United States Patent Application Publication	20250265478
Kind Code	A1
Publication Date	August 21, 2025
Inventor(s)	Partow; Rustin et al.

EPISODIC OFF-POLICY EVALUATION OF JOINT ACTIONS FOR A MACHINE-LEARNED POLICY FOR AN ONLINE SYSTEM

Abstract

An off-policy evaluation system performs episodic off-policy evaluations to perform off-policy evaluation (OPE) for multiple, joint episodes. For a single episode, a first machine learning model outputs a propensity for each action for the user and selects a first action for the user from the set of propensities. For a second episode, a second machine learning model outputs a propensity for each action for the user and selects a first action for the user from the set of propensities. The second machine learning model is evaluated by determining an importance weight for the first model and the second model to determine the inverse propensity score of the second machine learning model.

Inventors: **Partow; Rustin (San Francisco, CA), Sweeney; Mackenzie Patrick (Haymarket, VA), Huang; Tianyue (San Francisco, CA), Levinson; Trace (Brooklyn, NY)**

Applicant: **Maplebear Inc.** (San Francisco, CA)

Family ID: **1000007710909**

Appl. No.: **18/582460**

Filed: **February 20, 2024**

Publication Classification

Int. Cl.: **G06N5/022** (20230101); **G06N20/00** (20190101)

U.S. Cl.:

CPC **G06N5/022** (20130101); **G06N20/00** (20190101);

Background/Summary

BACKGROUND
[0001] An online system evaluates a machine-learned policy model for assigning an action from a set of actions to a user. Current online systems utilize machine-learned contextual policies to predict what actions should be assigned to a user such that, for example, the user would engage with the online system the most. An off-policy evaluation method evaluates how a new policy would perform using old experimental data that is generated by a logging policy. However, current off-policy evaluations estimate the effect of a treatment or policy for a single observational episode for the user. However, in many instances, users are assigned a sequence of actions across multiple episodes, and the joint effects of multiple episodes are not accounted for when performing a single-episode off-policy evaluation.

Description

BRIEF DESCRIPTION OF THE DRAWINGS
[0002] FIG. **1**A illustrates an example system environment for an online concierge system, in accordance with one or more embodiments.
[0003] FIG. **1**B illustrates an example system environment for an online concierge system, in accordance with one or more embodiments.
[0004] FIG. **2** illustrates an example system architecture for an online concierge system, in accordance with one or more embodiments.
[0005] FIG. **3** illustrates an example timeline wherein an episodic off policy is used to evaluate the evaluation policy.
[0006] FIG. **4** is a flowchart for a policy evaluation module, in accordance with one or more embodiments.
DETAILED DESCRIPTION
[0007] FIG. **1**A illustrates an example system environment for an online concierge system **140**, in accordance with one or more embodiments. The system environment illustrated in FIG. **1**A includes a customer client device **100**, a picker client device **110**, a retailer computing system **120**, a network **130**, and an online concierge system **140**. Alternative embodiments may include more, fewer, or different components from those illustrated in FIG. **1**A, and the functionality of each component may be divided between the components differently from the description below. Additionally, each component may perform their respective functionalities in response to a request from a human, or automatically without human intervention.
[0008] As used herein, customers, pickers, and retailers may be generically referred to as “users” of the online concierge system **140**. Additionally, while one customer client device **100**, picker client device **110**, and retailer computing system **120** are illustrated in FIG. **1**, any number of customers, pickers, and retailers may interact with the online concierge system **140**. As such, there may be more than one customer client device **100**, picker client device **110**, or retailer computing system **120**.
[0009] The customer client device **100** is a client device through which a customer may interact with the picker client device **110**, the retailer computing system **120**, or the online concierge system **140**. The customer client device **100** can be a personal or mobile computing device, such as a smartphone, a tablet, a laptop computer, or desktop computer. In some embodiments, the customer client device **100** executes a client application that uses an application programming interface (API) to communicate with the online concierge system **140**.
[0010] A customer uses the customer client device **100** to place an order with the online concierge system **140**. An order specifies a set of items to be delivered to the customer. An “item”, as used herein, means a good or product that can be provided to the customer through the online concierge system **140**. The order may include item identifiers (e.g., a stock keeping unit or a price look-up

code) for items to be delivered to the user and may include quantities of the items to be delivered. Additionally, an order may further include a delivery location to which the ordered items are to be delivered and a timeframe during which the items should be delivered. In some embodiments, the order also specifies one or more retailers from which the ordered items should be collected.
[0011] The customer client device **100** presents an ordering interface to the customer. The ordering interface is a user interface that the customer can use to place an order with the online concierge system **140**. The ordering interface may be part of a client application operating on the customer client device **100**. The ordering interface allows the customer to search for items that are available through the online concierge system **140** and the customer can select which items to add to a "shopping list." A "shopping list," as used herein, is a tentative set of items that the user has selected for an order but that has not yet been finalized for an order. The ordering interface allows a customer to update the shopping list, e.g., by changing the quantity of items, adding or removing items, or adding instructions for items that specify how the item should be collected.
[0012] The customer client device **100** may receive additional content from the online concierge system **140** to present to a customer. For example, the customer client device **100** may receive coupons, recipes, or item suggestions. The customer client device **100** may present the received additional content to the customer as the customer uses the customer client device **100** to place an order (e.g., as part of the ordering interface).
[0013] Additionally, the customer client device **100** includes a communication interface that allows the customer to communicate with a picker that is servicing the customer's order. This communication interface allows the user to input a text-based message to transmit to the picker client device **110** via the network **130**. The picker client device **110** receives the message from the customer client device **100** and presents the message to the picker. The picker client device **110** also includes a communication interface that allows the picker to communicate with the customer. The picker client device **110** transmits a message provided by the picker to the customer client device **100** via the network **130**. In some embodiments, messages sent between the customer client device **100** and the picker client device **110** are transmitted through the online concierge system **140**. In addition to text messages, the communication interfaces of the customer client device **100** and the picker client device **110** may allow the customer and the picker to communicate through audio or video communications, such as a phone call, a voice-over-IP call, or a video call.
[0014] The picker client device **110** is a client device through which a picker may interact with the customer client device **100**, the retailer computing system **120**, or the online concierge system **140**. The picker client device **110** can be a personal or mobile computing device, such as a smartphone, a tablet, a laptop computer, or desktop computer. In some embodiments, the picker client device **110** executes a client application that uses an application programming interface (API) to communicate with the online concierge system **140**.
[0015] The picker client device **110** receives orders from the online concierge system **140** for the picker to service. A picker services an order by collecting the items listed in the order from a retailer. The picker client device **110** presents the items that are included in the customer's order to the picker in a collection interface. The collection interface is a user interface that provides information to the picker on which items to collect for a customer's order and the quantities of the items. In some embodiments, the collection interface provides multiple orders from multiple customers for the picker to service at the same time from the same retailer location. The collection interface further presents instructions that the customer may have included related to the collection of items in the order. Additionally, the collection interface may present a location of each item in the retailer location, and may even specify a sequence in which the picker should collect the items for improved efficiency in collecting items. In some embodiments, the picker client device **110** transmits to the online concierge system **140** or the customer client device **100** which items the picker has collected in real time as the picker collects the items.
[0016] The picker can use the picker client device **110** to keep track of the items that the picker has

collected to ensure that the picker collects all of the items for an order. The picker client device **110** may include a barcode scanner that can determine an item identifier encoded in a barcode coupled to an item. The picker client device **110** compares this item identifier to items in the order that the picker is servicing, and if the item identifier corresponds to an item in the order, the picker client device **110** identifies the item as collected. In some embodiments, rather than or in addition to using a barcode scanner, the picker client device **110** captures one or more images of the item and determines the item identifier for the item based on the images. The picker client device **110** may determine the item identifier directly or by transmitting the images to the online concierge system **140**. Furthermore, the picker client device **110** determines a weight for items that are priced by weight. The picker client device **110** may prompt the picker to manually input the weight of an item or may communicate with a weighing system in the retailer location to receive the weight of an item.

[0017] When the picker has collected all of the items for an order, the picker client device **110** instructs a picker on where to deliver the items for a customer's order. For example, the picker client device **110** displays a delivery location from the order to the picker. The picker client device **110** also provides navigation instructions for the picker to travel from the retailer location to the delivery location. Where a picker is servicing more than one order, the picker client device **110** identifies which items should be delivered to which delivery location. The picker client device **110** may provide navigation instructions from the retailer location to each of the delivery locations. The picker client device **110** may receive one or more delivery locations from the online concierge system **140** and may provide the delivery locations to the picker so that the picker can deliver the corresponding one or more orders to those locations. The picker client device **110** may also provide navigation instructions for the picker from the retailer location from which the picker collected the items to the one or more delivery locations.

[0018] In some embodiments, the picker client device **110** tracks the location of the picker as the picker delivers orders to delivery locations. The picker client device **110** collects location data and transmits the location data to the online concierge system **140**. The online concierge system **140** may transmit the location data to the customer client device **100** for display to the customer such that the customer can keep track of when their order will be delivered. Additionally, the online concierge system **140** may generate updated navigation instructions for the picker based on the picker's location. For example, if the picker takes a wrong turn while traveling to a delivery location, the online concierge system **140** determines the picker's updated location based on location data from the picker client device **110** and generates updated navigation instructions for the picker based on the updated location.

[0019] In one or more embodiments, the picker is a single person who collects items for an order from a retailer location and delivers the order to the delivery location for the order. Alternatively, more than one person may serve the role as a picker for an order. For example, multiple people may collect the items at the retailer location for a single order. Similarly, the person who delivers an order to its delivery location may be different from the person or people who collected the items from the retailer location. In these embodiments, each person may have a picker client device **110** that they can use to interact with the online concierge system **140**.

[0020] Additionally, while the description herein may primarily refer to pickers as humans, in some embodiments, some or all of the steps taken by the picker may be automated. For example, a semi- or fully-autonomous robot may collect items in a retailer location for an order and an autonomous vehicle may deliver an order to a customer from a retailer location.

[0021] The retailer computing system **120** is a computing system operated by a retailer that interacts with the online concierge system **140**. As used herein, a “retailer” is an entity that operates a “retailer location,” which is a store, warehouse, or other building from which a picker can collect items. The retailer computing system **120** stores and provides item data to the online concierge system **140** and may regularly update the online concierge system **140** with updated item data. For

example, the retailer computing system **120** provides item data indicating which items are available at a retailer location and the quantities of those items. Additionally, the retailer computing system **120** may transmit updated item data to the online concierge system **140** when an item is no longer available at the retailer location. Additionally, the retailer computing system **120** may provide the online concierge system **140** with updated item prices, sales, or availabilities. Additionally, the retailer computing system **120** may receive payment information from the online concierge system **140** for orders serviced by the online concierge system **140**. Alternatively, the retailer computing system **120** may provide payment to the online concierge system **140** for some portion of the overall cost of a user's order (e.g., as a commission).
[0022] The customer client device **100**, the picker client device **110**, the retailer computing system **120**, and the online concierge system **140** can communicate with each other via the network **130**. The network **130** is a collection of computing devices that communicate via wired or wireless connections. The network **130** may include one or more local area networks (LANs) or one or more wide area networks (WANs). The network **130**, as referred to herein, is an inclusive term that may refer to any or all of standard layers used to describe a physical or virtual network, such as the physical layer, the data link layer, the network layer, the transport layer, the session layer, the presentation layer, and the application layer. The network **130** may include physical media for communicating data from one computing device to another computing device, such as MPLS lines, fiber optic cables, cellular connections (e.g., 3G, 4G, or 5G spectra), or satellites. The network **130** also may use networking protocols, such as TCP/IP, HTTP, SSH, SMS, or FTP, to transmit data between computing devices. In some embodiments, the network **130** may include Bluetooth or near-field communication (NFC) technologies or protocols for local communications between computing devices. The network **130** may transmit encrypted or unencrypted data.
[0023] The online concierge system **140** is an online system by which customers can order items to be provided to them by a picker from a retailer. The online concierge system **140** receives orders from a customer client device **100** through the network **130**. The online concierge system **140** selects a picker to service the customer's order and transmits the order to a picker client device **110** associated with the picker. The picker collects the ordered items from a retailer location and delivers the ordered items to the customer. The online concierge system **140** may charge a customer for the order and provides portions of the payment from the customer to the picker and the retailer.
[0024] As an example, the online concierge system **140** may allow a customer to order groceries from a grocery store retailer. The customer's order may specify which groceries they want delivered from the grocery store and the quantities of each of the groceries. The customer's client device **100** transmits the customer's order to the online concierge system **140** and the online concierge system **140** selects a picker to travel to the grocery store retailer location to collect the groceries ordered by the customer. Once the picker has collected the groceries ordered by the customer, the picker delivers the groceries to a location transmitted to the picker client device **110** by the online concierge system **140**.
[0025] An online concierge system **140** performs episodic off-policy evaluations to perform off-policy evaluation (OPE) for multiple, joint episodes. Typically, a contextual bandit experimentation assigns an action to a user among one or more actions based on a contextual machine-learning model. For a single episode, a user may interact with an application of the online concierge system **140**, and the ML model outputs a propensity for each action that is a likelihood the action should be assigned to the user to elicit a desired response from the user (e.g., conversion on products, clicking on a content item, etc.). For example, an interaction may be a user opening and logging into a mobile application of an online concierge system **140** to order grocery items. A contextual policy is a set of decision rules prescribing a propensity for taking an action based on context. For example, a contextual policy for a user may assign propensities, or likelihoods, that the user selects a respective action based on the user's context.
[0026] An action for a user is an action performed by the online concierge system **140** for the

experiment. In one or more embodiments, a set of actions presented to a user often refer to a set of commands, controls, content items, or options presented to a user through a user interface. For example, a set of actions presented to the user is a set of digital coupons presented to the digital platform of the user. A set of actions in an action space may be no coupon presented to the user, a 10% off coupon, and a 20% off coupon to the user.
[0027] Often times, developers develop new contextual machine-learned policy models, but it is expensive and difficult to deploy the contextual bandit for the new model in a new experiment, because it is time-intensive and potentially financially expensive to do so. Moreover, even if a new policy is deployed in an experiment, untested and ineffective contextual policies would lead to a loss in expected profits and may deter users from increased engagement into the online concierge system **140**. Therefore, it is important to evaluate the performance of a new machine-learned policy offline before the policy is deployed to users.
[0028] An inverse propensity score (IPS) is an offline policy evaluation method where the ratio of propensities between the contextual policy to be simulated and a logged contextual policy that was deployed in an experiment are determined as an importance weight to compute a weighted average reward for the new machine-learned policy. Usually, some inverse propensity score is calculated offline to evaluate new policies based on the expected impacts of the contextual policy to be evaluated (hereinafter "evaluation policy"). By evaluating a policy offline, the performance of the evaluation policy is accurately simulated on old experimental data instead of running on a new experiment that would be costly and time inefficient.
[0029] Current evaluation methods measure contextual policies in the context of a single instance or episode of a user's interaction. As a result, current evaluation methods do not provide comprehensive insight into the user's interactions with the online concierge system **140** when they happen over multiple episodes over time. This loses information about the behavior of the user that may be reflected across the multiple episodes of the user. For example, some users, colloquially named a "high-fatigue group," do not respond favorably to a sequence in which they are shown both a coupon in the first instance and a coupon in the second instance, but will respond favorably if they are shown only a coupon in the second instance with a significantly higher average rate of reward (e.g., conversion rate, expected profit). As a result, evaluating a contextual policy over a single episode, for example only the first episode, to estimate how the user group would respond to the action does not accurately portray how the user would respond to a particular sequence of actions in both instances.
[0030] Therefore, in one or more embodiments, the online concierge system **140** described herein performs OPE for multiple episodes by dividing up time interval into regular intervals, defining a null action for a time that user did not interact with the application, and applying a multi-episode OPE framework. The online concierge system **140** divides up the time interval into regular intervals over time for a received dataset wherein the user has interacted with the online concierge system **140** over several instances, or episodes. In an instance where the user does not interact with the online concierge system **140**, the propensity for the set of actions is set to 0 and the propensity for the null action is set to 1. The multi-episode OPE policy evaluation is described in further detail below with respect to the policy evaluation module **225** and with regards to FIG. **3**.
[0031] The model serving system **150** receives requests from the online concierge system **140** to perform tasks using machine-learned models. The tasks include, but are not limited to, natural language processing (NLP) tasks, audio processing tasks, image processing tasks, video processing tasks, and the like. In one or more embodiments, the machine-learned models deployed by the model serving system **150** are models configured to perform one or more NLP tasks. The NLP tasks include, but are not limited to, text generation, query processing, machine translation, chatbots, and the like. In one or more embodiments, the language model is configured as a transformer neural network architecture. Specifically, the transformer model is coupled to receive sequential data tokenized into a sequence of input tokens and generates a sequence of output tokens

depending on the task to be performed.
[0032] The model serving system **150** receives a request including input data (e.g., text data, audio data, image data, or video data) and encodes the input data into a set of input tokens. The model serving system **150** applies the machine-learned model to generate a set of output tokens. Each token in the set of input tokens or the set of output tokens may correspond to a text unit. For example, a token may correspond to a word, a punctuation symbol, a space, a phrase, a paragraph, and the like. For an example query processing task, the language model may receive a sequence of input tokens that represent a query and generate a sequence of output tokens that represent a response to the query. For a translation task, the transformer model may receive a sequence of input tokens that represent a paragraph in German and generate a sequence of output tokens that represents a translation of the paragraph or sentence in English. For a text generation task, the transformer model may receive a prompt and continue the conversation or expand on the given prompt in human-like text.
[0033] When the machine-learned model is a language model, the sequence of input tokens or output tokens are arranged as a tensor with one or more dimensions, for example, one dimension, two dimensions, or three dimensions. For example, one dimension of the tensor may represent the number of tokens (e.g., length of a sentence), one dimension of the tensor may represent a sample number in a batch of input data that is processed together, and one dimension of the tensor may represent a space in an embedding space. However, it is appreciated that in other embodiments, the input data or the output data may be configured as any number of appropriate dimensions depending on whether the data is in the form of image data, video data, audio data, and the like. For example, for three-dimensional image data, the input data may be a series of pixel values arranged along a first dimension and a second dimension, and further arranged along a third dimension corresponding to RGB channels of the pixels.
[0034] In one or more embodiments, the language models are large language models (LLMs) that are trained on a large corpus of training data to generate outputs for the NLP tasks. An LLM may be trained on massive amounts of text data, often involving billions of words or text units. The large amount of training data from various data sources allows the LLM to generate outputs for many tasks. An LLM may have a significant number of parameters in a deep neural network (e.g., transformer architecture), for example, at least 1 billion, at least 15 billion, at least 135 billion, at least 175 billion, at least 500 billion, at least 1 trillion, at least 1.5 trillion parameters.
[0035] Since an LLM has significant parameter size and the amount of computational power for inference or training the LLM is high, the LLM may be deployed on an infrastructure configured with, for example, supercomputers that provide enhanced computing capability (e.g., graphic processor units) for training or deploying deep neural network models. In one instance, the LLM may be trained and deployed or hosted on a cloud infrastructure service. The LLM may be pre-trained by the online concierge system **140** or one or more entities different from the online concierge system **140**. An LLM may be trained on a large amount of data from various data sources. For example, the data sources include websites, articles, posts on the web, and the like. From this massive amount of data coupled with the computing power of LLM's, the LLM is able to perform various tasks and synthesize and formulate output responses based on information extracted from the training data.
[0036] In one or more embodiments, when the machine-learned model including the LLM is a transformer-based architecture, the transformer has a generative pre-training (GPT) architecture including a set of decoders that each perform one or more operations to input data to the respective decoder. A decoder may include an attention operation that generates keys, queries, and values from the input data to the decoder to generate an attention output. In another embodiment, the transformer architecture may have an encoder-decoder architecture and includes a set of encoders coupled to a set of decoders. An encoder or decoder may include one or more attention operations.
[0037] While a LLM with a transformer-based architecture is described as a primary embodiment,

it is appreciated that in other embodiments, the language model can be configured as any other appropriate architecture including, but not limited to, long short-term memory (LSTM) networks, Markov networks, BART, generative-adversarial networks (GAN), diffusion models (e.g., Diffusion-LM), and the like.

[0038] In one or more embodiments, the task for the model serving system **150** is based on knowledge of the online concierge system **140** that is fed to the machine-learned model of the model serving system **150**, rather than relying on general knowledge encoded in the model weights of the model. Thus, one objective may be to perform various types of queries on the external data in order to perform any task that the machine-learned model of the model serving system **150** could perform. For example, the task may be to perform question-answering, text summarization, text generation, and the like based on information contained in an external dataset.

[0039] Thus, In one or more embodiments, the online concierge system **140** is connected to an interface system **160**. The interface system **160** receives external data from the online concierge system **140** and builds a structured index over the external data using, for example, another machine-learned language model or heuristics. The interface system **160** receives one or more queries from the online concierge system **140** on the external data. The interface system **160** constructs one or more prompts for input to the model serving system **150**. A prompt may include the query of the user and context obtained from the structured index of the external data. In one instance, the context in the prompt includes portions of the structured indices as contextual information for the query. The interface system **160** obtains one or more responses from the model serving system **160** and synthesizes a response to the query on the external data. While the online concierge system **140** can generate a prompt using the external data as context, often times, the amount of information in the external data exceeds prompt size limitations configured by the machine-learned language model. The interface system **160** can resolve prompt size limitations by generating a structured index of the data and offers data connectors to external data sources.

[0040] FIG. **1**B illustrates an example system environment for an online concierge system **140**, in accordance with one or more embodiments. The system environment illustrated in FIG. **1**B includes a customer client device **100**, a picker client device **110**, a retailer computing system **120**, a network **130**, and an online concierge system **140**. Alternative embodiments may include more, fewer, or different components from those illustrated in FIG. **1**B, and the functionality of each component may be divided between the components differently from the description below. Additionally, each component may perform their respective functionalities in response to a request from a human, or automatically without human intervention.

[0041] The example system environment in FIG. **1**A illustrates an environment where the model serving system **150** and/or the interface system **160** is managed by a separate entity from the online concierge system **140**. In one or more embodiments, as illustrated in the example system environment in FIG. **1**B, the model serving system **150** and/or the interface system **160** is managed and deployed by the entity managing the online concierge system **140**.

[0042] FIG. **2** illustrates an example system architecture for an online concierge system **140**, in accordance with some embodiments. The system architecture illustrated in FIG. **2** includes a data collection module **200**, a content presentation module **210**, an order management module **220**, a machine learning training module **230**, and a data store **240**. Alternative embodiments may include more, fewer, or different components from those illustrated in FIG. **2**, and the functionality of each component may be divided between the components differently from the description below. Additionally, each component may perform their respective functionalities in response to a request from a human, or automatically without human intervention.

[0043] The data collection module **200** collects data used by the online concierge system **140** and stores the data in the data store **240**. The data collection module **200** may only collect data describing a user if the user has previously explicitly consented to the online concierge system **140** collecting data describing the user. Additionally, the data collection module **200** may encrypt all

data, including sensitive or personal data, describing users.
[0044] For example, the data collection module **200** collects customer data, which is information or data that describe characteristics of a customer. Customer data may include a customer's name, address, shopping preferences, favorite items, or stored payment instruments. The customer data also may include default settings established by the customer, such as a default retailer/retailer location, payment instrument, delivery location, or delivery timeframe. The data collection module **200** may collect the customer data from sensors on the customer client device **100** or based on the customer's interactions with the online concierge system **140**.
[0045] The data collection module **200** also collects item data, which is information or data that identifies and describes items that are available at a retailer location. The item data may include item identifiers for items that are available and may include quantities of items associated with each item identifier. Additionally, item data may also include attributes of items such as the size, color, weight, stock keeping unit (SKU), or serial number for the item. The item data may further include purchasing rules associated with each item, if they exist. For example, age-restricted items such as alcohol and tobacco are flagged accordingly in the item data. Item data may also include information that is useful for predicting the availability of items in retailer locations. For example, for each item-retailer combination (a particular item at a particular warehouse), the item data may include a time that the item was last found, a time that the item was last not found (a picker looked for the item but could not find it), the rate at which the item is found, or the popularity of the item. The data collection module **200** may collect item data from a retailer computing system **120**, a picker client device **110**, or the customer client device **100**.
[0046] An item category is a set of items that are a similar type of item. Items in an item category may be considered to be equivalent to each other or that may be replacements for each other in an order. For example, different brands of sourdough bread may be different items, but these items may be in a “sourdough bread” item category. The item categories may be human-generated and human-populated with items. The item categories also may be generated automatically by the online concierge system **140** (e.g., using a clustering algorithm).
[0047] The data collection module **200** also collects picker data, which is information or data that describes characteristics of pickers. For example, the picker data for a picker may include the picker's name, the picker's location, how often the picker has services orders for the online concierge system **140**, a customer rating for the picker, which retailers the picker has collected items at, or the picker's previous shopping history. Additionally, the picker data may include preferences expressed by the picker, such as their preferred retailers to collect items at, how far they are willing to travel to deliver items to a customer, how many items they are willing to collect at a time, timeframes within which the picker is willing to service orders, or payment information by which the picker is to be paid for servicing orders (e.g., a bank account). The data collection module **200** collects picker data from sensors of the picker client device **110** or from the picker's interactions with the online concierge system **140**.
[0048] Additionally, the data collection module **200** collects order data, which is information or data that describes characteristics of an order. For example, order data may include item data for items that are included in the order, a delivery location for the order, a customer associated with the order, a retailer location from which the customer wants the ordered items collected, or a timeframe within which the customer wants the order delivered. Order data may further include information describing how the order was serviced, such as which picker serviced the order, when the order was delivered, or a rating that the customer gave the delivery of the order. In some embodiments, the order data includes user data for users associated with the order, such as customer data for a customer who placed the order or picker data for a picker who serviced the order.
[0049] The content presentation module **210** selects content for presentation to a customer. For example, the content presentation module **210** selects which items to present to a customer while the customer is placing an order. The content presentation module **210** generates and transmits the

ordering interface for the customer to order items. The content presentation module **210** populates the ordering interface with items that the customer may select for adding to their order. In some embodiments, the content presentation module **210** presents a catalog of all items that are available to the customer, which the customer can browse to select items to order. The content presentation module **210** also may identify items that the customer is most likely to order and present those items to the customer. For example, the content presentation module **210** may score items and rank the items based on their scores. The content presentation module **210** displays the items with scores that exceed some threshold (e.g., the top n items or the p percentile of items).
[0050] The content presentation module **210** may use an item selection model to score items for presentation to a customer. An item selection model is a machine learning model that is trained to score items for a customer based on item data for the items and customer data for the customer. For example, the item selection model may be trained to determine a likelihood that the customer will order the item. In some embodiments, the item selection model uses item embeddings describing items and customer embeddings describing customers to score items. These item embeddings and customer embeddings may be generated by separate machine learning models and may be stored in the data store **240**.
[0051] In some embodiments, the content presentation module **210** scores items based on a search query received from the customer client device **100**. A search query is free text for a word or set of words that indicate items of interest to the customer. The content presentation module **210** scores items based on a relatedness of the items to the search query. For example, the content presentation module **210** may apply natural language processing (NLP) techniques to the text in the search query to generate a search query representation (e.g., an embedding) that represents characteristics of the search query. The content presentation module **210** may use the search query representation to score candidate items for presentation to a customer (e.g., by comparing a search query embedding to an item embedding).
[0052] In some embodiments, the content presentation module **210** scores items based on a predicted availability of an item. The content presentation module **210** may use an availability model to predict the availability of an item. An availability model is a machine learning model that is trained to predict the availability of an item at a retailer location. For example, the availability model may be trained to predict a likelihood that an item is available at a retailer location or may predict an estimated number of items that are available at a retailer location. The content presentation module **210** may weight the score for an item based on the predicted availability of the item. Alternatively, the content presentation module **210** may filter out items from presentation to a customer based on whether the predicted availability of the item exceeds a threshold.
[0053] The order management module **220** that manages orders for items from customers. The order management module **220** receives orders from a customer client device **100** and assigns the orders to pickers for service based on picker data. For example, the order management module **220** assigns an order to a picker based on the picker's location and the location of the retailer from which the ordered items are to be collected. The order management module **220** may also assign an order to a picker based on how many items are in the order, a vehicle operated by the picker, the delivery location, the picker's preferences on how far to travel to deliver an order, the picker's ratings by customers, or how often a picker agrees to service an order.
[0054] In some embodiments, the order management module **220** determines when to assign an order to a picker based on a delivery timeframe requested by the customer with the order. The order management module **220** computes an estimated amount of time that it would take for a picker to collect the items for an order and deliver the ordered item to the delivery location for the order. The order management module **220** assigns the order to a picker at a time such that, if the picker immediately services the order, the picker is likely to deliver the order at a time within the timeframe. Thus, when the order management module **220** receives an order, the order management module **220** may delay in assigning the order to a picker if the timeframe is far enough in the

future.
[0055] When the order management module **220** assigns an order to a picker, the order management module **220** transmits the order to the picker client device **110** associated with the picker. The order management module **220** may also transmit navigation instructions from the picker's current location to the retailer location associated with the order. If the order includes items to collect from multiple retailer locations, the order management module **220** identifies the retailer locations to the picker and may also specify a sequence in which the picker should visit the retailer locations.
[0056] The order management module **220** may track the location of the picker through the picker client device **110** to determine when the picker arrives at the retailer location. When the picker arrives at the retailer location, the order management module **220** transmits the order to the picker client device **110** for display to the picker. As the picker uses the picker client device **110** to collect items at the retailer location, the order management module **220** receives item identifiers for items that the picker has collected for the order. In some embodiments, the order management module **220** receives images of items from the picker client device **110** and applies computer-vision techniques to the images to identify the items depicted by the images. The order management module **220** may track the progress of the picker as the picker collects items for an order and may transmit progress updates to the customer client device **100** that describe which items have been collected for the customer's order.
[0057] In some embodiments, the order management module **220** tracks the location of the picker within the retailer location. The order management module **220** uses sensor data from the picker client device **110** or from sensors in the retailer location to determine the location of the picker in the retailer location. The order management module **220** may transmit to the picker client device **110** instructions to display a map of the retailer location indicating where in the retailer location the picker is located. Additionally, the order management module **220** may instruct the picker client device **110** to display the locations of items for the picker to collect, and may further display navigation instructions for how the picker can travel from their current location to the location of a next item to collect for an order.
[0058] The order management module **220** determines when the picker has collected all of the items for an order. For example, the order management module **220** may receive a message from the picker client device **110** indicating that all of the items for an order have been collected. Alternatively, the order management module **220** may receive item identifiers for items collected by the picker and determine when all of the items in an order have been collected. When the order management module **220** determines that the picker has completed an order, the order management module **220** transmits the delivery location for the order to the picker client device **110**. The order management module **220** may also transmit navigation instructions to the picker client device **110** that specify how to travel from the retailer location to the delivery location, or to a subsequent retailer location for further item collection. The order management module **220** tracks the location of the picker as the picker travels to the delivery location for an order, and updates the customer with the location of the picker so that the customer can track the progress of their order. In some embodiments, the order management module **220** computes an estimated time of arrival for the picker at the delivery location and provides the estimated time of arrival to the customer.
[0059] In some embodiments, the order management module **220** facilitates communication between the customer client device **100** and the picker client device **110**. As noted above, a customer may use a customer client device **100** to send a message to the picker client device **110**. The order management module **220** receives the message from the customer client device **100** and transmits the message to the picker client device **110** for presentation to the picker. The picker may use the picker client device **110** to send a message to the customer client device **100** in a similar manner.
[0060] The order management module **220** coordinates payment by the customer for the order. The

order management module **220** uses payment information provided by the customer (e.g., a credit card number or a bank account) to receive payment for the order. In some embodiments, the order management module **220** stores the payment information for use in subsequent orders by the customer. The order management module **220** computes a total cost for the order and charges the customer that cost. The order management module **220** may provide a portion of the total cost to the picker for servicing the order, and another portion of the total cost to the retailer.
[0061] The policy evaluation module **225** evaluates the performance of contextual policies for several instances in time. The policy evaluation module **225** predicts which policies are improving outcomes by bringing an understanding of what works and why for a user through the evaluation of contextual policies. In one or more embodiments, the policy evaluation module **225** estimates the impact of multiple episodes for a new contextual policy rather than a single episode which would not fully represent the joint effects of multiple episodes on a user's behavior.
[0062] During an experiment, the policy evaluation module **225** receives a first indication a user interacts with an application at a first time. For a first time, the policy evaluation module **225** predicts a set of propensities for a set of actions for the user by applying a first machine learning model to user features for the user. The first machine learning model may be a logging contextual policy. For example, the policy evaluation module **225** receives the first indication a user interacts with a mobile application or web application of the online concierge system **140** and predicts the set of propensities for a set of actions under the logging policy. In one example, the set of actions are a set of coupons presented to the user: a first action may be no coupon presented, a second action may be presenting the user with a 10% off coupon, a third action may be presenting the user with a 20% off coupon. In the example illustrated in FIG. **3**, the policy evaluation module **225** selects the first action for the first time for the user based on the set of propensities because, for example, the first action had the highest propensity by the logging policy. The policy evaluation module **225** treats the user with the selected action.
[0063] The policy evaluation module **225** receives a second indication that the user interacts with the application for a second time. For the second time, the policy evaluation module **225** predicts a second set of propensities for the set of actions under the logging policy. The user selects the third action for the second time for the user based on the second set of propensities and treats the user with the selected action.
[0064] For a second machine learning model for evaluation, the policy evaluation module **225** divides the time period into regular time intervals where each interval may be denoted h=1, 2, ... H. The intervals are divided, for instance, over a month into separate days where h=1, 2, . . . , H indicates each day of the month over which the evaluation policy is evaluated. FIG. **3** illustrates an example timeline and process for performing multi-episodic OPE, according to one embodiment. For a particular user, the policy evaluation module **225** divides up the time range into regular intervals of h=1, 2, 3, . . . , 8.
[0065] In one or more embodiments, when the user does not interact with an application of the online concierge system **140**, the policy evaluation module **225** “assigns” the user a null action for that interval and sets the propensity for the remaining actions as 0 and sets the propensity for the null action as 1. The policy evaluation module **225**, when the user does not interact with the online concierge system **140** selects the null action.
[0066] To evaluate the second machine learning model, the policy evaluation module **225** at least determines an importance weight based on the actions assigned to the user during the experiment using the first ML model as the logging policy. In one instance, the multi-episodic OPE framework is given by at each time interval h, combining the importance weight and the reward for the time interval, and summing the values across the time intervals. The importance weight for a current time interval h is given by a cumulative multiplication of the ratios between the propensity of the selection action under the evaluation policy over the propensity of the selection action under the logging policy for the current time interval h and also previous time intervals h−1, h−2, h−3, . . . ,

0.
[0067] In one or more embodiments, the IPS for an evaluation policy π.sub.T is given by:

[00001] $$\hat{v} = \frac{1}{N} \underset{n=1}{\overset{N}{.\text{Math}.}} \quad \underset{h=1}{\overset{H}{.\text{Math}.}} \quad {}^{h-1} \quad {}_{h} r_{h}$$

[0068] where ρ.sub.h is the importance weight for time interval h=1 through H, r.sub.h is the reward associated with the action selected at time interval h=1 through H, and N is the number of users, and gamma is a weighting factor (constant).
[0069] In one instance, ρ.sub.h is given by:

[00002] $${}_{h} = \underset{h'=1}{\overset{h}{.\text{Math}.}} \frac{{}_{T}(a_i \text{ .Math. } x_i)}{{}_{L}(a_i \text{ .Math. } x_i)}$$

where ρ.sub.h is the importance weight containing the ratio of the evaluation policy's propensity score to the logging policy's propensity score.
[0070] FIG. **3** illustrates an example timeline wherein an episodic off policy is used to evaluate the evaluation policy. The policy evaluation module **225** regularly divides time intervals from h=1, 2, 3, . . . , 8. For time intervals h=1, 3-6, and 8, the null action, Action 4, is selected for the user. At time interval h=2, a first indication that the user interacted with the online concierge system **140** is indicated. Action 1 is selected for time interval h=2 under the logging policy. The importance ratio, denoted by the term ρ.sub.h=2, is the ratio of the propensity of action 1 under the evaluation policy over the propensity of action 1 under the logging policy. The importance weight **320** at time h=2 is:

[00003] $${}_{h=2} = \frac{{}_{T}(a_1 \text{ .Math. } x_i)}{{}_{L}(a_1 \text{ .Math. } x_i)}.$$

[0071] At time interval h=7, a second indication that the user interacted with the online concierge system **140** is illustrated. Action 3 is selected for time interval h=7 under the logging policy. The importance weight **310** at time h=7 is:

[00004] $${}_{h=7} = \frac{{}_{T}(a_1 \text{ .Math. } x_i)}{{}_{L}(a_1 \text{ .Math. } x_i)} \times \frac{{}_{T}(a_3 \text{ .Math. } x_i)}{{}_{L}(a_3 \text{ .Math. } x_i)}$$

[0072] where the π.sub.T(a.sub.1 |x.sub.i) is the propensity of Action 1 under the evaluation policy π.sub.T under the context x.sub.i for user i, and π.sub.L(a.sub.1|x.sub.i) is the propensity of Action 1 under the logging policy π.sub.L under the context for user i. Similarly, the quantity π.sub.T(a.sub.3|x.sub.i) is the propensity of Action 3 under the evaluation policy π.sub.T under the context x.sub.i for user i, and π.sub.L(a.sub.3|x.sub.i) is the propensity of Action 3 under the logging policy π.sub.L under the context for user i. ρ.sub.h=7 is the cumulative multiplication of the ratio at h=2 and the ratio at h=7. A larger value of the second importance weight may be an indicator that the evaluation policy has a strong propensity of selecting an action for the user with a strong potential reward.
To evaluate the evaluation policy, the policy evaluation module **225** determines the inverse propensity score (IPS) for the evaluation policy. For example, in FIG. **3**, the IPS for the evaluation policy π.sub.T is

[00005] $$\hat{v} = \underset{h=1}{\overset{8}{.\text{Math}.}} \quad {}_{h} r_{h} = \quad {}_{h=2} r_{h=2} + \quad {}_{h=7} r_{h=7} \hat{v} = \left(\frac{{}_{T}(a_1 \text{ .Math. } x_i)}{{}_{L}(a_1 \text{ .Math. } x_i)} \times r_1\right) + \left(\frac{{}_{T}(a_1 \text{ .Math. } x_i)}{{}_{L}(a_1 \text{ .Math. } x_i)} \times \frac{{}_{T}(a_3 \text{ .Math. } x_i)}{{}_{L}(a_3 \text{ .Math. } x_i)} \times r_3\right)$$

[0073] The IPS evaluates the evaluation policy π.sub.T for a single user (N=1), which serves as a natural unbiased estimator for policy π.sub.T.
[0074] In one or more embodiments, the policy evaluation module **225** may compute the IPS for multiple, different types of untested contextual policies. The policy evaluation module **225** may select a policy for experiment or deployment based on the values of the IPS analysis. For example, the policies with the highest IPS values may be chosen. By performing the multi-episodic off-policy evaluation described herein, the joint effects across multiple episodes can be incorporated in the evaluation, leading to more accurate evaluations.
[0075] Each machine learning model includes a set of parameters. A set of parameters for a

machine learning model are parameters that the machine learning model uses to process an input. For example, a set of parameters for a linear regression model may include weights that are applied to each input variable in the linear combination that comprises the linear regression model. Similarly, the set of parameters for a neural network may include weights and biases that are applied at each neuron in the neural network. The machine learning training module **230** generates the set of parameters for a machine learning model by "training" the machine learning model. Once trained, the machine learning model uses the set of parameters to transform inputs into outputs.
[0076] The machine learning training module **230** trains a machine learning model based on a set of training examples. Each training example includes input data to which the machine learning model is applied to generate an output. For example, each training example may include customer data, picker data, item data, or order data. In some cases, the training examples also include a label which represents an expected output of the machine learning model. In these cases, the machine learning model is trained by comparing its output from input data of a training example to the label for the training example.
[0077] For example, when training a contextual policy, the training examples may include multiple instances of users, where each instance denotes a user, contextual information obtained for the user (e.g., demographic information, geographic information, etc.), labels for the action space for the user (e.g., 1 for action that resulted in reward, 0 otherwise).
[0078] The machine learning training module **230** may apply an iterative process to train a machine learning model whereby the machine learning training module **230** trains the machine learning model on each of the set of training examples. To train a machine learning model based on a training example, the machine learning training module **230** applies the machine learning model to the input data in the training example to generate an output. The machine learning training module **230** scores the output from the machine learning model using a loss function. A loss function is a function that generates a score for the output of the machine learning model such that the score is higher when the machine learning model performs poorly and lower when the machine learning model performs well. In cases where the training example includes a label, the loss function is also based on the label for the training example. Some example loss functions include the mean square error function, the mean absolute error, hinge loss function, and the cross entropy loss function. The machine learning training module **230** updates the set of parameters for the machine learning model based on the score generated by the loss function. For example, the machine learning training module **230** may apply gradient descent to update the set of parameters.
[0079] In one or more embodiments, the contextual policies described herein may be implemented via one or more neural network models, each including a plurality of layers of nodes connected to each other by edges. A logging policy (e.g., TL) and a new evaluation policy may be different from one another because, for example, the evaluation policy (e.g., AT) has a different architecture for a neural network or may have a different set of parameters than the logging policy. For example, as the policy evaluation module **225** updates training data with recent instances, the evaluation policy may be re-trained using the new training instances, resulting in a new set of parameters.
[0080] The data store **240** stores data used by the online concierge system **140**. For example, the data store **240** stores customer data, item data, order data, and picker data for use by the online concierge system **140**. The data store **240** also stores trained machine learning models trained by the machine learning training module **230**. For example, the data store **240** may store the set of parameters for a trained machine learning model on one or more non-transitory, computer-readable media. The data store **240** uses computer-readable media to store data, and may use databases to organize the stored data.
[0081] With respect to the machine-learned models hosted by the model serving system **150**, the machine-learned models may already be trained by a separate entity from the entity responsible for the online concierge system **140**. In another embodiment, when the model serving system **150** is included in the online concierge system **140**, the machine-learning training module **230** may further

train parameters of the machine-learned model based on data specific to the online concierge system **140** stored in the data store **240**. As an example, the machine-learning training module **230** may obtain a pre-trained transformer language model and further fine tune the parameters of the transformer model using training data stored in the data store **240**. The machine-learning training module **230** may provide the model to the model serving system **150** for deployment.

[0082] FIG. **4** is a flowchart for a method of a policy evaluation module **225** in accordance with some embodiments. Alternative embodiments may include more, fewer, or different steps from those illustrated in FIG. **4**, and the steps may be performed in a different order from that illustrated in FIG. **4**. These steps may be performed by an online concierge system (e.g., online concierge system **140**). Additionally, each of these steps may be performed automatically by the online concierge system without human intervention. A policy evaluation module receives **400** a first indication where a user interacts with an application. The module predicts **410** a set of propensities for a set of actions for the user by applying a first machine learning model for the user. The module selects **420** the first action for the first time for the user and treats the user with the selected action. The policy evaluation module receives **430** a second indication the user interacts with the application. The policy evaluation module predicts **440** a second set of propensities for the set of actions for the second indication. The module selects **450** a second action for the user based on the predicted set of propensities. The module evaluates **460** a second machine learning model by determining a first importance ratio, determining a second importance weight, and combining the first importance ratio with the second importance weight. The module deploys **470** the second machine learning model.

Additional Considerations

[0083] The foregoing description of the embodiments has been presented for the purpose of illustration; many modifications and variations are possible while remaining within the principles and teachings of the above description.

[0084] Any of the steps, operations, or processes described herein may be performed or implemented with one or more hardware or software modules, alone or in combination with other devices. In some embodiments, a software module is implemented with a computer program product comprising one or more computer-readable media storing computer program code or instructions, which can be executed by a computer processor for performing any or all of the steps, operations, or processes described. In some embodiments, a computer-readable medium comprises one or more computer-readable media that, individually or together, comprise instructions that, when executed by one or more processors, cause the one or more processors to perform, individually or together, the steps of the instructions stored on the one or more computer-readable media. Similarly, a processor comprises one or more processors or processing units that, individually or together, perform the steps of instructions stored on a computer-readable medium.

[0085] Embodiments may also relate to a product that is produced by a computing process described herein. Such a product may store information resulting from a computing process, where the information is stored on a non-transitory, tangible computer-readable medium and may include any embodiment of a computer program product or other data combination described herein.

[0086] The description herein may describe processes and systems that use machine learning models in the performance of their described functionalities. A “machine learning model,” as used herein, comprises one or more machine learning models that perform the described functionality. Machine learning models may be stored on one or more computer-readable media with a set of weights. These weights are parameters used by the machine learning model to transform input data received by the model into output data. The weights may be generated through a training process, whereby the machine learning model is trained based on a set of training examples and labels associated with the training examples. The training process may include: applying the machine learning model to a training example, comparing an output of the machine learning model to the label associated with the training example, and updating weights associated for the machine

learning model through a back-propagation process. The weights may be stored on one or more computer-readable media, and are used by a system when applying the machine learning model to new data.
[0087] The language used in the specification has been principally selected for readability and instructional purposes, and it may not have been selected to narrow the inventive subject matter. It is therefore intended that the scope of the patent rights be limited not by this detailed description, but rather by any claims that issue on an application based hereon.
[0088] As used herein, the terms “comprises,” “comprising,” “includes,” “including,” “has,” “having,” or any other variation thereof, are intended to cover a non-exclusive inclusion. For example, a process, method, article, or apparatus that comprises a list of elements is not necessarily limited to only those elements but may include other elements not expressly listed or inherent to such process, method, article, or apparatus. Further, unless expressly stated to the contrary, “or” refers to an inclusive “or” and not to an exclusive “or”. For example, a condition “A or B” is satisfied by any one of the following: A is true (or present) and B is false (or not present), A is false (or not present) and B is true (or present), and both A and B are true (or present). Similarly, a condition “A, B, or C” is satisfied by any combination of A, B, and C being true (or present). As a not-limiting example, the condition “A, B, or C” is satisfied when A and B are true (or present) and C is false (or not present). Similarly, as another not-limiting example, the condition “A, B, or C” is satisfied when A is true (or present) and B and C are false (or not present).

Claims

1. A method comprising: receiving a first indication that a user interacts with an application of an online system at a first time; for a first time, predicting a set of propensities for a set of actions for the user by applying a first machine learning model to user features for the user; selecting a first action for the first time for the user based on the set of propensities and treating the user with the selected first action; receiving a second indication the user interacts with the application at a second time; for the second time, predicting a second set of propensities for the set of actions; selecting a second action for the second time for the user based on the second set of propensities and treating the user with the selected second action; evaluating a second machine learning model by at least: identifying a first importance weight including at least a ratio between a propensity generated by the second machine learning model to the first machine learning model for the first action; identifying a second importance weight including at least the first importance weight and a ratio between a propensity generated by the second model to the first model for the second action; and combining the first importance weight for the first time with a reward associated with the first action and the second importance weight for the second time with a second reward to evaluate the second machine learning model; and deploying the second machine learning model responsive to identifying that the result of the evaluation meets a criteria.
2. The method of claim 1, wherein evaluating the second machine learning model further comprises dividing a time span including the first time and the second time into regular time intervals.
3. The method of claim 2, wherein evaluating the second machine learning model further comprises for a time where the user did not select an action, assigning a null action to user with a propensity of 1.
4. The method of claim 1, wherein the second machine learning model has a different set of parameters or a different architecture from the first machine learning model.
5. The method of claim 1, wherein identifying the second importance weight comprises multiplying the first importance weight and a ratio between a propensity generated by the second machine learning model to a propensity generated by the first machine learning model for the second action.
6. The method of claim 1, wherein receiving the first indication the user interacts with the application of the online system further comprises receiving an indication the user accesses the

application on a client device associated with the user by logging into the application.
7. The method of claim 1, wherein training the second machine learning model comprises of training data that may include multiple instances of users, where each instance denotes a user, contextual information obtained for the user, or labels for an action space for the user.
8. The method of claim 1, wherein deploying the second machine learning model comprises deploying the second machine learning model responsive to identifying that the result of the evaluation is above a threshold value.
9. The method of claim 8, further comprising evaluating a third machine learning model and identifying not to deploy the third machine learning model responsive to identifying the result of the evaluation for the third machine learning model is below the threshold value.
10. A non-transitory computer-readable storage medium storing instructions that when executed by a computer processor cause the computer processor to perform steps comprising: receiving a first indication that a user interacts with an application of an online system at a first time; for a first time, predicting a set of propensities for a set of actions for the user by applying a first machine learning model to user features for the user; selecting a first action for the first time for the user based on the set of propensities and treating the user with the selected first action; receiving a second indication the user interacts with the application at a second time; for the second time, predicting a second set of propensities for the set of actions; selecting a second action for the second time for the user based on the second set of propensities and treating the user with the selected second action; evaluating a second machine learning model by at least: identifying a first importance weight including at least a ratio between a propensity generated by the second machine learning model to the first machine learning model for the first action; identifying a second importance weight including at least the first importance weight and a ratio between a propensity generated by the second model to the first model for the second action; and combining the first importance weight for the first time with a reward associated with the first action and the second importance weight for the second time with a second reward to evaluate the second machine learning model; and deploying the second machine learning model responsive to identifying that the result of the evaluation meets a criteria.
11. The non-transitory computer-readable storage medium of claim 10, wherein evaluating the second machine learning model further comprises dividing a time span including the first time and the second time into regular time intervals.
12. The non-transitory computer-readable storage medium of claim 11, wherein evaluating the second machine learning model further comprises for a time where the user did not select an action, assigning a null action to user with a propensity of 1.
13. The non-transitory computer-readable storage medium of claim 10, wherein the second machine learning model has a different set of parameters or a different architecture from the first machine learning model.
14. The non-transitory computer-readable storage medium of claim 10, wherein identifying the second importance weight comprises multiplying the first importance weight and a ratio between a propensity generated by the second machine learning model to a propensity generated by the first machine learning model for the second action.
15. The non-transitory computer-readable storage medium of claim 10, wherein receiving the first indication the user interacts with the application of the online system further comprises receiving an indication the user accesses the application on a client device associated with the user by logging into the application.
16. The non-transitory computer-readable storage medium of claim 10, wherein training the second machine learning model comprises of training data that may include multiple instances of users, where each instance denotes a user, contextual information obtained for the user, or labels for an action space for the user.
17. The non-transitory computer-readable storage medium of claim 10, wherein deploying the

second machine learning model comprises deploying the second machine learning model responsive to identifying that the result of the evaluation is above a threshold value.
18. The non-transitory computer-readable storage medium of claim 17, further comprising evaluating a third machine learning model and identifying not to deploy the third machine learning model responsive to identifying the result of the evaluation for the third machine learning model is below the threshold value.
19. A computer system, the computer system comprising: a computer processor; and a non-transitory computer-readable storage medium storing instructions that when executed by a computer processor cause the computer processor to perform steps comprising: receiving a first indication that a user interacts with an application of an online system at a first time; for a first time, predicting a set of propensities for a set of actions for the user by applying a first machine learning model to user features for the user; selecting a first action for the first time for the user based on the set of propensities and treating the user with the selected first action; receiving a second indication the user interacts with the application at a second time; for the second time, predicting a second set of propensities for the set of actions; selecting a second action for the second time for the user based on the second set of propensities and treating the user with the selected second action; evaluating a second machine learning model by at least: identifying a first importance weight including at least a ratio between a propensity generated by the second machine learning model to the first machine learning model for the first action; identifying a second importance weight including at least the first importance weight and a ratio between a propensity generated by the second model to the first model for the second action; and combining the first importance weight for the first time with a reward associated with the first action and the second importance weight for the second time with a second reward to evaluate the second machine learning model; and deploying the second machine learning model responsive to identifying that the result of the evaluation meets a criteria.
20. The computer system of claim 19, wherein evaluating the second machine learning model further comprises dividing a time span including the first time and the second time into regular time intervals.
21. The computer system of claim 20, wherein evaluating the second machine learning model further comprises, for a time where the user did not select an action, assigning a null action to user with a propensity of 1.
22. The computer system of claim 19, wherein the second machine learning model has a different set of parameters or a different architecture from the first machine learning model.
23. The computer system of claim 19, wherein identifying the second importance weight comprises multiplying the first importance weight and a ratio between a propensity generated by the second machine learning model to a propensity generated by the first machine learning model for the second action.
24. The computer system of claim 19, wherein receiving the first indication the user interacts with the application of the online system further comprises receiving an indication the user accesses the application on a client device associated with the user by logging into the application.
25. The computer system of claim 19, wherein deploying the second machine learning model comprises deploying the second machine learning model responsive to identifying that the result of the evaluation is above a threshold value.