

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

$$\mathbf{ca}_j = \mathbf{w}_5 \cdot \text{leakyRELU}(\mathbf{w}_4 \cdot [\mathbf{a}_j, \mathbf{c}_j, \mathbf{a}_j \otimes \mathbf{c}_j, \mathbf{q} \otimes \mathbf{c}_j]) \tag{8}$$

$$\mathbf{c}_j = \frac{\sum_i \exp(\alpha_{ij}) \cdot \mathbf{o}_i}{\sum_i \exp(\alpha_{ij})}, \quad \mathbf{q} = \frac{\sum_{j'} \exp(\max_i \alpha_{ij'}) \mathbf{a}_{j'}}{\sum_{j'} \exp(\max_i \alpha_{ij'})} \tag{9}$$

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

$$S(o, a) = \mathbf{w}_6 \cdot \frac{1}{n_a} \sum_{j \leq n_a} \mathbf{ca}_j \in \mathbb{R} \tag{10}$$

where n_a is the number of tokens for action a.

B.2 RNN Baseline

Our RNN baseline is inspired by Guo et al. [12], 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) [7] 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.

C WebShop Experiment Details

C.1 IL Training Details

The training code for our IL models is adapted from Huggingface 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.

C.2 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.

C.3 Results with a `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.[2] 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 2). 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.[3] `Choice` improves the success rate of rule heuristics from

[2] 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.

[3] 74.8% of the time there is only one query in the trajectory.

	Instr. text	IL BART	Human expert (first)	Human expert (last)
Score	79.73	82.97	82.13	**84.36**
SR	52.6%	57.6%	57.9%	**61.0%**

Table 8: 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 9: 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 10: Sampling vs. top-1.

9.6% to 52.6%, and the IL model from 29.1% to 57.6% (Table 8), 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.

C.4 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 10. 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.

C.5 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 (11). Removing image 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.

D Sim-to-real Details

We show the complete sim2real transfer performance in Table 9.

D.1 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 $\rightarrow$ Amazon HTML $\rightarrow$ Amazon Page Information: Using ScraperAPI [32], 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).

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

Table 11: Image ablations.

- 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.

D.2 Sim-to-real Transfer Results

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. (1). As shown in Table 9, model performances on the two website are similar to WebShop performances in Figure 3, except for the rule baseline which doubles its success rate in Amazon, likely due to the better search engine of Amazon than WebShop. For both websites, our IL and IL+RL agents still outperform the rule baseline considerably, confirming positive sim-to-real values of trained agents for real-world web tasks despite domain shifts in data (products) and dynamics (search engine)[4]. 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 (from accepting the task to completing it) 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, especially if we are able to solve the aforementioned challenges to improve performance.

	Amazon					eBay				
	All	Att	Opt	Type	Price	All	Att	Opt	Type	Price
Rule	45.8	45.6	38.0	66.2	90.0	31.7	62.3	25.9	49.0	67.0
IL	61.9	60.7	53.7	85.6	96.0	58.2	60.2	52.3	85.1	96.9
IL+RL	65.9	71.6	47.0	87.8	100.0	62.3	69.1	39.5	91.7	97.0
Human	88.2	86.2	76.3	99.0	100.0	79.7	80.3	70.1	99.5	100.0

Table 12: Sim-to-real Transfer Statistics.

The resulting numbers in Table 12 closely cohere to the reported numbers of WebShop found in Figure 3, suggesting that the WebShop has promise for developing grounded agents that can operate on real web environments. The following Table 13 is an example of a trajectory generated by the IL agent searching on the real Amazon website.

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. With that said, these results also encourage

[4]Between 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) eBay search engine seems weaker, and would sometimes display no results for lengthy instructions.

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 13: 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.

us to look into expanding on the current limitations in our work regarding both the model and the WebShop environment.

E 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 labeller'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 [30] 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.

Instruction for Human Trajectory Collection

The following pages display the human trajectory collection document mentioned in §A.6.

The WebShop Task

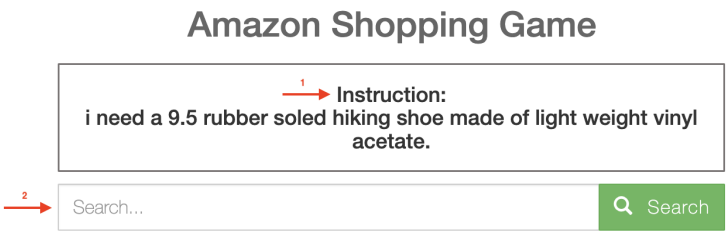
Thank you for taking part in this project! In this task, you need to buy a designated product given an instruction on our Amazon Shopping Game site. You will get a score in the end indicating how close you are. Please try to score as high as you can.

<u>If you find in some cases the scoring seems weird/unfair, please reach out. We will look into the cases.</u>

Please read the following instructions carefully before you start.

Instructions

1) Go to the home page. The instruction will immediately show up on the landing page.



2) Given this instruction, please write a search query that would produce search results matching the description.

Please *do not* copy-paste the entire instruction. We encourage you come up with more targeted queries, see the result, and search again if needed.
Example:

- `Instruction`:
  `I need a 9.5 rubber soled hiking shoe made of lightweight vinyl acetate.`

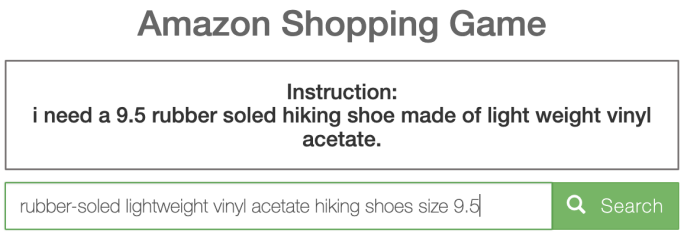
- `Bad query`: (copy pasting)
  `9.5 rubber soled hiking shoe made of lightweight vinyl acetate`

```
- Ideal query: (1st attempt)
  rubber soled hiking shoe vinyl acetate (say the results are not great)
- Ideal query: (2nd attempt)
  hiking shoe lightweight vinyl acetate  (the results are better)
- Ideal query: (3nd attempt)
  lightweight climbing shoe vinyl        (gives promising results)
```

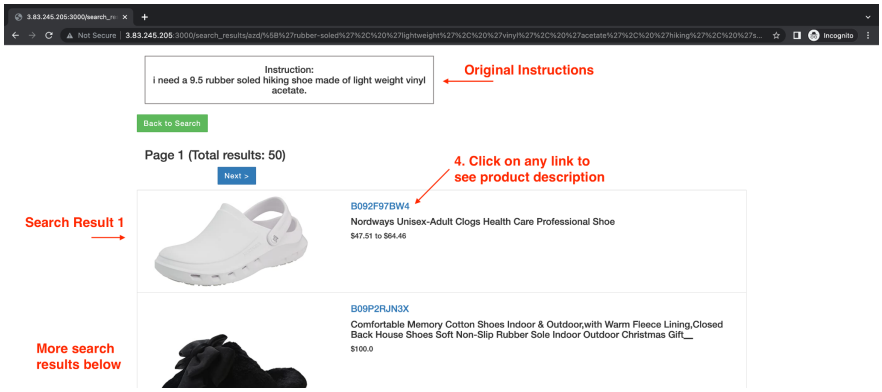
Essentially, you need to hack the search engine a little bit.

Note that our search engine is limited. Tricks that work on Google Search such as adding quotation marks around the query won't work.

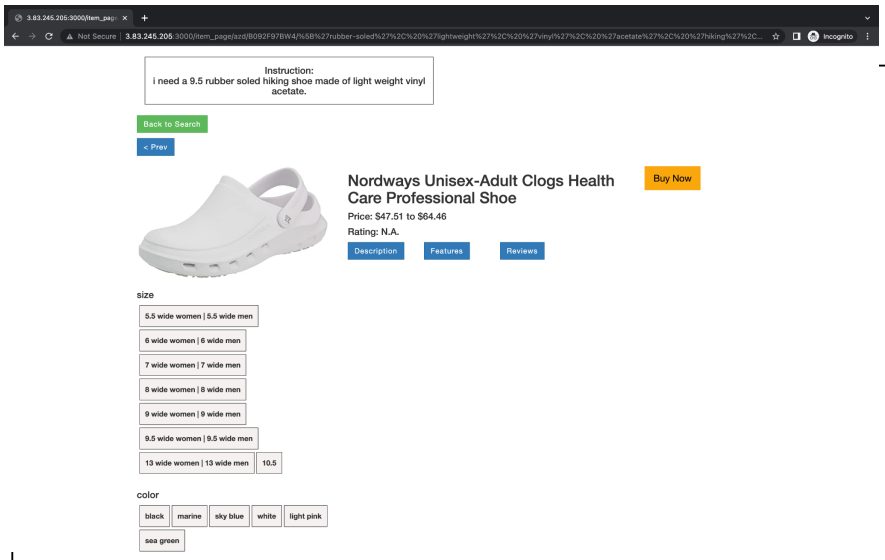
Click **Search** after filling out the search bar like below.



3) Upon clicking **Search**, you will be sent to a page of results. The below screenshot is an example of the results displayed from the example query in Step 2. Each page shows up to 10 results. Click the **Next** button to see more results.



4) Click on any of the blue product title text (i.e. *"B092F97B24"* in above screenshot) to see a product detail page, like the below.



Guidelines for searching for matching products:

- Some pages have **Options** (i.e. *Size*, *Color* in above screenshot). If the instructions contain such information, please select the corresponding options (even if the title / features / desc. / reviews may already contain such info). In most cases, if you find the options verbatim as in the instruction, you've likely found the right product.
- Do **not** use the product image to determine whether the instruction's information matches the product.

  An example:
  - Given instruction: "Find me a pair of ankle socks that are blue and size 11"

Between this product…

TITLE: Ankle socks for casual wear, sports, and leisure. Pack of 4, 8, or 12

DESC: 100% made in the USA. These socks are good for any occasion.
FEATURES: Made with cotton, breathable fabric. Machine washable okay.

OPTIONS:
- Sizes: 8, 9, 10, 11, 12
- Color: red, green, black, white, blue

And this product…

TITLE: Kirkland athletic socks with rubber soles and heels. Easy slip on

DESC: Costco wholesale socks, limited stock.
FEATURES: Polyester and Rayon fabric. Guaranteed long lasting or your money back.

OPTIONS: None

The left hand is a **better match** because the product's title, features, description, and options reflect the instruction's information.

While the right hand product appears to be a pair of blue ankle socks, because this information is not reflected in the text, we do **not** consider this a match.

Therefore, feel free to use the product image as a reference when looking for matches, but keep in mind that the experiment we're running accounts for a text's

5) Decide whether the product is a match

A **match** should
- Contain all of the instruction's information in the product detail page's text (i.e. title, description, feature, options)
- Have options (if they exist), which correspond to the product info, be selected.

A match does **not** account for
- The product image

- *You think it is a match!* → Click the Buy Now button on the product detail page
- *You think it is not a match OR another product might be a better match…*
  - Click on the Back button to go to the original list of search results (page **3**). From here, repeat steps **3-4** until you find a product that matches best.
  - Click on the **Back to Search** button. This will take you back to the search bar page (page **2**). If you feel none of the results are good matches, try another search query.

6) Once you clicked Buy Now, you will see your score (won't be used to decide the pay), and a code you need to paste in the MTurk interface. And you're done!

Tips

Patterns that often result in HIGH scores:
- Refine search queries until promising products show up
- Explore different product pages (go to next page if needed) to see if options and different aspects are covered
- Make sure all aspects in the instructions are covered by either the title, description page, or the feature page.
- Make sure all options are found almost verbatim in the product page

Patterns that often result in LOW scores:

- Low effort copy-paste the entire instructions as the search query
- Always click the first item without checking if the aspects in the instructions are covered
- Click items that obviously don’t have any option matches