A Deep Reinforcement Learning-Based Approach to Query-Free Interactive Target Item Retrieval

Anna Sepliarskaia
TU Wien
anna.sepliarskaia@tuwien.ac.at

Sahika Genc
Amazon
sahika@amazon.com

Maarten de Rijke
University of Amsterdam
& Ahold Delhaize
m.derijke@uva.nl

ABSTRACT

We consider the task of query-free interactive target item retrieval. In this task, a user has a concept or category of items in mind and the retrieval system has to find the right item that falls within the category. Early in a session, the degree of uncertainty about the category of items of interest to the user is high. Hence, it may be more efficient to explicitly ask users about their preference than to use a traditional recommender system (RS) approach that displays very similar items that have the highest estimate of being relevant. We propose a deep reinforcement learning-based approach for relevance feedback interactions between user and system. We introduce an actor-critic framework to iteratively select sets of items based on real-time relevance feedback from users and their purchase history, thereby maximizing satisfaction with the entire session. We compare our proposal with state-of-the-art relevance feedback methods as well as RSs; it leads to increased user satisfaction within a session, independent of the way in which we measure user satisfaction and of the number of items displayed on the result page.

1 INTRODUCTION

E-commerce sites have become ubiquitous [4]. Customers use them to explore available items or purchase a specific item [21, 22]. Often, users want to find and obtain an item from a particular category [22]. This category can be represented by various "mental pictures," which vary from a particular image to the user's impression [5]. In this scenario the goal of a user at the start of a session may be highly ambiguous from the point of view of the retrieval system. Hence, it may be efficient for the system to ask the user to provide explicit feedback, such as relevance feedback about the current result page. In this context, it is important to show result pages for which relevance feedback is informative and that contain items that are close to the target category. This is the problem that we address in this paper, i.e., the task of *showing to a user result pages that the user likes and for which relevance feedback is informative.*

A common approach for helping a user find a desired product is to use relevance feedback (RF) mechanisms [3, 7, 8, 26, 27] to minimize the recommender system (RS)'s uncertainty about the user's target category. In RF, a system shows a subset of items to a user and the user interacts by clicking on the item that is (supposedly) most similar to the target item. Systems that use an RF mechanism to understand a user's current preference aim to minimize the length of the session by displaying diverse items on the result pages. Our goal is not only to minimize the length of the session, but also to show to a user result pages that she likes and for which relevance feedback is informative.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGIR eCom'21, July 15, 2021, Virtual Event, Montreal, Canada
© 2021 Copyright held by the owner/author(s).

We propose a reinforcement learning (RL)-based approach for showing to a user result pages that the user likes and for which relevance feedback is informative. Our approach, called *ActorCriticRS*, is based on the Actor-Critic framework; it maximizes the expected long-term cumulative reward per user, where the cumulative reward corresponds to the user's satisfaction during the session. ActorCriticRS can automatically trade-off between greedy exploitation of learned past interests and exploration to uncover current interests.

Our contribution is three-fold: (1) We propose a deep reinforcement learning-based framework for relevance feedback-based item retrieval. (2) We propose simple and effective implementations of each component of the framework. (3) We analyze the quality of the framework along three dimensions: (a) the number of items displayed on the result page; (b) the duration of the session measured by the number of result pages shown during the session; and (c) our approximation of user satisfaction with the result page.

2 PROBLEM DESCRIPTION

We consider the problem of *showing to a user result pages that the user likes and for which relevance feedback is informative.* A solution to this problem is useful when a RS does not have enough information about a user's goal at the beginning of a session and should infer it during the session using relevance feedback. There, a user wants to find an item from a target category but does not know the category name and, therefore, does not provide a query.

The task. Consider a user with a history of purchases who initiates a new session on an e-commerce site. When she starts the session, she has a concept or category in mind, but cannot formulate it and does not submit a query. The RS and user interact through relevance feedback to uncover the user's implicit need. Those interactions consist of the following steps: (1) the user starts a search session in which she wants to find an item from the target category c; (2) the recommender shows n items from its catalogue $I = \{i\}_1^n$; (3) the user provides feedback by clicking on the best item from I; (4) the recommender obtains a reward r equal to the user satisfaction of the result page; (5) the recommender processes the feedback and creates a new list of items; and (6) the session ends when the session is too long. The task of the RS is to maximize the cumulative reward:

$$\sum_{t=0}^{t=\infty} \gamma^t \cdot r_t, \tag{1}$$

where r_t is the user satisfaction obtained on the t-th result page of the session; r_t is equal to 0 if the user leaves the recommender system examining less than t result pages. We propose a *reinforcement learning* (RL) approach to finding a strategy that a recommender agent should follow so as to maximize Eq. 1.

Recommendation as a Markov decision process. We formulate the interaction between a user and a RS as a Markov decision process (MDP) and use RL to automatically learn an optimal strategy for selecting items to display on the result page. The optimal strategy is obtained by maximizing the long-term cumulative reward (Eq. 1) during the session. We define five components required for an MDP (S, A, r, p, γ). First, a *state* $s \in S$ is an approximation of the current user preference. At the start of the session, the initial state s_0 is generated using the user's purchase history:

$$s_0 = Purchases = \{i_1, \dots, i_k\}, \tag{2}$$

where $i_1, \dots, i_k$ are the items bought before. On the t-th result page a user provides feedback by clicking on one of the shown items:

$$F_t = (i^t_{clicked}, \{i^t_{1,non\text{-}clicked}, \dots, i^t_{n-1,non\text{-}clicked}\}), \tag{3}$$

where $i^{t+1}_{clicked}$ is a clicked item, $i^{t+1}_{j,non\text{-}clicked}$ is a non-clicked item. After the t-th round of interactions between, the state s_t is generated based on the user's purchase history and feedback received so far during the session: $s_t = (Purchases, F_1, \dots, F_t)$. Second, the *action space* A is defined as the set of all subsets containing n items: $\{i_1, \dots, i_n\} \in A$. The chosen items are displayed on the result page. Third, the user examines the result page, and the RS obtains an *immediate reward* r_t, equal to the user's satisfaction with the current result page. Fourth, the *transition probability* $p(s_{t+1} \mid s_t, a_t)$ defines the state transition from s_t given an action a_t. Fifth, $\gamma \in [0, 1]$ determines the discount factor for future rewards. If $\gamma = 0$, the RS becomes a greedy agent and considers only immediate rewards. If $\gamma = 1$, the RS considers all future rewards without discount.

The RS interacts with a user during a sequence of time steps. At each time step the RS chooses an action by displaying a set of items and receives a reward from the user, that is, her satisfaction with the result page. A user clicks on the item closest to the target category. The environment updates its state.

3 FRAMEWORK

Our framework for solving the task of showing to a user result pages that the user likes and for which relevance feedback is informative has three components: (1) an *item and user embedder*; (2) a *state generator*, and (3) a *recommender agent.*

Item and user embedder. The item and user embedder uses the history of users' purchases to find a low-dimensional representation of users and items. The representation of users is used as an initial state when a new query-free session starts, and should approximate users' preferences. Dimensions in the latent representation of items should reflect items' most valuable characteristics. The dot product between a user's representation and an item's representation should reflect the user's preference for this item. The problem of finding such a low-dimensional representation using the history of users' feedback for items, can be solved by a recommendation algorithm; we choose one that uses not only a user's purchase history but also images of items, Visual Bayesian Personalized Ranking (VBPR) [9].

State generator. A state generator generates a low-dimensional representation of the current state. As a vector representation of the initial state $\mathbf{s}_0$ we use the user representation generated by the *item and user embedder.* While the user interacts with the system, the representation of the state changes to reflect information gleaned from the session in progress. The representation of the state $\mathbf{s_t}$ should be changed to reflect the user's clicks on the $(t + 1)$-st result page of the session. We use an RF mechanism to improve the relevance of items on a result page. As a state generator, we use the Rocchio algorithm [18]: move the state $\mathbf{s}_t$ in the direction of the clicked item and in a direction opposite to non-clicked items:

$$\mathbf{s}_{t+1} = a \cdot \mathbf{s}_t + (1 - a) \cdot \frac{1}{n-1} \cdot \sum (\mathbf{i}_{clicked} - \mathbf{i}_{non\text{-}clicked}), \tag{4}$$

where a balances the influence of new information on information gained earlier, and n is the number of elements displayed on the page. As feedback from the user becomes less informative once we have received a user's relevance for several result pages we set a to be a function of t: $a(t) = 1 - 0.8^t \cdot a'$, where a' is a parameter of the *state generator.* The final formula for updating states is:

$$\mathbf{s}_{t+1} = a(t) \cdot \mathbf{s}_t + (1 - a(t)) \cdot \frac{1}{n-1} \sum (\mathbf{i}_{clicked} - \mathbf{i}_{non\text{-}clicked}). \tag{5}$$

Recommender agent. The recommender agent uses a low-dimensional representation of the state to produce a result page of items. After displaying the page, the recommender agent obtains a reward, which is the user satisfaction for the page. The task for the agent is to maximize the expected cumulative reward (see Eq. 1) obtained during the session. This task can be solved by an RL approach. However, the action space is very large; the number of items alone exceeds 100K on ordinary e-commerce platforms. Moreover, the space of states is continuous. Following [29], where a complete result page is viewed as an action, we use the Actor-Critic framework [12]. It consists of two parts: an Actor and a Critic.

The *Actor* chooses n items to display on a result page, depending on the current state $\mathbf{s}_t$. The Actor first generates a pseudo-action, consisting of n vectors in the latent space $\{\hat{\mathbf{i}}_1, \dots, \hat{\mathbf{i}}_n\}$. Then, for each generated vector $\hat{\mathbf{i}}_j$, the most similar item is selected that was not yet shown in the session:

$$i_j = \arg\max_{i_j} (\mathbf{i}_j^T \cdot \hat{\mathbf{i}}_j). \tag{6}$$

This procedure of selecting items, using vectors generated by the Actor, follows the Mapping Algorithm [29]; see Algorithm 1. The Actor should change its behavior over time. With more feedback received from a user, the state $\mathbf{s}_t$ moves closer to the embedding of the target item. To achieve this, we design an Actor consisting of two components: (1) one is trained, and (2) the other is greedy. The trained component $A_{trained}$ generates n vectors $\{\widetilde{\mathbf{i}}_1, \dots, \widetilde{\mathbf{i}}_n\}$ in the latent space and is a neural network with one fully connected layer and a tanh non-linearity. Vectors generated by the Actor linearly combine the current state $\mathbf{s}_t$ and vectors generated by $A_{trained}$:

$$\hat{\mathbf{i}}_j = (1 - \epsilon^{actor}_t)\, \widetilde{\mathbf{i}}_j + \epsilon^{actor}_t \mathbf{s}_t, \tag{7}$$

where $\epsilon^{actor}_t \in [0, 1]$ is monotonically increasing with t:

$$\epsilon^{actor}_t = (1 + \exp(-a_{actor} \cdot t + b_{actor}))^{-1}, \tag{8}$$

and a_{actor} and b_{actor} are trainable parameters of the Actor.

Algorithm 1 Mapping from vector generated by the Actor to items

```
1: Input: Set of items $\mathbb{I}$ that have not been displayed before, $n$ vectors in the latent space $\{\hat{\mathbf{i}}_1, \dots, \hat{\mathbf{i}}_n\}$ that are generated by the Actor.
2: Output: Set of items $I_{current} = \{i_1, \dots, i_n\}$, that will be shown to a user.
3: for $j = 1, \dots, n$ do
4:     Select the most similar item $i_j \in \mathbb{I}$ according to Eq. 6
5:     Add $i_j$ to $I_{current}$
6:     Remove item $i_j$ from $\mathbb{I}$
```

The *Critic* predicts the action-value function $Q(\mathbf{s}_t, \mathbf{a}_t)$ in the current state $\mathbf{s}_t$, if the action taken is a_t. In RSs, the state and action spaces are very large, therefore, estimating $Q(s_t, a_t)$ for each state-action pair is infeasible. We approximate the Q-value using a neural network as the Critic, which has a similar architecture as the Actor. The Critic also consists of two parts. The first is a neural network with a fully connected layer and a tanh non-linearity. The second is an addition ϵ_t^{critic} that depends on step t:

$$\epsilon_t^{critic} = (1 + \exp(-a_{critic} \cdot t + b_{critic}))^{-1}, \tag{9}$$

where a_{critic} and b_{critic} are trainable parameters of the Critic.

4 EXPERIMENTS

We address two questions: (RQ1) Does user satisfaction within a session increase when items are displayed to a user according to the strategy obtained by an RL-based method instead of a traditional greedy strategy? (RQ2) Does the number of items displayed on a result page affect the performance of the recommender agent?

Dataset. We use a crawl of time stamped item reviews from Amazon [13], with two groups of items, *Women's Clothing* and *Electronics.* Following [9], we take users' review histories as implicit feedback (if the user wrote a review, this is considered positive, otherwise it is negative). We process the groups differently since item popularity differs between them: for *Electronics* there is a subset of very popular products, but not for *Women's Clothing*. We process *Women's Clothing* so that each user has ≥ 5 reviews, resulting in ~100K users, ~330K items, ~850K reviews. We process *Electronics* so that each item is bought ≤ 20 times and each user has ≥ 3 reviews, resulting in ~40K users, ~80K items, ~160K reviews.

As visual information about items, we use the image features extracted in [9]. One image is collected for each item. The Caffe reference model [6], which implements a CNN architecture [10], is used. It has 5 convolutional layers followed by 3 fully-connected layers. The output of the second fully connected layer (i.e., FC7) is used to obtain a 4,096-dimensional vector of visual features.

The dataset contains a taxonomy. We create a *Categorical space* by a hard-code embedding of the items: each item contains 0 and 1 as coordinates, where the coordinate equals 1 if an item belongs to the corresponding category in the taxonomy. Below, we use the *Categorical space* to simulate user clicks in a session, with ~2,700 categories in *Women's Clothing* and ~1,300 in *Electronics.*

Simulation. To evaluate our strategy of selecting items to show to a user we need to obtain users' feedback on the selected items. This information is available to us, hence, we simulate users' feedback using the *Categorical space*: in the simulation, a user always clicks on the item that is most similar to the target one in the *Categorical space.* When items are equally good, a simulated user randomly clicks on one. In our simulation for each user, we choose the last purchased item as the target item; all other purchases are used to estimate the user's preferences. The taxonomy is *not* used by any of our recommendation methods: *all methods only use the users' history of purchases and features from images of the items.*

Baselines. Below, we describe four baseline strategies for selecting items to show to a user. The *item and user embedder* and *state generator* remain the same for all the methods that we compare. First, the *Random* baseline randomly selects items to display on the result page. Second, an efficient algorithm for query-free image retrieval using interactive RF, *IRF*, has been proposed in [8]. IRF achieves good quality in content-based image retrieval with RF [2, 23, 24]. Since IRF does not scale to large collections of items, we first select 500 items with the highest predicted relevance scores, and then use IRF only for those items. Third, a *greedy* strategy is commonly used for selecting items to show to a user [3, 9, 14, 16, 26, 27]; it selects items with the highest predicted relevance scores. In this paper, the predicted relevance scores are proportional to the scalar product between low-dimensional representations of items received from the *item and user embedder* and low-dimensional representations of the current state. Fourth, we use *multileave gradient descent* (MGD) [19, 20]; a similar algorithm has already been used in RS [31]. At each timestep t, MGD selects n rankers. Then, using clicks, the best ranker of the n selected rankers is inferred. MGD adjusts the prediction of the vector representation of the globally best ranker. MGD is based on Dueling Bandit Gradient Descent (DBGD) [28] but selects n rankers, instead of 2.

Evaluation methodology. To answer the research questions listed above we experiment with a standard configuration for RSs, changing one parameter of the standard configuration at a time. We display 10 items on the result page and use r_{mean} as an estimation of user satisfaction with the result page:

$$r_{mean} = \frac{1}{n} \cdot \sum_{i_k} relevance(i_k). \tag{10}$$

To answer RQ1, we compare the r_{mean} score obtained by MGD and ActorCriticRS with the r_{mean} score obtained by the *Greedy* approach. To understand the impact of the number of items shown to a user on the result page, we compare r_{mean} obtained by *Random*, *IRF*, *Greedy*, *MGD* and ActorCriticRS in the standard configuration and vary the number of items shown (3, 5, 7, 10 items).

Training models. To obtain visual-semantic representations of users and items, we use the code from [9]. All purchases are divided into three subsets: training, validation and testing. We use the last purchases as a test set. The validation set is created by selecting for each user u a random item v_u from her purchases that are not in the test set. All remaining data is used for training. We train the VBPR model with 32 dimensions, each, of the non-visual and visual latent representations. The state generator has only one hyperparameter, namely a', which we set to 0.5. We tried other values of a' (0.3, 0.5, 0.8), but the cumulative reward obtained by *Greedy*, *IRF* and MGD on the validation set was the largest when a' is set to 0.5 (for both categories). For all configurations, we use the same hyperparameters of ActorCriticRS. The batch size is equal to 128, the learning rate of the Actor is $1e-6$, the learning rate of the

Table 1: Rewards obtained on the result pages 1–5 on the *Women's Clothing* and *Electronics* datasets. Methods: (1) Random; (2) IRF; (3) Greedy; (4) MGD; (5) ActorCriticRS. Results marked with $*$ are significantly better than the next best performing approach; results marked with $\triangleright$ perform equally well and are significantly better than other approaches (t-test, p-value < 0.01).

		Women's Clothing					*Electronics*				
		$r_{mean}(\cdot)$					$r_{mean}(\cdot)$				
	Method	p_1	p_2	p_3	p_4	p_5	p_1	p_2	p_3	p_4	p_5
3 items shown	(1)	0.39	0.39	0.39	0.39	0.39	0.32	0.32	0.32	0.32	0.32
	(2)	0.46	0.49	0.50	0.51	0.52	0.33	**0.34**	**0.35**$^{\triangleright}$	**0.36**$^{\triangleright}$	**0.37**
	(3)	**0.47**$^{\triangleright}$	0.48	0.49	0.49	0.50	**0.34***	**0.34***	**0.35**$^{\triangleright}$	**0.36**$^{\triangleright}$	0.36
	(4)	**0.47**$^{\triangleright}$	0.49	0.50	0.51	**0.52**$^{\triangleright}$	0.33	0.33	0.34	0.35	0.36
	(5)	0.46	**0.50***	**0.51***	**0.52***	**0.52**$^{\triangleright}$	0.32	**0.34**	**0.35**$^{\triangleright}$	**0.36**$^{\triangleright}$	**0.37***
5 items shown	(1)	0.39	0.39	0.39	0.39	0.39	0.32	0.32	0.32	0.32	0.32
	(2)	0.46	0.50	0.51	0.52	0.53	0.33	0.34	0.35	0.36	0.37
	(3)	**0.47**$^{\triangleright}$	0.49	0.50	0.51	0.51	**0.34***	**0.35***	**0.36**	0.37	0.37
	(4)	**0.47**$^{\triangleright}$	0.50	0.51	0.53	0.53	0.32	0.33	0.34	0.36	0.37
	(5)	0.45	**0.51***	**0.53***	**0.54***	**0.55***	0.32	0.34	**0.36**	**0.38***	**0.39***
7 items shown	(1)	0.39	0.39	0.39	0.39	0.39	0.32	0.32	0.32	0.32	0.32
	(2)	0.46	0.50	0.52	0.53	0.54	**0.33**	0.34	**0.36**	0.36	0.37
	(3)	**0.47**	0.49	0.50	0.51	0.52	**0.33***	**0.35***	**0.36**	0.37	0.38
	(4)	0.46	0.50	0.52	0.54	**0.55**	**0.33**	0.34	0.35	0.37	0.38
	(5)	0.46	**0.51***	**0.53***	**0.55***	**0.55**	0.32	0.34	**0.36**	**0.38***	**0.40***
10 items shown	(1)	0.39	0.39	0.39	0.39	0.39	0.32	0.32	0.32	0.32	0.32
	(2)	0.46	0.51	0.52	0.53	0.54	**0.33**	0.34	0.36	0.37	0.37
	(3)	**0.47***	0.50	0.51	0.52	0.53	**0.33***	**0.35***	0.36	0.37	0.38
	(4)	0.46	0.50	0.52	0.54	**0.56**$^{\triangleright}$	**0.33**	0.34	0.36	0.37	0.39
	(5)	0.45	**0.52***	**0.54***	**0.55***	**0.56**$^{\triangleright}$	0.32	0.34	**0.37***	**0.39***	**0.40***

Critic is $1e-4$. To train the Actor-Critic framework we use Deep Deterministic Policy Gradients (DDPGs) [12]. For every method, we perform 10-fold cross-validation and report the average outcome.

5 RESULTS

Table 1 lists the experimental outcomes in terms of reward $r_{mean}(\cdot)$ after displaying the first (p_1) through fifth (p_5) result page in a session, for both the *Women's Clothing* and *Electronics* datasets. ActorCriticRS consistently obtains the highest reward from the second or third result page of the session.

Does reinforcement learning help to improve user satisafaction? To understand whether a reinforcement learning approach helps to improve user satisfaction with a session, we turn to Table 1. We compare the *Greedy* method (3) with two RL-based methods: *MGD* (4) and ActorCriticRS (5). The results support our claim that for the task of query-free interactive target item retrieval for experienced users, an RL-based approach is useful and outperforms traditional RSs in terms of a user's satisfaction with a session once the session as moved beyond the initial result page. However, an RL-based approach should be chosen carefully when the *item and user embedder* provides a noisy representation (i.e., early on in the session).

Does the number of items displayed impact the performance? To understand how the number of items displayed on a result page affects the performance, we turn to Table 1 again. We compare the quality of the algorithms depending on the number of items displayed on the result page. For each method we compare its results depending on the number of displayed items. For both datasets, *Random* obtains the same r_{mean} regardless of the number of displayed items. Other methods obtain larger rewards when the number of items displayed on the result page increases.

6 RELATED WORK

Collaborative Filtering (CF) is an effective traditional technique to build personalized recommender systems [17]. CF-based methods collect user-item ratings and derive preference patterns but do not use information about the sequence in which the ratings were given nor contextual information about the items. Context-aware CF methods use both the contextual item features as well as ratings [9, 13]. Unlike traditional recommendation techniques, the items displayed on the result page produced by ActorCriticRS adapt to evolving user tastes and are diverse.

Methods that use RL for recommendation consider recommendation as a sequential process. Each episode consists of interactions between a user and an RS, ordered by time. In traditional RSs, items displayed on the page are predicted to satisfy users' preference the most, whereas in RL-based approaches the system maximizes long-term user satisfaction with recommendations. *Multi-armed bandits* are a successful RL technique used for recommendation [1, 11, 25]. Another RL-technique that is widely used for recommendation is Deep Reinforcement Learning (DRL) [12, 15, 29–31]. Unlike those methods, ActorCriticRS uses a RF mechanism.

In RF, a user interacts with a system, marking some items as relevant, some as irrelevant, or noting that some items are more relevant than others [32]. Using users' feedback, the system revises its results and displays items that better satisfy users' tastes. RF is used when it is difficult to formulate a good query, but it is easy to judge a particular set of items [2, 3, 5, 8, 23, 26]. Unlike traditional RF mechanisms, ActorCriticRS works on top of historical interactions.

7 CONCLUSION

We have considered the problem of showing to a user result pages that the user likes and for which relevance feedback is informative. Solving it is useful when a user does not know how to formulate a query but wants to buy a product from a specific category. In this case she is unlikely to find an item that satisfies her preferences on conventional e-commerce sites where interaction is query-based.

We have proposed a three-part framework: (1) an *item and user embedder*, (2) a *state generator*, and (3) a *recommender agent*. We have focused on the implementation of the recommender agent while using simple yet effctive solutions for the other components. We have proposed a model-free deep reinforcement learning-based algorithm, ActorCriticRS, that selects a set of items to display on the result page. We find an optimal strategy by maximizing the expected user satisfaction with the whole session. ActorCriticRS outperforms competing approaches when showing result pages that maximize user satisfaction with the entire session. In future work we want to investigate how different instantiations of the components of the framework affect overall user satisfaction.

REFERENCES

[1] Avinash Balakrishnan, Djallel Bouneffouf, Nicholas Mattei, and Francesca Rossi. 2018. Using Contextual Bandits with Behavioral Constraints for Constrained Online Movie Recommendation.. In *IJCAI*. 5802–5804.

[2] Björn Barz, Christoph Käding, and Joachim Denzler. 2018. Information-Theoretic Active Learning for Content-Based Image Retrieval. *arXiv preprint arXiv:1809.02337* (2018).

[3] Keping Bi, Qingyao Ai, and W. Bruce Croft. 2018. Revisiting Iterative Relevance Feedback for Document and Passage Retrieval. *arXiv preprint arXiv:1812.05731* (2018).

[4] Census Bureau 2015. E-Stats: Measuring the Electronic Economy. https://www.census.gov/programs-surveys/e-stats.html.

[5] Zhuoxiang Chen, Zhe Xu, Ya Zhang, and Xiao Gu. 2018. Query-Free Clothing Retrieval via Implicit Relevance Feedback. *IEEE Transactions on Multimedia* 20, 8 (2018), 2126–2137.

[6] Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. 2014. Decaf: A Deep Convolutional Activation Feature for Generic Visual Recognition. In *International Conference on Machine Learning*. 647–655.

[7] Bahaeddin Eravci and Hakan Ferhatosmanoglu. 2018. Diverse Relevance Feedback for Time Series with Autoencoder based Summarizations. *IEEE Transactions on Knowledge and Data Engineering* 30, 12 (2018), 2298–2311.

[8] Marin Ferecatu and Donald Geman. 2009. A Statistical Framework for Image Category Search from a Mental Picture. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 31, 6 (2009), 1087–1101.

[9] Ruining He and Julian McAuley. 2016. VBPR: Visual Bayesian Personalized Ranking from Implicit Feedback.. In *AAAI*. 144–150.

[10] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems*. 1097–1105.

[11] Lihong Li, Wei Chu, John Langford, and Robert E Schapire. 2010. A Contextual-bandit Approach to Personalized News Article Recommendation. In *Proceedings of the 19th International Conference on World Wide Web*. ACM, 661–670.

[12] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. 2015. Continuous Control with Deep Reinforcement Learning. *arXiv preprint arXiv:1509.02971* (2015).

[13] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton Van Den Hengel. 2015. Image-based Recommendations on Styles and Substitutes. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 43–52.

[14] Andriy Mnih and Ruslan R Salakhutdinov. 2008. Probabilistic Matrix Factorization. In *Advances in Neural Information Processing Systems*. 1257–1264.

[15] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Human-level Control through Deep Reinforcement Learning. *Nature* 518, 7540 (2015), 529.

[16] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian Personalized Ranking from Implicit Feedback. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*. AUAI Press, 452–461.

[17] Francesco Ricci, Lior Rokach, and Bracha Shapira. 2011. Introduction to Recommender Systems Handbook. In *Recommender Systems Handbook*. Springer, 1–35.

[18] Joseph John Rocchio. 1971. Relevance Feedback in Information Retrieval. In *The SMART Retrieval System: Experiments in Automatic Document Processing*. Prentice-Hall Inc., 313–323.

[19] Anne Schuth, Harrie Oosterhuis, Shimon Whiteson, and Maarten de Rijke. 2016. Multileave Gradient Descent for Fast Online Learning to Rank. In *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining*. ACM, 457–466.

[20] Anne Schuth, Floor Sietsma, Shimon Whiteson, Damien Lefortier, and Maarten de Rijke. 2014. Multileaved Comparisons for Fast Online Evaluation. In *CIKM 2014: 23rd ACM Conference on Information and Knowledge Management*. ACM, 71–80.

[21] Parikshit Sondhi, Mohit Sharma, Pranam Kolari, and ChengXiang Zhai. 2018. A Taxonomy of Queries for E-commerce Search. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. ACM, 1245–1248.

[22] Ning Su, Jiyin He, Yiqun Liu, Min Zhang, and Shaoping Ma. 2018. User Intent, Behaviour, and Perceived Satisfaction in Product Search. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*. ACM, 547–555.

[23] Nicolae Suditu and Francois Fleuret. 2011. HEAT: Iterative Relevance Feedback with One Million Images. In *2011 International Conference on Computer Vision*. IEEE, 2118–2125.

[24] Nicolae Suditu and François Fleuret. 2012. Iterative Relevance Feedback with Adaptive Exploration/Exploitation Trade-off. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*. ACM, 1323–1331.

[25] Liang Tang, Yexi Jiang, Lei Li, and Tao Li. 2014. Ensemble Contextual Bandits for Personalized Recommendation. In *Proceedings of the 8th ACM Conference on Recommender Systems*. ACM, 73–80.

[26] Angadpreet Walia, Tanisha Gahlawat, Parul Kalra, and Deepti Mehrotra. 2017. An Emperical Study: Relevance Feedback in Information Retrieval Systems. In *2017 International Conference on Current Trends in Computer, Electrical, Electronics and Communication (CTCEEC)*. IEEE, 982–985.

[27] Heng Xu, Jun-yi Wang, and Lei Mao. 2017. Relevance Feedback for Content-based Image Retrieval using Deep Learning. In *2nd International Conference on Image, Vision and Computing (ICIVC)*. IEEE, 629–633.

[28] Yisong Yue and Thorsten Joachims. 2009. Interactively Optimizing Information Retrieval Systems as a Dueling Bandits Problem. In *Proceedings of the 26th Annual International Conference on Machine Learning*. ACM, 1201–1208.

[29] Xiangyu Zhao, Long Xia, Liang Zhang, Zhuoye Ding, Dawei Yin, and Jiliang Tang. 2018. Deep Reinforcement Learning for Page-wise Recommendations. In *Proceedings of the 12th ACM Conference on Recommender Systems*. ACM, 95–103.

[30] Xiangyu Zhao, Liang Zhang, Zhuoye Ding, Dawei Yin, Yihong Zhao, and Jiliang Tang. 2017. Deep Reinforcement Learning for List-wise Recommendations. *arXiv preprint arXiv:1801.00209* (2017).

[31] Guanjie Zheng, Fuzheng Zhang, Zihan Zheng, Yang Xiang, Nicholas Jing Yuan, Xing Xie, and Zhenhui Li. 2018. DRN: A Deep Reinforcement Learning Framework for News Recommendation. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 167–176.

[32] Xiang Sean Zhou and Thomas S. Huang. 2003. Relevance Feedback in Image Retrieval: A Comprehensive Review. *Multimedia Systems* 8, 6 (2003), 536–544.