Reinforcement Learning for Effective Few-Shot Ranking

ABSTRACT

Neural rankers have achieved strong retrieval effectiveness but require large amounts of labeled data, limiting their applicability in *few-shot settings*. In this paper, we address the sample inefficiency of neural ranking methods by introducing a Reinforcement Learning (RL)-based re-ranking model that achieves high effectiveness with minimal training data. Built on a Deep Q-learning Network (DQN) framework, our approach is designed for *few-shot* settings, maximizing sample efficiency to ensure robust generalization from limited interactions. Extensive experiments show that our model significantly outperforms data-intensive methods and existing few-shot baselines, demonstrating RL's potential to enhance IR capabilities in *few-shot* scenarios.

1 INTRODUCTION

Neural rankers have greatly enhanced IR effectiveness [1], with transformer architectures [2] playing a key role in capturing contextual and semantic relationships between queries and documents. However, these models typically require vast amounts of labeled training data to perform well, limiting their applicability in *few-shot* settings, where only a small number of labeled examples are available due to time, cost, or data constraints [3]. A natural solution in *few-shot settings* is lexical retrievers like BM25 [4], which rank documents based on term frequency statistics without requiring training. However, these models rely on surface-level term matching and so fail to capture deep semantic relationships [5, 6]. Neural rankers overcome this limitation by leveraging large-scale training data [7, 8], but their reliance on extensive labeled data makes them impractical in few-shot settings. To address this, recent work has explored self-supervised learning [9] and weak supervision [10]. However, these methods still often require substantial data and remain insufficient for *few-shot* environments.

Background Approaches. Given the limitations of both traditional lexical models and current neural ranking methods in few-shot settings, there is a growing need for ranking approaches that can effectively operate with minimal labeled data. One promising direction is to explore methods that can learn from limited feedback, rather than relying on large labeled datasets. Sun et al. [11] proposed MetaAdaptRank which employs synthesizing contrastive weak supervision and using meta-learning to filter noisy signals. Unlike MetaAdaptRank, which generates synthetic data, Sinhababu et al. [12] proposed a method that leverages prompting by retrieving similar queries from a training set and using them as pairwise ranking examples during inference. This augmentation allows LLMs to make more informed ranking decisions, improving both in-domain and out-of-domain retrieval without requiring model fine-tuning. On the other hand, P^3 Ranker [13] bridges the gap between pre-trained language models (PLMs) and ranking tasks by using prompt-based learning to align ranking with PLM training and pre-finetuning to inject ranking-specific knowledge. Unlike the two aforementioned methods, the P^3 Ranker focuses on structured PLM adaptation, making it suitable for *few-shot ranking with minimal labeled data*. While P^3 Ranker demonstrates strong performance in few-shot settings, its effectiveness still depends on pre-finetuning, which may not always be feasible when intermediate tasks are unavailable or when labeled data is highly limited.

On the other hand, Reinforcement Learning (RL) [14] provides a suitable framework by enabling models to learn optimal ranking behaviors through interactions and rewards rather than extensive labeled data. Contrary to common belief, RL can be effective in certain *few-shot scenarios* [15–17]. By framing ranking as a sequential decision-making task [18], RL allows models to iteratively refine rankings based on feedback signals, making it particularly adaptable in *few-shot scenarios*. Additionally, RL's ability to optimize actions based on accumulated rewards aligns with the objective of ranking in information retrieval—maximizing document relevance over time. This capability enables RL-based ranking methods to achieve strong performance even with limited annotated data.

Reinforcement learning (RL) [19] has gained traction in several information retrieval (IR) tasks, particularly in modeling document ranking as a sequential decision-making process through Markov Decision Processes (MDPs). In this framework, at each time step, an agent selects a document based on the current observation (e.g., ranking position and remaining unranked documents), with rewards often defined in terms of ranking metrics like NDCG (Normalized Discounted Cumulative Gain). Various IR tasks, such as session search, have been formulated as MDPs to model user interactions over multiple queries, optimizing document ranking across sessions [20, 21]. Similarly, RL-based ranking has been applied to search result diversification [20, 22] and multi-page search [23], where the RL agent learns to balance relevance and diversity across search results. Specific approaches such as MDPRank [23–25] and REINFORCE-based document ranking [18] optimize ranking policies using policy gradient methods. For instance, in [22], search result diversification is modeled as an MDP, where each ranking position represents a decision point, and the agent selects documents sequentially. However, while policy gradient methods provide flexibility in handling high-dimensional action spaces, they tend to be sample-inefficient, requiring extensive interactions with the environment due to noisy gradient estimates and high variance in training [26]. This inefficiency poses challenges for few-shot settings, where only limited labeled examples are available.

The CUOLR model [27] extends the MDP-based ranking framework by making the ranking task click model-agnostic, enabling generalization across different user feedback models. To achieve this, CUOLR incorporates the Soft Actor-Critic (SAC) algorithm, a reinforcement learning approach originally designed for continuous action spaces. However, SAC's performance and sample efficiency degrade in discrete action spaces due to its design for continuous domains. Additionally, actor-critic algorithms like SAC rely on an on-policy critic, whereas value-based methods like DQN typically achieve better performance in discrete-action environments [28].

Rationale and Proposed Approach. To address sample inefficiency, which limits methods like MDPRank in few-shot settings, we propose a ranking strategy based on Deep Q-learning Networks (DQN), a sample-efficient value-based RL approach [29, 30]. In this framework, we approximate the Q-function with a neural network to learn the expected reward of ranking decisions. The key features of our approach that make it well-suited for few-shot settings include: **(1)** *Experience replay*, which stores and reuses past interactions, such as previous rankings and document selections, breaking correlations between consecutive ranking decisions and enhancing learning diversity—critical when training data is limited. **(2)** *Temporal credit assignment* [31], which evaluates long-term rewards, allowing the model to learn cumulative effects over time rather than focusing solely on immediate rewards. This is particularly valuable in ranking, where a document's position may have delayed effects on overall ranking quality.

Key Contributions. We address the challenge of sample inefficiency in RL-based ranking for few shot settings, proposing a re-ranking model specifically designed to perform effectively with limited training data. Our approach leverages DQN to maximize data efficiency and improve generalization in few-shot settings. Our approach enables the model to learn robust ranking policies from a minimal training dataset, achieving competitive ranking effectiveness even in data-constrained scenarios. We provide empirical evidence that our model can achieve ranking performance surpassing lexical ranking methods that do not require training data and far superior performance to neural rankers that by their nature require significantly larger training datasets. We further show that our approach surpasses earlier RL-based rankers, such as MDPRank, in learning from limited training data. This advantage stems from our explicit focus on managing and improving sample efficiency, making our model more effective in *few-shot* settings.

2 PROPOSED APPROACH

Let us assume that for a few-shot settings (FSS), there exists a query pool Q_{FSS} consisting of a limited set of queries $Q_{FSS} = \{q_1, q_2, \ldots, q_n\}$. Further let each query $q_i \in Q_{FSS}$ be associated with a set of relevant documents D_{q_i} with k documents $D_{q_i} = \{d_1, d_2, \ldots, d_k\}$. The objective of our task is to train a re-ranking model *FSRank* with this small-size training dataset.

Description of the RL Model. We formulate document re-ranking as a Markov Decision Process (MDP) and optimize it using reinforcement learning, where the re-ranking objective is modeled as a sequence of decisions by an RL agent. Markov Decision Processes (MDPs) [32] are stochastic models well-suited for sequential decision-making. We frame ranking as an MDP, where each step involves selecting a document for the next position in the list. This formulation enables the integration of contextual information, such as the current time step and remaining documents, into the state representation, leading to more informed ranking decisions. The MDP in our work is defined as a quadruple $\langle S, A, T, R \rangle$, representing states, actions, a transition function, and rewards as follows:

States. The states S represent the environment. For ranking, the agent must be aware of the current ranking position and the set of candidate documents C. At time step t, the state s_t is defined as the pair $[t, C_t]$, where C_t denotes the unsorted set of candidate documents that remain to be ranked.

Actions. The actions A refer to the set of discrete actions available to the agent. The feasible actions are determined by the current state s_t and are represented as $A(s_t)$. At each time step t, the agent takes action $a_t \in A(s_t)$, which involves selecting a document $c_i \in C_t$ for the next ranking position $t + 1$.

Transition function. The transition function $T(s, a)$ returns the next state $s_{t+1} \in S$ resulting from taking action a_t in state s_t. In a deterministic environment, the outcome of this function is unique, meaning that for each state-action pair, there is a specific next state. In a given state, s_t, after taking action a_t, the next state is constructed by updating the candidate set and also incrementing the time step. the candidate set C_t is updated by removing the chosen document c_i from the candidate set and the time step is incremented by one, forming the next state s_{t+1} according to Equation 1:

$$s_{t+1} = T(s_t, a_t) = [t + 1, C_{t+1}] \quad \text{where} \quad C_{t+1} = C_t \setminus \{c_i\} \tag{1}$$

Reward. The reward $\mathcal{R}(S, A)$ provides immediate feedback, also known as reinforcement. It represents the reward the agent receives for executing action $a_t \in A(s_t)$. In the context of ranking, the action a_t corresponds to the selection of a document c_i and $\mathcal{R}(s_t, a_t)$ is correlated to the quality of c_i. The function $R(s, a)$ is designed to prioritize positioning the most relevant documents at the top. Thus, it can depend on the relevancy of the document c_i selected by action a_t, denoted as $\Psi(c_i)$, and its position. To promote the early selection of highly relevant documents, we apply a time-based penalty. The reward function is formulated according to Equation 2:

$$\mathcal{R}(s_t, a_t) = \frac{\Psi(c_i)}{\log_2(t + 1)} \quad where \quad a_t : select \quad c_i \in C_t \tag{2}$$

As shown in Equation 2, the logarithmic denominator of the current time step t ensures that selecting relevant documents earlier yields a higher reward, encouraging the agent to place the most relevant documents at the top of the ranked list.

In this context, the model consists of two components: **(1)** a language model which serves as the feature extractor and whose weights are not updated during the training. This language model takes a concatenated query and document pair as input and generates a vector representation. The current time-step t is then concatenated to this vector representation to build a feature vector, x, which acts as the feature for the RL agent: $x = LM(q \oplus d) + t$.
(2) The agent consists of two components: a) an experience replay buffer, B, which stores and randomizes past experiences to enhance stability, and b) a neural network $\mathcal{N}$. The agent interacts with the environment to determine the optimal action for each state. Optimal parameter estimation for MDPs can be achieved using dynamic programming methods like value iteration [33] or RL approaches like Q-learning [34]. Given the limited data in few-shot settings, we require a sample-efficient RL algorithm. DQN is one of the most data-efficient RL methods as it leverages experience replay [35], allowing the agent to reuse past experiences, break sample correlations, and enhance training stability. Traditional RL methods like Q-learning struggle with high-dimensional state spaces due to their inflexibility in scaling state-action pairs. To address this, we use Deep Q-Network (DQN), which employs a neural network as a non-linear function approximator to estimate the action-value function in RL. Unlike standard Q-learning, DQN learns an action-value function rather than the optimal policy [36]. At each time step t, corresponding to a ranking position, the RL agent selects the next document to fill that position.

Through the action-value function, the agent determines the optimal action in each state by interacting with the environment. Optimal parameter estimation for MDPs can be achieved using

Algorithm Sample Collection and RL Model Training

```
1:  Q ← Initialize(φ_0)
2:  B ← ∅,    max_size = N     ▷ Phase 1: Filling the Replay Buffer
3:  for q ∈ P_FSS do
4:      for timestep t: 0 ≤ t < len(D_{q_i}) do
5:          a_t ~ Uniform(D_{q_i})
6:          Execute action a_t, observe (r_t, s_{t+1})
7:          B ← B ∪ {(s_t, a_t, r_t, s_{t+1})}
8:          if |B| = max_size then
9:              break
10:         end if
11:     end for
12: end for
            ▷ Phase 2: Sampling from Replay Buffer and RL Model Training
13: for {(s_i, a_i, r_i, s_{i+1})} ~ Uniform(B) do
14:     Q(s_i, a_i; φ_i) = N(x_{a_i}; φ_i)
15:     for a′ ∈ C_{i+1} do
16:         Q′ = N(x_{a′}; φ_i)
17:     end for
18:     U_i = max_{Q′}
19:     Ω_{a_i} = r_i + γU_i
20:     L(φ) = (Ω_{a_i} − Q(s_i, a_i))^2
21:     φ_{i+1} ← φ_i − η∇L(φ_i)
```

dynamic programming methods like value iteration [33] or RL approaches like Q-learning [34]. Given the limited data in few-shot settings, we require a sample-efficient RL algorithm. DQN is among the most data-efficient RL methods, leveraging experience replay [35] to enable multiple reuses of past experiences, break correlations between consecutive samples, and improve training stability. Additionally, traditional RL methods like Q-learning become inflexible as the number of state-action pairs increases, especially in high-dimensional state representations. To address this, we use Deep Q-Network (DQN), which employs a neural network as a non-linear approximator to estimate the action-value function. Unlike standard Q-learning, DQN learns an action-value function rather than directly optimizing the MDP policy [36]. At each time step t, corresponding to a specific ranking position, the RL agent selects the next document to fill that position.

Action Value Function. The action value function, i.e., $Q(s,a)$, estimates the expected future rewards of taking action a_t in state s_t. It combines immediate rewards and discounted future rewards to provide a measure of the value of actions. In an MDP, the future reward is worth less than the current reward and therefore a discount factor $\gamma \in (0,1)$ is applied to future rewards. This discount factor along with the time step-related penalty in the reward function encourages the RL agent to try to select the most relevant documents sooner in order to maximize its total reward. The neural network in our RL agent, $\mathcal{N}$, attempts to estimate the value of Q. We propose to train this network using Deep Q-Networks (DQN) [30]. DQN is a popular reinforcement learning algorithm that utilizes a neural network parameterized by ϕ to estimate the Q-value. The input to this network, x_{a_t}, is the feature vector of action a_t. At a given time step t, we calculate the value of action a_t according to Equation 3:

$$Q(s, a_t; \phi_t) = \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k} \mid s_0 = s_t, a_0 = a_t\right] \tag{3}$$

where $\gamma \in [0,1)$ is the discount factor, which determines the importance of future rewards, and r_t is the reward received at time step t. For each action, the expected value is defined as the sum of the immediate reward and the expected future reward.

The Learning Process. Our proposed learning process consists of two phases, explained below and formally described in Algorithm 1:

The Experience Collection Phase: The first phase of our RL process accumulates experiences in the experience replay buffer B to stabilize training and improve efficiency in low-resource settings. Experience replay enhances data efficiency by allowing each experience tuple to contribute to multiple weight updates [29, 37]. We ensure sufficient sampling so each experience is revisited multiple times. Additionally, randomizing samples mitigates correlations between consecutive experiences, reducing variance and improving stability [29]. Finally, experience replay helps prevent catastrophic forgetting, where new experiences overwrite prior knowledge, a critical issue in data-scarce domains (*few shot scenario*) where forgetting learned interactions can degrade performance [38, 39].

The Training Phase: Once the replay buffer is filled, our RL process randomly samples from the replay buffer and updates the network based on these samples as shown on Line 14 of Algorithm 1. Randomly sampling experiences to update the network breaks the correlation between consecutive experiences, leading to better learning performance [37, 40]. Additionally, to compensate for limited data availability, our approach ensures that each experience has multiple opportunities to be seen by the network by sampling from our replay buffer sufficiently. For each experience tuple (s_t, a_t, r_t, s_{t+1}), the feature vector of action a_t, x_{a_t}, is constructed using the language model LM. Then x_{a_t} is processed through the network $\mathcal{N}$ to output the current Q-value, $\hat{Q}(a_t)$, which needs to approximate the target value, Ω_{a_t}. The current Q-value is defined in Equation 4 as follows:

$$\hat{Q}(s_t, a_t; \phi) = \mathcal{N}(x_{a_t}; \phi_t) \tag{4}$$

On this basis, Q' is calculated for all the possible actions in s_{t+1}. The maximum value of Q' is denoted as U_i and represents the maximum reward that can be expected by taking action a_t and transitioning to state s_{t+1}. The target Q-value, Ω_{a_t}, is calculated as the sum of immediate reward, r_t and the discounted U_i. This is shown in Lines 15-19 of Algorithm 1 and Equation 5, as follows:

$$\Omega_{a_t} = r_t + \gamma \max_{a'} Q(s_{t+1}, a'; \phi_t) \tag{5}$$

As the RL model is trained, it is expected $\hat{Q}$ to move towards Ω to indicate how much reward can be expected if an action a_t is taken in time step t. In order to find the optimal values of $\phi*$ for the networks, we adopt the mean squared error (MSE) between $\hat{Q}$ and Ω, shown in Equation 6, as the training loss function. This corresponds to Line 20 in Algorithm 1.

$$L(\phi) = \mathbb{E}\left[\left(\Omega_{a_t} - \hat{Q}(s_t, a_t; \phi_t)\right)^2\right] \tag{6}$$

Finally, the weights of the network are updated using gradient descent and learning rate η, as shown on Line 21 of Algorithm 1.

3 EXPERIMENTS

Research Questions (RQs). We explore three research questions as follows: (**RQ1**) we assess whether our proposed model is *generalizabile* on different language models and whether it shows *stable performance* when the number of training samples change; (**RQ2**) we investigate whether the performance of our proposed model is competitive with existing state of the art neural ranking models, a state-of-the art few shot ranker, and the unsupervised lexical BM25 approach; and, (**RQ3**) we further explore whether our RL-based

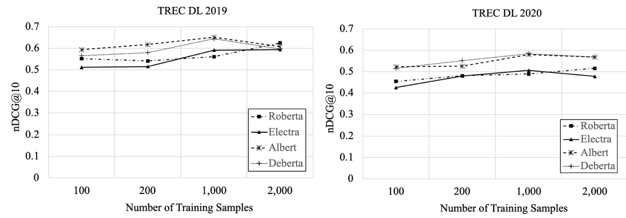

Figure 1: Generalizability on different LLMs & train set sizes.

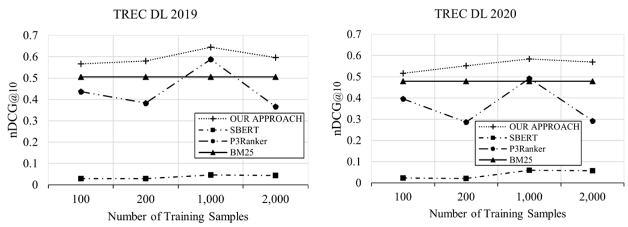

Figure 2: Comparison *w.* neural, few-shot & lexical baselines.

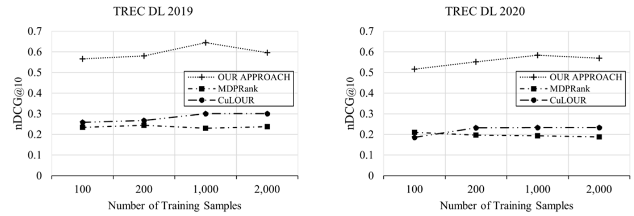

Figure 3: Benchmarking with state-of-the-art RL baselines.

approach is able to show better performance compared to strong RL-based rankers.

Dataset. We conduct experiments on the MS MARCO v1 dataset [41], which contains 8.8 million passages. For training, we randomly sample 2,000 queries (*a small set to replicate a few shot scenario*) from the 501k queries with relevance judgments. For evaluation, we use the TREC Deep Learning Track (DL-2019, DL-2020), which features more challenging queries and richer relevance labels.

Implementation Details.[1]**.** We trained our model, on a 9-layer FFN, with the learning rate of 0.001, a discount factor of 0.99, a batch size of 1, a replay buffer size of 10,000, and 100,000 episodes.

Findings. In **(RQ1)**, we investigate the generalizability and stability of our proposed approach. For the sake of *generalizability*, we report the performance of our proposed approach when applied on different language models, namely RoBERTa [42], ELECTRA [43], DeBERTa [44], and ALBERT [45]. These models are used in their original pre-trained format without any further fine-tuning for the ranking task. As shown in Figure 1, our proposed approach shows similar performance on both TREC DL 2019 and TREC 2020 regardless of the language model that is used for its training. Furthermore, in order to assess the *stability*, we train our proposed model on all four language models using four different train set sizes including 100, 200, 1000, and 2000 training samples. The results can again be seen in Figure 1. As seen in the figure, model performances are enhanced as the size of the training set increases from 100 samples to 2,000 samples by approximately 10%. The increase in performance is smooth for all models on both datasets. We also note that regardless of the test set and the language model, all models perform quite strongly even when trained on 100 samples and exhibit stable performance as train set size increases.

In the second research question (**RQ2**), we compare our approach against a state-of-the-art SBERT neural ranking baseline using a cross-encoder architecture [46], as well as the state-of-the-art few-shot neural ranker [13], and the lexical-based BM25 baseline, which requires no training. In few-shot settings, lexical models like BM25 are often preferred for their strong out-of-the-box performance.

[1]Our code and data is available on GitHub: https://shorturl.at/w4OEv

Based on findings from RQ1, we report results only for the DeBERTa model due to space constraints. Figure 2 compares our model with SBERT, P^3*Rank*, and BM25. BM25 remains unaffected by training size, achieving nDCG@10 scores of 0.505 and 0.479 on TREC DL 2019 and DL 2020, respectively. The key finding in RQ2 is that SBERT fails to learn effectively from limited samples, maintaining nDCG@10 below 0.1 across all training sizes, even with 2,000 training samples. Our model consistently outperforms P^3*Rank* (*the state of the art few-shot ranker*) and scales effectively with increasing training samples, unlike SBERT and P^3*Rank*. It also achieves consistently higher performance than the lexical BM25 baseline.

In **RQ3**, we compare our approach against two state-of-the-art RL-based ranking models: MDPRank [18] and CUOLR [27]. This research question examines (1) whether the efficiency of our RL-based method in learning from limited samples extends to other RL baselines, and (2) whether our approach is more sample-efficient due to its architectural design. Figure 3 compares our approach with MDPRank and CUOLR on both test sets, leading to three key observations. *(i)* Both MDPRank and CUOLR outperform neural rankers like SBERT in low-resource settings, consistently achieving nDCG@10 above 0.2, whereas SBERT remains below 0.05 under similar conditions. This highlights the effectiveness of RL-based methods for few-shot learning. *(ii)* While more effective than neural rankers, MDPRank employs a policy gradient algorithm, which is sample inefficient due to noisy gradient estimates and high variance during training [26]. As a result, it performs worse than our approach, which is more sample-efficient. *(iii)* MDPRank plateaus in performance as training data increases, whereas our model continues improving with more training samples. *(iv)* Although CUOLR outperforms neural rankers, it relies on a soft actor-critic algorithm originally designed for continuous action spaces, making it inefficient for discrete action spaces [28]. Additionally, actor-critic methods depend on an on-policy critic, limiting their effectiveness compared to DQN-based models in discrete settings [28]. Consequently, CUOLR exhibits lower performance than our approach, which is significantly more efficient in practice.

4 CONCLUDING REMARKS

We propose a reinforcement learning (RL)-based re-ranking model to address data inefficiency in neural rankers for *few shot scenarios*. Built on a Deep Q-learning Network (DQN), our approach enhances sample efficiency through experience replay and optimized action selection via Q-value estimation. Extensive experiments show our model significantly outperforms both data-intensive, RL-based and strong few-shot ranking baselines achieving high effectiveness in NDCG while learning meaningful ranking policies from limited data.

REFERENCES

[1] Andrew Yates, Rodrigo Frassetto Nogueira, and Jimmy Lin. Pretrained transformers for text ranking: BERT and beyond. In Fernando Diaz, Chirag Shah, Torsten Suel, Pablo Castells, Rosie Jones, and Tetsuya Sakai, editors, *SIGIR '21: The 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, Virtual Event, Canada, July 11-15, 2021*, pages 2666–2668. ACM, 2021.

[2] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018.

[3] Navid Rekabsaz and Markus Schedl. Do neural ranking models intensify gender bias? In Jimmy X. Huang, Yi Chang, Xueqi Cheng, Jaap Kamps, Vanessa Murdock, Ji-Rong Wen, and Yiqun Liu, editors, *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, SIGIR 2020, Virtual Event, China, July 25-30, 2020*, pages 2065–2068. ACM, 2020.

[4] Stephen E. Robertson and Hugo Zaragoza. The probabilistic relevance framework: BM25 and beyond. *Found. Trends Inf. Retr.*, 3(4):333–389, 2009.

[5] Faezeh Ensan and Ebrahim Bagheri. Document retrieval model through semantic linking. In Maarten de Rijke, Milad Shokouhi, Andrew Tomkins, and Min Zhang, editors, *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining, WSDM 2017, Cambridge, United Kingdom, February 6-10, 2017*, pages 181–190. ACM, 2017.

[6] Ebrahim Bagheri, Faezeh Ensan, and Feras N. Al-Obeidat. Neural word and entity embeddings for ad hoc retrieval. *Inf. Process. Manag.*, 54(4):657–673, 2018.

[7] Bhaskar Mitra, Fernando Diaz, and Nick Craswell. Learning to match using local and distributed representations of text for web search. In Rick Barrett, Rick Cummings, Eugene Agichtein, and Evgeniy Gabrilovich, editors, *Proceedings of the 26th International Conference on World Wide Web, WWW 2017, Perth, Australia, April 3-7, 2017*, pages 1291–1299. ACM, 2017.

[8] Michael J. Taylor, Hugo Zaragoza, Nick Craswell, Stephen Robertson, and Chris Burges. Optimisation methods for ranking functions with multiple parameters. In Philip S. Yu, Vassilis J. Tsotras, Edward A. Fox, and Bing Liu, editors, *Proceedings of the 2006 ACM CIKM International Conference on Information and Knowledge Management, Arlington, Virginia, USA, November 6-11, 2006*, pages 585–593. ACM, 2006.

[9] Xialei Liu, Joost van de Weijer, and Andrew D. Bagdanov. Exploiting unlabeled data in cnns by self-supervised learning to rank. *IEEE Trans. Pattern Anal. Mach. Intell.*, 41(8):1862–1878, 2019.

[10] Peng Xu, Xiaofei Ma, Ramesh Nallapati, and Bing Xiang. Passage ranking with weak supervsion. *CoRR*, abs/1905.05910, 2019.

[11] Si Sun, Yingzhuo Qian, Zhenghao Liu, Chenyan Xiong, Kaitao Zhang, Jie Bao, Zhiyuan Liu, and Paul Bennett. Few-shot text ranking with meta adapted synthetic weak supervision. In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli, editors, *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pages 5030–5043. Association for Computational Linguistics, 2021.

[12] Nilanjan Sinhababu, Andrew Parry, Debasis Ganguly, Debasis Samanta, and Pabitra Mitra. Few-shot prompting for pairwise ranking: An effective non-parametric retrieval model. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen, editors, *Findings of the Association for Computational Linguistics: EMNLP 2024, Miami, Florida, USA, November 12-16, 2024*, pages 12363–12377. Association for Computational Linguistics, 2024.

[13] Xiaomeng Hu, Shi Yu, Chenyan Xiong, Zhenghao Liu, Zhiyuan Liu, and Ge Yu. P3 ranker: Mitigating the gaps between pre-training and ranking fine-tuning with prompt-based learning and pre-finetuning. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '22, page 1956–1962. ACM, July 2022.

[14] Richard S Sutton, Andrew G Barto, et al. Reinforcement learning. *Journal of Cognitive Neuroscience*, 11(1):126–134, 1999.

[15] Scott Jeen, Tom Bewley, and Jonathan Cullen. Zero-shot reinforcement learning from low quality data. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[16] Max Schwarzer, Ankesh Anand, Rishab Goel, R. Devon Hjelm, Aaron C. Courville, and Philip Bachman. Data-efficient reinforcement learning with momentum predictive representations. *CoRR*, abs/2007.05929, 2020.

[17] Max Schwarzer, Nitarshan Rajkumar, Michael Noukhovitch, Ankesh Anand, Laurent Charlin, R. Devon Hjelm, Philip Bachman, and Aaron C. Courville. Pretraining representations for data-efficient reinforcement learning. *CoRR*, abs/2106.04799, 2021.

[18] Zheng Wei, Jun Xu, Yanyan Lan, Jiafeng Guo, and Xueqi Cheng. Reinforcement learning to rank with markov decision process. In Noriko Kando, Tetsuya Sakai, Hideo Joho, Hang Li, Arjen P. de Vries, and Ryen W. White, editors, *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, Shinjuku, Tokyo, Japan, August 7-11, 2017*, pages 945–948. ACM, 2017.

[19] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction.* The MIT Press, second edition, 2018.

[20] Yue Feng, Jun Xu, Yanyan Lan, Jiafeng Guo, Wei Zeng, and Xueqi Cheng. From greedy selection to exploratory decision-making: Diverse ranking with policy-value networks. In Kevyn Collins-Thompson, Qiaozhu Mei, Brian D. Davison, Yiqun Liu, and Emine Yilmaz, editors, *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, SIGIR 2018, Ann Arbor, MI, USA, July 08-12, 2018*, pages 125–134. ACM, 2018.

[21] Sicong Zhang, Jiyun Luo, and Hui Yang. A POMDP model for content-free document re-ranking. In Shlomo Geva, Andrew Trotman, Peter Bruza, Charles L. A. Clarke, and Kalervo Järvelin, editors, *The 37th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '14, Gold Coast , QLD, Australia - July 06 - 11, 2014*, pages 1139–1142. ACM, 2014.

[22] Long Xia, Jun Xu, Yanyan Lan, Jiafeng Guo, Wei Zeng, and Xueqi Cheng. Adapting markov decision process for search result diversification. In Noriko Kando, Tetsuya Sakai, Hideo Joho, Hang Li, Arjen P. de Vries, and Ryen W. White, editors, *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, Shinjuku, Tokyo, Japan, August 7-11, 2017*, pages 535–544. ACM, 2017.

[23] Wei Zeng, Jun Xu, Yanyan Lan, Jiafeng Guo, and Xueqi Cheng. Multi page search with reinforcement learning to rank. In Dawei Song, Tie-Yan Liu, Le Sun, Peter Bruza, Massimo Melucci, Fabrizio Sebastiani, and Grace Hui Yang, editors, *Proceedings of the 2018 ACM SIGIR International Conference on Theory of Information Retrieval, ICTIR 2018, Tianjin, China, September 14-17, 2018*, pages 175–178. ACM, 2018.

[24] Yujing Hu, Qing Da, Anxiang Zeng, Yang Yu, and Yinghui Xu. Reinforcement learning to rank in e-commerce search engine: Formalization, analysis, and application. In Yike Guo and Faisal Farooq, editors, *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018, London, UK, August 19-23, 2018*, pages 368–377. ACM, 2018.

[25] Lixin Zou, Long Xia, Zhuoye Ding, Jiaxing Song, Weidong Liu, and Dawei Yin. Reinforcement learning to optimize long-term user engagement in recommender systems. In Ankur Teredesai, Vipin Kumar, Ying Li, Rómer Rosales, Evimaria Terzi, and George Karypis, editors, *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2019, Anchorage, AK, USA, August 4-8, 2019*, pages 2810–2818. ACM, 2019.

[26] Ofir Nachum, Mohammad Norouzi, Kelvin Xu, and Dale Schuurmans. Bridging the gap between value and policy based reinforcement learning. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 2775–2785, 2017.

[27] Zeyu Zhang, Yi Su, Hui Yuan, Yiran Wu, Rishab Balasubramanian, Qingyun Wu, Huazheng Wang, and Mengdi Wang. Unified off-policy learning to rank: a reinforcement learning perspective. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine, editors, *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023.

[28] Denis Steckelmacher, Hélène Plisnier, Diederik M. Roijers, and Ann Nowé. Sample-efficient model-free reinforcement learning with off-policy critics. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2019, Würzburg, Germany, September 16–20, 2019, Proceedings, Part III*, page 19–34, Berlin, Heidelberg, 2019. Springer-Verlag.

[29] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin A. Riedmiller, Andreas Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nat.*, 518(7540):529–533, 2015.

[30] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin A. Riedmiller. Playing atari with deep reinforcement learning. *CoRR*, abs/1312.5602, 2013.

[31] Eduardo Pignatelli, Johan Ferret, Matthieu Geist, Thomas Mesnard, Hado van Hasselt, and Laura Toni. A survey of temporal credit assignment in deep reinforcement learning. *Trans. Mach. Learn. Res.*, 2024, 2024.

[32] Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming.* Wiley Series in Probability and Statistics. Wiley, 1994.

[33] Omid Madani. Polynomial value iteration algorithms for detrerminstic mdps. In Adnan Darwiche and Nir Friedman, editors, *UAI '02, Proceedings of the 18th Conference in Uncertainty in Artificial Intelligence, University of Alberta, Edmonton, Alberta, Canada, August 1-4, 2002*, pages 311–318. Morgan Kaufmann, 2002.

[34] Stefanos Doltsinis, Pedro Ferreira, and Niels Lohse. An MDP model-based reinforcement learning approach for production station ramp-up optimization: Q-learning analysis. *IEEE Trans. Syst. Man Cybern. Syst.*, 44(9):1125–1138, 2014.

[35] Anirudh Goyal, Abram L. Friesen, Andrea Banino, Theophane Weber, Nan Rosemary Ke, Adrià Puigdomènech Badia, Arthur Guez, Mehdi Mirza, Peter Conway

Humphreys, Ksenia Konyushkova, Michal Valko, Simon Osindero, Timothy P. Lillicrap, Nicolas Heess, and Charles Blundell. Retrieval-augmented reinforcement learning. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvári, Gang Niu, and Sivan Sabato, editors, *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pages 7740–7765. PMLR, 2022.

[36] Jesse Clifton and Eric Laber. Q-learning: Theory and applications. *Annual Review of Statistics and Its Application*, 7(1):279–301, March 2020.

[37] Ziyu Wang, Victor Bapst, Nicolas Heess, Volodymyr Mnih, Rémi Munos, Koray Kavukcuoglu, and Nando de Freitas. Sample efficient actor-critic with experience replay. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

[38] David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy P. Lillicrap, and Gregory Wayne. Experience replay for continual learning. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 348–358, 2019.

[39] Guangyuan Shi, Jiaxin Chen, Wenlong Zhang, Li-Ming Zhan, and Xiao-Ming Wu. Overcoming catastrophic forgetting in incremental few-shot learning by finding flat minima. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 6747–6761, 2021.

[40] Long Ji Lin. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Mach. Learn.*, 8:293–321, 1992.

[41] Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. MS MARCO: A human generated machine reading comprehension dataset. In Tarek Richard Besold, Antoine Bordes, Artur S. d'Avila Garcez, and Greg Wayne, editors, *Proceedings of the Workshop on Cognitive Computation: Integrating neural and symbolic approaches 2016 co-located with the 30th Annual Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain, December 9, 2016*, volume 1773 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2016.

[42] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692, 2019.

[43] Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. ELECTRA: pre-training text encoders as discriminators rather than generators. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

[44] Pengcheng He, Jianfeng Gao, and Weizhu Chen. Debertav3: Improving deberta using electra-style pre-training with gradient-disentangled embedding sharing. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023.

[45] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. ALBERT: A lite BERT for self-supervised learning of language representations. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

[46] Nils Reimers and Iryna Gurevych. Making monolingual sentence embeddings multilingual using knowledge distillation. In Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu, editors, *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 4512–4525. Association for Computational Linguistics, 2020.