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

Anonymous Author(s)
Affiliation
Address
email

Abstract

Most existing benchmarks for grounding language in interactive environments either lack realistic linguistic elements, or prove difficult to scale up due to substantial human involvement in the collection of data or feedback signals. We develop WebShop – a simulated e-commerce website environment with 1.18 million real-world products and $12,087$ crowd-sourced text instructions. In this environment, an agent needs to navigate multiple types of webpages and issue diverse actions to find, customize, and purchase a product given an instruction. WebShop provides several challenges including understanding compositional instructions, query (re-)formulation, dealing with noisy text in webpages, and performing strategic exploration. We collect over $1,600$ human trajectories to first validate the benchmark, then 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 significantly outperforms rule heuristics but is far lower than expert human 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 our agent trained on WebShop exhibits non-trivial *sim-to-real* transfer when evaluated on `amazon.com`, indicating the potential value of our benchmark for developing practical web 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 [20, 35, 39]. On the other hand, large-scale language models like GPT-3 [5] and BERT [8] 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 datasets that lack adequate grounding to extra-linguistic concepts [3]. In order to make further progress in building **grounded** language models, there is a need for 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 scalable interactive learning, without the need for expensive feedback from humans every time.

The world wide web (WWW) is a massive open-domain interactive environment that inherently satisfies the above requirements, through its interconnected set of pages with natural text, images

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

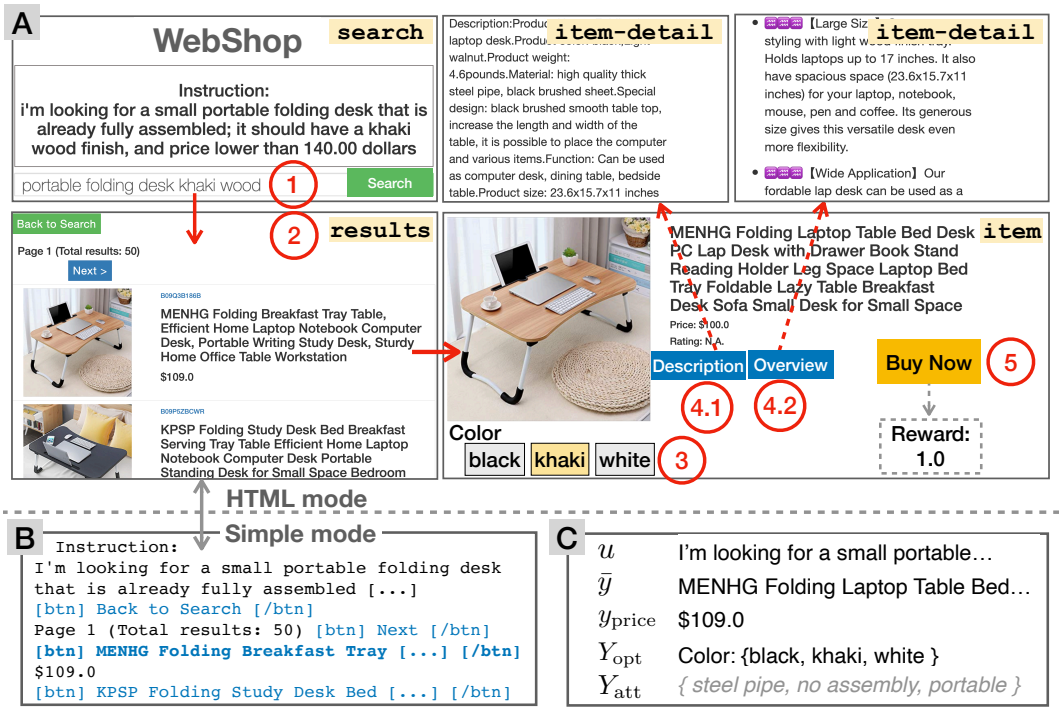

Figure 1: The WebShop environment. **A**: An example task trajectory in `HTML` mode, where a user can (1) search a query in a `search` page, (2) click a product item in a `results` page, (3) choose a color option in a `item` page, (4) check `item-detail` pages and go back to the item page, and (5) finally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

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 fall short in one of two ways. Some benchmarks only contain either a single classification task [28, 32, 21] or interactions containing only a handful of different pages in each episode [30]. Others propose tasks with longer horizons but are either limited to following hyperlinks for web navigation [26] or require human-in-the-loop feedback due to the lack of an automated reward function [23].

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. WebShop emulates the task of online shopping on an e-commerce website, where the agent's goal is to understand a human instruction and *purchase* the best product that matches 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 have to backtrack, view, and compare various products, then potentially perform multiple searches if the agent deems no current search result a good match. The benchmark contains over one million real products scraped from `amazon.com`, $12,087$ crowdsourced instructions, and a diverse semantic action space – all packaged into a convenient OpenAI Gym [4] environment for training and evaluating RL agents. The environment can be rendered in two modes – `HTML` or `simple` – with different observation spaces, and 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 easily scale up the benchmark in the future.

WebShop is a challenging environment for autonomous agents due to several reasons: (1) the average branching factor (around 15 for non-search pages) is very large, making exploration hard, (2) the agent has to understand real-world text and images on webpages and generate natural language for search queries, (3) the textual instructions require compositional generalization, and (4) the agent has to remember previous visited states and search queries in order to compare items before making its choice. This makes the benchmark an ideal testbed for grounded language and visual understanding in a realistic setting (as opposed to games) while being fast enough to simulate and easily scale to more domains and tasks (compared to simulated 3D navigation or physical robotic manipulation).

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 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 that can achieve 82.1 and 59.6% on this task. 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 our agent trained on WebShop to operate on the real `amazon.com`, and find that it is able to achieve a similar success rate (27%). This demonstrates the practical potential of our work towards enabling agents that can operate autonomously on the real world wide web (WWW). Our code is available in the supplementary material.

2 Related Work

Reinforcement learning on the web. Nogueira and Cho [26] 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 [30] 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 [19], curriculum and meta-learning [9], DOM tree representation [14], adversarial environment generation [10] and large-scale behavioral cloning [13]. 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 [34] 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 [28], generating API calls [32, 33, 36] and semantic parsing into concept-level navigation actions [21]. Perhaps most similar content-wise to our work is the Klarna product page dataset [12] 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 [24] and retrieval [1], question answering [38, 16] and dialog [31], and training language models on webtext [2]. These approaches use search engines primarily as a knowledge

retriever for gathering additional evidence for the task at hand. Perhaps most similar to our work is WebGPT [23], 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 is 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 $\mathcal{O}$.

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 clicking a 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 [30]. 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).

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 of $\texttt{TextMatch}(\cdot)$ is in the Appendix. We report both a *task score* (100% scaled reward) and *success rate* (SR) – the ratio of full reward episodes – as our task metrics. Note that it is possible to get a full score even if the final product is not y^*.

Observation. Using Flask [29] 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 § 4.1 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. [30]), 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.

3.2 Implementation Details

Data scraping. We use ScraperAPI [25] 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. More details about product scraping is in the Appendix.

Search engine. We use Pyserini [18] for the search engine, where indices are built offline using a BM25 sparse retriever model with text concatenated from the product title, description, overview, and customization options. While the search engine is deterministic in this work, which in turn eases model training and imitation learning, future work could consider a stochastic version to provide a greater challenge for models in terms of generalization and robustness (e.g. by adding "promoted items" at the top of results, or effecting slight changes in item ranking).

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.

Natural language instruction. 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.

3.3 Verification with human studies

Human performance. We collect trajectories from humans performing the task in `HTML` mode of the environment to understand the task difficulty for humans and analyze how humans solve the task. Humans achieve an overall score of 75.5 and a success rate of 50.0%. We use qualification tests to train and select motivated workers to perform the task. We recruit a total of 13 workers to perform the task, and among them we also report the "expert" performance of the top 7 workers (see Appendix). The experts achieve an overall score of 82.1 and a success rate of 59.6%. We also make use of this data to train some of our models using imitation learning (§ 4.1). These trajectories are further analyzed in § 5.2, which reveals research challenges in WebShop such as (1) generating good search queries, (2) strategic exploration for navigating through the website, (3) robust language

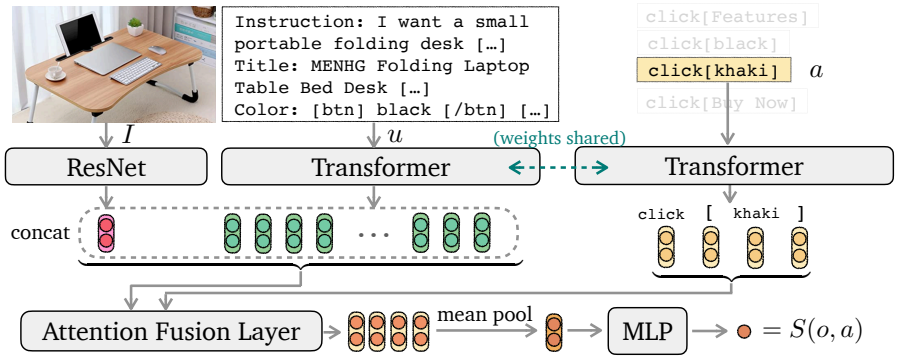

Figure 2: Architecture of our choice-based imitation learning (IL) model. The image I is passed to a ResNet to obtain the image representation. The instruction text u is passed to a transformer (initialized with BERT) to obtain the text representations. The concatenated bi-modal representations are fused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

understanding for reading product descriptions and overviews, then choosing the right options, and (4) long-term memory for comparing items (Figure 1).

Faithfulness of reward function. We take 100 samples each of trajectories performed by both average and expert humans and manually recalculate the automatic rewards for the selected product according to a human grading reward metric; the purpose is to determine whether the reward function mimics a human's judgment. For the sample of average trajectories, the overall score was 74.9 and the human based evaluation was 76.3 with a Pearson correlation of 0.856. For expert trajectories, respective scores were 81.5 and 89.9 with a Pearson correlation of 0.773. The source of error is mainly due to lexically similar or synonymous words that did not get points due to the exact matching criteria of the reward function.

4 Methods

We propose models that combine language and image pre-training with imitation learning (IL) and reinforcement learning (RL). As a lower bound for all other models, a simple **rule baseline** is to search the exact instruction text, then click and buy the first item in the results page without choosing any options. The heavy lifting of the lexical search engine would lead to a non-trivial attribute reward, but 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.

4.1 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 [17] 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 [7] 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 $S(o, a)$:

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

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

Handling Images. We use a pre-trained ResNet-50 [11] 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).

4.2 Reinforcement Learning (RL)

We also fine-tune the choice-based IL model with online RL. Prior work suggests that directly fine-tuning text generation via RL might lead to language drifting [15] 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 [37] and referential games [15]. We use the policy gradient method [22] 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): $\mathcal{L}_{\text{PG}} = \mathbb{E}_\pi\left[-\left(R_t - V(o_t)\right)\log \pi\left(a_t \mid o_t, \mathcal{A}(o_t)\right)\right]$. 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

Setup. 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 reinforcement learning (RL). We randomly sample a subset of the $10,587$ training instructions, then collect $1,012$ human demonstrations for imitation learning (IL) and a further 54 demonstrations from instances in the development set for IL hyper-parameter 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. We use two evaluation metrics: (1) **Score** defined as $r \times 100$, which captures the total reward obtained; and (2) **Success Rate (SR)** defined as the portion of instructions where $r = 1$. More setup details are in the Appendix.

5.1 Results

Main results. From Figure 3, we observe that the rule-based model 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. 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 §5.2). 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%). There is also a sizable gap between average and expert human performance, which indicates the task could be non-trivial even for humans.

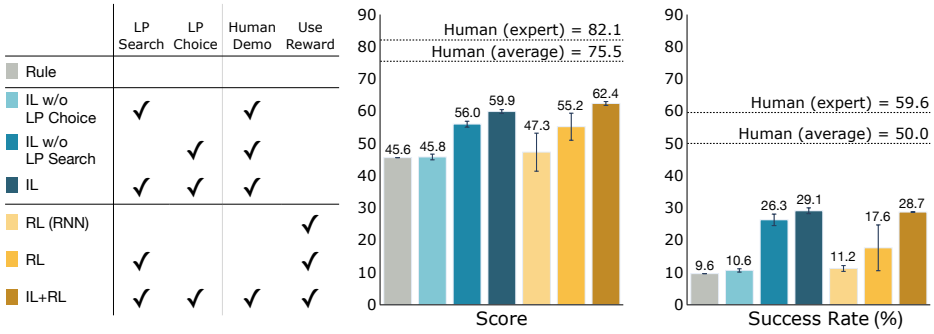

Figure 3: Task scores and Success Rate (%) for our models over 3 trials. LP Search uses a pre-trained BART model to generate the search query and IL w/o LP Search uses the rule-based heuristic. LP Choice uses pre-trained BERT weights to initialize the choice action model and IL w/o LP Choice trains a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

	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.

	Score	SR
Rule	45.8	0.19
IL	**61.5**	**0.27**

Table 3: Sim-to-real transfer experiments.

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)**), 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.

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.

5.2 Analysis

To better understand the differences between the agents and human experts, we 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 also report trajectory statistics such as the average number of states, unique items visited, and number of searches per episode (Table 2). We provide qualitative examples of the trajectories in Table 4.

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 4 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

Instruction 1: 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..	**Instruction 2** 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] click[item-calyx..] click[back to search] search[66 x 66 blackout shades] click[item-milin..] click[66"w x 66"h] click[cordless bottom up-blackout-white] click[buy]	**Human Actions** (r = 1.0, len = 17) search[gingko light 20"x20" pillow cover hand painted] click[item-maison..] ... click[description] click[<prev] click[overview] click[<prev] ... click[item-maison..] click[20"x20"] click[nudes (gingko light)] click[buy]
IL+RL Actions (r = 0.2, length = 3) search[white blackout shades 65 inches in width and 66 inches in height] click[item-window..] click[buy]	**IL+RL Actions** (r = 0.25, length = 3) search[gingko light and 20x20 pillow cover hand painted] click[item-upoos..] click[buy]

Table 4: Two example trajectories (showing only actions) from the human and the IL+RL model. We omit instructions and some human actions for instruction and trim item names for readability. Red denotes options and blue denotes attributes.

models since it involves reasoning and adapting to grounded environments, and ideas from query reformulation [27, 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 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.

5.3 Sim-to-real Transfer

Finally, we explore a '*sim-to-real*' transfer scenario where our IL model trained on WebShop is tested on the real Amazon website (`amazon.com`). We use 100 test instructions and employ the rule baseline and IL model to interact with Amazon, and manually score each episode based on Eq. (1). As shown in Table 3, the IL model achieves an average score of 61.5, whereas directly typing instruction text into Amazon search engine (which is much stronger than the WebShop search engine) and selecting the first item leads to a much lower score of 45.8. The performance also closely match our WebShop reported numbers in Figure 3. The transfer only requires two minor coding additions, suggesting that our environment is suitable for grounded agent development that to enable sim-to-real transfer. More details and example trajectories are provided in the Appendix.

6 Discussion

We introduced 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 several agents trained using imitation and reinforcement learning, identified several challenges for future model development and also demonstrated promising results on transfer to real websites. The modular design of WebShop allows for new 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. There are some limitations to our work – our data is restricted to products from one shopping website (Amazon), contains only English text and our current agents lack long-term memory and robust reasoning capabilities. And despite being a simulated environment, WebShop can enable training of agents that operate on the real web, which raises issues around the safety and human control of RL agents.

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

[8] 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.

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

[10] 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.

[11] 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.

[12] 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.

[13] 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.

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

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

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] 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.

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

[22] 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.

[23] 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.

[24] 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.

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

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

[27] 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.

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

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

[30] 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.

[31] 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.

[32] 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.

[33] 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.

[34] 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.

[35] 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.

[36] 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.

[37] 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.

[38] 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.

[39] 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.

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.