Corpus-Level End-to-End Exploration for Interactive Systems

Zhiwen Tang, Grace Hui Yang
InfoSense, Department of Computer Science
Georgetown University
zt79@georgetown.edu, huiyang@cs.georgetown.edu

Abstract

A core interest in building Artificial Intelligence (AI) agents is to let them interact with and assist humans. One example is Dynamic Search (DS), which models the process that a human works with a search engine agent to accomplish a complex and goal-oriented task. Early DS agents using Reinforcement Learning (RL) have only achieved limited success for (1) their lack of direct control over which documents to return and (2) the difficulty to recover from wrong search trajectories. In this paper, we present a novel corpus-level end-to-end exploration (CE3) method to address these issues. In our method, an entire text corpus is compressed into a global low-dimensional representation, which enables the agent to gain access to the full state and action spaces, including the under-explored areas. We also propose a new form of retrieval function, whose linear approximation allows end-to-end manipulation of documents. Experiments on the Text REtrieval Conference (TREC) Dynamic Domain (DD) Track show that CE3 outperforms the state-of-the-art DS systems.

1 Introduction

Retrieval-based interactive systems, including multi-turn Question Answering (QA) (Reddy, Chen, and Manning 2019), dialogue systems (Sankar et al. 2019), and dynamic search systems (Yang, Sloan, and Wang 2016), study the interaction between a human user and an intelligent agent when they work together to accomplish a goal-oriented task. Reinforcement Learning (RL) becomes a natural solution to these interactive systems (Yang, Sloan, and Wang 2016; Li, Resnick, and Mei 2016; Hu et al. 2018) for its emphasis on adaptation and exploration. Prior work on this topic has investigated bandits-based (Li, Resnick, and Mei 2016), value-based (Luo, Zhang, and Yang 2014), and policy-based (Hu et al. 2018) RL methods. In these approaches, oftentimes, a repository of documents (or knowledge) and inputs from a human user, are treated as the learning environment for the AI agent; and the agent's actions usually take two steps – first, reformulating queries (or questions) based on user responses; second, retrieving relevant information to fulfill those queries via some off-the-shelf retrieval tools. Such a pipeline is a convenient use of existing Information Retrieval (IR) techniques; however, it comes with a few drawbacks.

First, most existing retrieval functions are optimized over precision at top ranks, which is demanded by the limited cognitive load a human user could afford when examining the results. This bias is engraved in all ready-to-use retrieval tools. The consequence is that results that are good but not as optimal would have little chance to show up. It might be ideal when there is only one run of retrieval, such as in single-turn QA or ad-hoc document retrieval. In multi-turn interactions, however, early triage of lower-ranked documents would lead to long-term loss that can not be easily recovered. Classic works on exploratory search (White and Roth 2009) and information seeking (Marchionini 2006) named this phenomenon "berry picking" – which depicts a tortuous search trajectory where only very limited useful information can be obtained at each single step because the search space is so restricted by the top results. This makes the RL agent's learning very challenging because the agent is not able to "explicitly consider the *whole* problem of a goal-oriented" process (Sutton and Barto 2018).

Second, widely-used retrieval functions, including TF-IDF (Salton and Buckley 1988) and BM25 (Robertson, Zaragoza, and others 2009), are non-differentiable. Common functions to reformulate queries (Huang and Efthimiadis 2009) are non-differentiable, too. These non-differentiable functions prevent a gradient-based RL method from updating its gradient through; thus a user's feedback would not have real control over which documents to return. Consequently, the retrieval results could look random. Nonetheless, these functions remain quite popular due to their readiness.

In this paper, using dynamic search as an illustrating example, we propose a different solution to retrieval-based interactive agents. In our corpus-level end-to-end exploration algorithm (CE3), at each time step, we compress a text corpus into a single global representation and used to support the agent's full exploration in the state and action spaces. In addition, we propose a novel differentiable retrieval function that allows an RL agent to directly manipulate documents. Experiments on the Text REtrieval Conference (TREC) Dynamic Domain 2017 Tracks demonstrate that our method significantly outperforms previous DS approaches. It also shows that our method is able to quickly adjust the search trajectories and recover from losses in early interactions. Given the fundamental issues it addresses, we believe CE3's success can be extended to other interactive AI systems if they access information using retrieval functions.

2 Related Work

The work closest to ours is perhaps KB-InfoBot (Dhingra et al. 2017). It is a dialogue system to find movies from a large movie knowledge base (KB). Similar to us, KB-InfoBot used a global representation for *all* movies in its database to represent the states. To do so, it estimated a global distribution over the entire set of movie entities, conditioned on user utterances. The distribution was fed into a deep neural network to learn the agent's action. Also similar to us, KB-InfoBot used a differentiable lookup function to support end-to-end manipulation of data entities. The two works differ in that their dialogue agent ran on a structured database and completed a task by iteratively filling the missing slots, while ours is for unstructured free text and accomplishes a task by iteratively retrieving documents that is relevant to the search task.

Knowing a global model that oversees the entire text collection has shown to be beneficial to retrieval in conventional IR research. For instance, Liu and Croft proposed to develop corpus-level clusters by K-means and then used them to smooth out multinomial language models. Wei and Croft also used Latent Dirichlet Allocation (LDA) to obtain global topic hierarchies to improve retrieval performance. In this paper, we encode the corpus and the user's search history for a global state representation.

Another related area to our work is dimension reduction. Many breakthroughs in neural models for Natural Language Processing (NLP) are built upon word2vec (Mikolov et al. 2013) and its derivation doc2vec (Le and Mikolov 2014). Doc2vec is able to transform a high-dimensional discrete text representation into low-dimensional continuous vectors. Unfortunately, however, doc2vec cannot solve a problem known as *crowding* (Cook et al. 2007). It refers to the situation where multiple high-dimensional data points are collapsed into one after dimension reduction and two data points belonging to different classes are then inseparable. In our case, each data point represents either a relevant or irrelevant document. We choose to use the t-Distributed Stochastic Neighbor Embedding (t-SNE) method (Maaten and Hinton 2008). It was used to support data visualization for high-dimensional images (Maaten and Hinton 2008), network parameters (Mnih et al. 2015) and word vectors (Li et al. 2016). By assuming a t-distribution for the post-reduction distribution, t-SNE provides more space to scatter data points that were supposed to be collapsed and the dimensions can be reduced from thousands to as low as 2 or 3. Our experiments (Section 5) shows that t-SNE outperforms doc2vec for us.

3 Dynamic Search Background

Dynamic search (DS) systems are multi-turn interactive agents that assist human users for goal-oriented information seeking (Yang, Sloan, and Wang 2016). DS shares similar traits with its sister AI applications such as task-oriented dialogue systems and multi-turn QA systems. These traits include (1) a goal-oriented task and (2) the interactions with a human user. They exhibit different forms of interactions, though. In DS, the form of interaction is querying and retrieving documents. In dialogue systems, it is generating natural language responses. In multi-turn QA, it is questioning and finding answers.

DS is backed up by a long line of prior research in information science, library science, and information retrieval (IR). It is originated from Information Seeking (IS) (Belkin and others 1993). Most IS research has focused on studying user behaviors (Daronnat et al. 2019). Luo, Zhang, and Yang simplified IS into DS by separating the modeling on the system side from that on the user side and emphasized on the first. In DS, a human user either becomes part of the environment (Luo, Dong, and Yang 2015) or another agent who also interacts with the environment.

RL offers natural solutions to DS. The RL agent, a.k.a. the search engine, observes state at time t ($\boldsymbol{s}_t$) from the environment (the user and the document collection) and takes actions (a_t) to retrieve documents to show to the user. An immediate reward r_t is encapsulated in the user's feedback, which expresses how much the retrieved documents would satisfy the user's informational need. The retrieved documents could also change the states and transition from the old state to a new one: $\boldsymbol{s}_t \rightarrow \boldsymbol{s}_{t+1}$. This process may continue for many iterations until the search task is accomplished or the user decides to abandon it. Model-based RL approaches, such as Markov Decision Processes (MDPs) and Partially Observable Markov Decision Processes (POMDPs), and policy-based RL approaches have been explored in the past (Yang, Sloan, and Wang 2016).

The Dynamic Domain (DD) Tracks held at the Text REtrieval Conference (TREC) from 2015 to 2017 (Yang, Tang, and Soboroff 2017) are campaigns that evaluate DS systems. In the DD Tracks, human users are replaced with a simulator to provide feedback to the DS agents. The simulator's feedback includes ratings to documents returned by the agent and which passages in those documents are relevant. It was created based on ground truth assessment done by third-party human annotators.

Dozens of teams participated in the DD Tracks. Their methods ranged from results diversification (Moraes, Santos, and Ziviani 2016), relevance feedback (Buccio and Melucci 2016), imitation learning (Xue et al. 2014), to the focus of this paper, reinforcement learning (Tang and Yang 2017; Aissa, Soulier, and Denoyer 2018).

4 The Approach

Most RL-based DS approaches can be summarized by a general formulation. It takes two steps. First, a new, temporary query is generated by the RL agent's policy π:

$$q' \leftarrow h(a_t, \sigma) \tag{1}$$

where $a_t = \pi(\boldsymbol{s}_t, \boldsymbol{\theta}_t)$ is the action generated by policy π for state $\boldsymbol{s}_t$ at time t and π is parameterized by $\boldsymbol{\theta}_t$. We call h the *query reformulation function*, which constructs the new query q' based on π and heuristics σ. Note that σ does not depend on t. h is a non-differentiable function.

Second, the newly formulated query q' is sent to

$$f(q', d_i, \phi) \tag{2}$$

to obtain documents that are, hopefully, relevant to q'. We call f the *retrieval function*. It is usually an existing retrieval method, such as BM25 or Language Modelling, which is non-differentiable. f returns a score to quantify the relevance between q' and (a document) d_i. f is parameterized by ϕ, which is a variable of the retrieval method and independent of t.

Overall, at each search iteration t, the RL agent assigns a ranking score to the i^{th} document d_i:

$$Score_{i,t} = f(h(\pi(\boldsymbol{s}_t, \boldsymbol{\theta}_t), \sigma), d_i, \phi) \tag{3}$$

and then all documents are ranked and retrieved based on this scoring.

The first issue of this formulation is that the retrieval function f only finds the top relevant documents for query q'. At any time, the agent is not aware of the global picture of the state space. This is different from how an RL agent is treated in AI, where the agent is always aware of the global status of the environment, e.g. AlphaGo knows the game board. This inadequate knowledge of the global picture hampers the RL agent to make longer-term plans for better decision-making.

The second issue in this formulation is that neither h nor f is differentiable. This prevents a gradient-based RL method from correctly updating its gradient. The RL agent is thus unable to effectively adjust the retrieval results based on user feedback. In addition, folding together multiple non-differentiable functions makes it very difficult to diagnose an failed action, which could result from a bad policy π, a sloppy σ, or an ineffective ϕ.

In this paper, we propose to convert and compress a collection of documents into a global representation. We also introduce a novel differentiable ranking function that can be easily incorporated into a gradient-based RL method (Schulman et al. 2017) to perform dynamic search.

Algorithm Overview

Our framework is based on a state-of-the-art Monte Carlo policy-gradient method, Proximal Policy Optimization (PPO) (Schulman et al. 2017). The RL agent consists of two networks, a value network and a policy network. Given the current state $\boldsymbol{s}_t$, the policy network outputs action $\boldsymbol{a}_t^{\boldsymbol{\theta}}$ and the value network outputs state-value $V(\boldsymbol{s}_t)$. Both networks are composed of a few layers of Convolutional Neural Networks (CNNs) and a Multi-Layer Perceptron (MLP). They share the same network structure but use different parameter settings. Figure 1 illustrates the system architecture.

Algorithm 1 describes the proposed CE3 method. It starts by sampling actions by the RL agent. The documents are then ranked by their estimated relevance scores calculated based on the action vectors. The top-ranked ones are shown

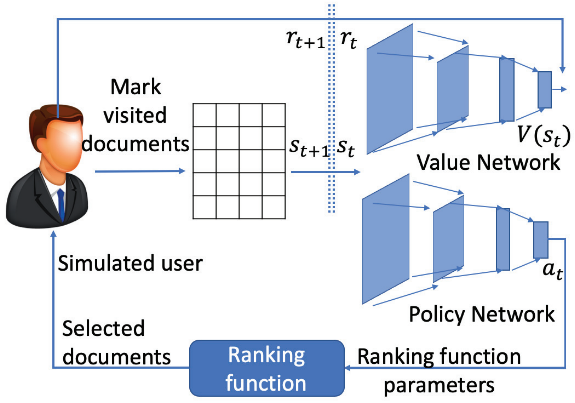

Figure 1: System architecture.

```
Initialize $\boldsymbol{\theta}$;
for iteration = 1, 2, ... do
    for t=1,2, ... , T do
        read in the global representation $\boldsymbol{s}_t$;
        sample action $\boldsymbol{a}_t^{\boldsymbol{\theta}} = \pi(\boldsymbol{s}_t, \boldsymbol{\theta}_t)$;
        for i=1,2, ... , C do
            estimate a relevance score for document $d_i$:
              $score_{i,t} = f(\boldsymbol{a}_t^{\boldsymbol{\theta}}, d_i)$ (Eq. 13)
        end
        rank the documents by $score_{*,t}$ and return the
          top-ranked documents $\mathcal{D}_t$;
        compute $r_t$ based on Eq. 14;
        mark returned documents as visited, generate
          next state $\boldsymbol{s}_{t+1}$;
        Compute advantage $\hat{A}_t$;
    end
    Optimize $L(\boldsymbol{\theta})$ (Eq. 4) w.r.t. $\boldsymbol{\theta}$;
end
```

Algorithm 1: CE3.

to the user. The user then examines the documents and submits feedback, which is used to derive the immediate reward r_t. As a Monte Carlo algorithm, CE3 generates search trajectories by sampling actions based on the current policy. Given enough trajectory samples, the following objective function is optimized w.r.t. the network parameter $\boldsymbol{\theta}$:

$$L(\boldsymbol{\theta}) = \hat{\mathbb{E}}_t[L_t^{Policy}(\boldsymbol{\theta}) - c_1 L_t^{Value}(\boldsymbol{\theta}) + c_2 S[\pi_{\boldsymbol{\theta}}](\boldsymbol{s}_t)] \tag{4}$$

where c_1 is the weight for the value network and c_2 is the coefficient for the policy's entropy, which encourages the agent to explore all areas in the action space.

Learning of the value network is done by minimizing L_t^{Value}. It is the mean squared error between the estimated state value $V_{\boldsymbol{\theta}}(\boldsymbol{s}_t)$ and the targeted state value V_t^{targ} in the sampled trajectories at time t:

$$L_t^{Value}(\boldsymbol{\theta}) = \hat{\mathbb{E}}_t[(V_{\boldsymbol{\theta}}(\boldsymbol{s}_t) - V_t^{targ})^2] \tag{5}$$

Learning of the policy network is done by minimizing L^{Policy}. It is a pessimistic bound of the effectiveness of the policy network at time t:

$$L_t^{Policy}(\boldsymbol{\theta}) = \hat{\mathbb{E}}_t[\min(\rho_t(\boldsymbol{\theta})\hat{A}_t, clip(\rho_t(\boldsymbol{\theta}), 1-\epsilon, 1+\epsilon)\hat{A}_t)] \quad (6)$$

where $\hat{A}_t$ is the advantage function (Sutton and Barto 2018).

Certain actions may increase the return in extreme situations but may not work in general. To avoid such situation, the algorithm adopts a surrogate clip function and discards actions when their rates-to-change are larger than ϵ:

$$clip(\rho_t(\boldsymbol{\theta}), 1-\epsilon, 1+\epsilon)\hat{A}_t = \begin{cases} \min(\rho_t(\boldsymbol{\theta}), 1+\epsilon)A_t & A_t > 0 \\ \max(\rho_t(\boldsymbol{\theta}), 1-\epsilon)A_t & A_t < 0 \end{cases} \quad (7)$$

where $\rho_t(\boldsymbol{\theta}) = \frac{\pi_{\boldsymbol{\theta}}(\boldsymbol{a}_t|\boldsymbol{s}_t)}{\pi_{\boldsymbol{\theta}_{old}}(\boldsymbol{a}_t|\boldsymbol{s}_t)}$ is the change rate of actions.

The algorithm employs stochastic gradient ascent (SGA) to optimize both the policy network and the value network. The process continues until no better policy is found.

This PPO-based method alone can be used in other applications. However, for the reasons motivating this paper, we think it should be used in combination with what we will present next – corpus-level document representation and differentiable ranking function.

Build a Global Representation

In this paper, we propose to compress an entire text corpus into a global low-dimensional representation and keep it at all time. Our goal is to enable a DS agent to always gain access to the full state space. We believe it is essential for a DS agent because not being able to reach documents in under-explored areas would mean not being able to recover from early bad decisions.

We summarize the procedure of creating global representation into three steps. First, each document is split into topic-coherent segments. The latest advances in Neural Information Retrieval (NeuIR) have demonstrated the effectiveness of using topical structures for NeuIR (Tang and Yang 2019; Fan et al. 2018). In this work, we follow (Tang and Yang 2019) for segmenting and standardizing documents. Each document is split into a fixed B number of segments (B is empirically set to 20). Within each segment, the content is expected to be topic-coherent since the segmentation is done based on Tilebars. Tilebars (Hearst 1995) is a classical text visualisation work and has been proven to be very effective in helping identify relevant documents by visualizing the term matches.

Second, bag-of-Words (BoW) is used as the feature vector for a segment and is of a size equal to the vocabulary's size W. This dimension is usually quite high and could easily reach millions in natural language tasks. Therefore, we compress each segment into a much manageable lower-dimension n ($n \ll W$). One challenge is that after the compression the relevant and irrelevant documents would be crowed together and difficult to be separated apart. To address this issue, We employ t-SNE (Maaten and Hinton 2008) for dimension reduction. The idea is based on Barnes-Hut approximation (Barnes and Hut 1986). Assume the high-dimensional input $\boldsymbol{x}_* \in \mathbb{R}^W$ follows Gaussian distribution. The probability that two random data points $\boldsymbol{x}_i$ and

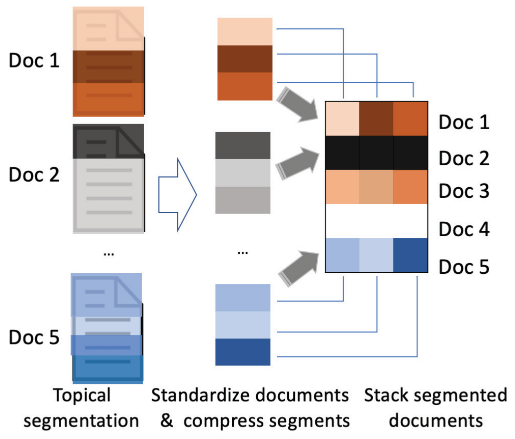
Doc 1
Doc 2
...
Doc 5
Topical segmentation
Standardize documents & compress segments
Stack segmented documents
Doc 1
Doc 2
Doc 3
Doc 4
Doc 5

Figure 2: Global representation of a toy corpus (of 5 documents): Documents are segmented and standardized following (Tang and Yang 2019). Similar colors suggest similar contents. Document 2 is darkened after being visited. Document 4 is currently selected by the RL agent and highlighted with white.

$\boldsymbol{x}_j$ are neighboring to each other is

$$p(\boldsymbol{x}_i, \boldsymbol{x}_j) = \frac{exp(-||\boldsymbol{x}_i - \boldsymbol{x}_j||^2/2\sigma^2)}{\sum_{k\neq l} exp(||\boldsymbol{x}_k - \boldsymbol{x}_l||^2/2\sigma^2)} \quad (8)$$

The algorithm then maps these data points in the high dimensional space to points $\boldsymbol{y}_*$ in a much lower dimensional space $\mathbb{R}^n$. Suppose $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ project into the lower dimension as $\boldsymbol{y}_i$ and $\boldsymbol{y}_j$. The probability that $\boldsymbol{y}_i$ and $\boldsymbol{y}_j$ are still neighboring to each other is

$$q(\boldsymbol{y}_i, \boldsymbol{y}_j) = \frac{(1 + ||\boldsymbol{y}_i - \boldsymbol{y}_j||^2)^{-1}}{\sum_{k\neq l}(1 + ||\boldsymbol{y}_k - \boldsymbol{y}_l||^2)^{-1}} \quad (9)$$

To establish the mapping between $\boldsymbol{x}$ and $\boldsymbol{y}$, the points' KL divergence

$$L_{tsne}(\boldsymbol{y}||\boldsymbol{x}) = \sum_i \sum_j p(\boldsymbol{x}_i, \boldsymbol{x}_j) \log \frac{p(\boldsymbol{x}_i, \boldsymbol{x}_j)}{q(\boldsymbol{y}_i, \boldsymbol{y}_j)} \quad (10)$$

is minimized. The solution to the new projection can be achieved step by step via gradient descent.

Third, segments from all documents are stacked together to form a global representation. The global representation is denoted by $\mathcal{C}$ and its dimensions are $C \times B \times n$. Here C is the number of documents, B is the number of segments per document, and n is the reduced feature dimension. In our work, n is empirically set to 3. In this global representation $\mathcal{C}$, each row represents a document and each column represents a segment at a certain position in the documents. Each row unfolds the segments horizontally, with their original order in a document preserved. For generality, we make no assumption about the stacking order of documents. The RL agent is expected to complete the search task even when dealing with randomly ordered documents. Figure 2 illustrates the global representation of a toy corpus.

This global representation constructs the states. Our state at time t, $\boldsymbol{s}_t$, has two parts, $\mathcal{C}$ and the retrieval history of documents from time 1 to $t-1$:

$$\boldsymbol{s}_t = S(\mathcal{C}, \mathcal{D}_1 \cup \mathcal{D}_2 \cup ...\mathcal{D}_i... \cup \mathcal{D}_{t-1}) \tag{11}$$

where $\mathcal{D}_i$ is the set of documents retrieved at time i.

In Algorithm 1, already-retrieved documents are marked as visited. In the global representation, it is done by assigning a reserved value to those documents' feature vectors. When the i^{th} document is visited, the feature vectors of all its segments, i.e. v_{i*}, are changed to the reserved value. Such change explicitly shows past search history and exploration status at the corpus level.

Retrieve using a Differentiable Ranking Function

It is crucial for an RL agent to employ a differentiable ranking function as its action so that it can perform end-to-end retrieval. Unfortunately, most existing DS approaches still use ranking functions that are non-differentiable.

The existing approaches' formulation is shown in Eq. 3 $Score_{i,t} = f(h(\pi(\boldsymbol{s}_t, \boldsymbol{\theta}_t), \sigma), d_i, \phi)$. It is is clearly a non-differentiable function. This prevents the RL agent from directly manipulating documents based on user feedback. We propose to omit query reformulation h completely, including its heuristic σ. Since our RL agent would not use any conventional retrieval model, their heuristic parameter ϕ is gone, too. The ranking function then becomes:

$$score_{i,t} = f(\pi(\boldsymbol{s}_t, \boldsymbol{\theta}_t), d_i) \tag{12}$$

We then focus on making f differentiable. It is achieved by using a linear formulation for f. In our formulation, f approximates a document d_i 's relevance as a linear function over the segments belonging to d_i:

$$score_{i,t} = f(\boldsymbol{a}_t^{\boldsymbol{\theta}}, d_i) = \sum_{j=1}^{B} \boldsymbol{y}_{ij} \cdot \boldsymbol{a}_t^{\boldsymbol{\theta}} \tag{13}$$

where $\boldsymbol{a}_t^{\boldsymbol{\theta}} = \pi(\boldsymbol{s}_t, \boldsymbol{\theta}_t)$ is the action that is generated by policy $\pi(\boldsymbol{s}_t, \boldsymbol{\theta}_t)$ and $\boldsymbol{y}_{ij}$ is the feature vector of the j^{th} segment in d_i after compression. The action vector $\boldsymbol{a}_t^{\boldsymbol{\theta}}$ can be sampled by the RL agent at each time step. When it gets updated, the new action $\boldsymbol{a}_{t+1}^{\boldsymbol{\theta}}$ allows the agent to retrieve and explore a different set of documents.

Get Reward based on User Feedback

In TREC DD (Yang, Tang, and Soboroff 2017), the relevance ratings are provided by the simulated user. We define the immediate reward as the accumulated relevance ratings (without discounting) for the retrieved documents. Duplicated results are excluded. The immediate reward is:

$$r_t = \sum_{d_i \in \mathcal{D}_t \setminus (\mathcal{D}_1 \cup \mathcal{D}_2 \cup ... \cup \mathcal{D}_{t-1})} rel(d_i) \tag{14}$$

where $rel(*)$ is a rating given by the simulator. The rating can be a positive number $rel \in (0, +\infty)$ for a relevant document ($d_i \in \mathcal{R}^+$), or 0 for an irrelevant document ($d_i \in \mathcal{R}^-$). The larger the rating, the better the retrieved document.

Topic (DD17-10)	Leaning Towers of Pisa Repairs
Subtopic 1 (id: 321)	Tourism impact of repairs/closing
Subtopic 2 (id: 319)	Repairs and plans
Subtopic 3 (id: 320)	Goals for future of the tower
Subtopic 4 (id: 318)	Closing of tower

Table 1: Example Search Topic.

5 Experiments

Experimental Settings

The Text REtrieval Conference (TREC) Dynamic Domain (DD) Tracks 2015 - 2017 (Yang, Tang, and Soboroff 2017) provides a standard testbed for DS. A simulated user[1] issues a starting query, and then provides feedback for all the subsequent runs of retrievals. The feedback includes graded document-level and passage-level relevance judgments in the scale of -1 to 4.

We experiment on the TREC DD 2017 Track for its judgements' completeness. TREC DD 2017 used LDC New York Times collections (Sandhaus 2008) as its corpus. The collection included more than 1.8 million news articles archived in the past 20 years. The Track released 60 search tasks created by human assessors. Each task consisted of multiple hierarchically-organized subtopics. The subtopics were not made available to the participating DS systems. Instead of post-submission pooling, the Track spent a great deal of efforts in obtaining a complete set of relevant passages before the evaluation started. These answers were used to generate feedback by the simulator. In total, 194 subtopics and 3,816 relevant documents were curated.

Table 1 shows an example DD search topic DD17-10. In this example, the search task is to find relevant information on " closing of Leaning Tower in Pisa". Table 2 shows an example interaction history.

Metrics The evaluation in DS focuses on gaining relevant information throughout the *whole* process. We adopt multiple metrics to evaluate the approaches from various perspectives. **Aspect recall** (Lagergren and Over 1998) measures subtopic coverage: $AsepctRecall = \frac{\text{\# subtopics found}}{\text{\# subtopics in the topic}}$. **Precision** and **Recall** measure the ratios of correctly retrieved documents over the retrieved document set or the entire correct set, respectively: $Precision = \frac{|(\cup_{i=1}^{n}\mathcal{D}_i)\cap\mathcal{R}^+|}{|\cup_{i=1}^{n}\mathcal{D}_i|}$, and $Recall = \frac{|(\cup_{i=1}^{n}\mathcal{D}_i)\cap\mathcal{R}^+|}{|\mathcal{R}^+|}$. **Normalized Session Discounted Cumulative Gain (nsDCG)** evaluates the graded relevance for a ranked document list, putting heavier weights on the early retrieved ones (Järvelin et al. 2008): $sDCG = \sum_{i=1}^{n} \sum_{d_j \in \mathcal{D}_i} rel(d_j) \left((1+\log_b j)(1+\log_{bq} i)\right)^{-1}$, and $nsDCG = \frac{sDCG}{ideal\ sDCG}$.

Systems We compare CE3 to the most recent DS systems. They were from the TREC DD 2017 submissions. We pick

[1]https://github.com/trec-dd/trec-dd-jig

	Search DD17-10
User:	*Leaning Towers of Pisa Repairs*
System:	Return document 0290537
User:	Non-relevant document.
System:	Return document 0298897
User:	Relevant on subtopic 320 with a rating of 2, *"No one doubts that it will collapse one day unless preventive measures are taken."*
System:	Return document 0984009
User:	Relevant on subtopic 318 with a rating of 4, *"The 12th-century tower was closed to tourists in 1990 for fear it might topple."*

Table 2: Example Interaction History.

the top submitted run from each team to best represent their approach. The runs are:

Galago (Croft, Metzler, and Strohman 2009):This approach does not use any user feedback. Documents are repeatedly retrieved with the same query Q at each iteration by Galago. Documents appeared in previous iterations are removed from current iteration.

Deep Q-Network (DQN) (Tang and Yang 2017): A DQN-based algorithm that selects query reformulation actions such as adding terms and removing terms and uses Galago to retrieve the documents.

Relevance Feedback (RF) (Rogers and Oard 2017) : The query Q is used to first retrieve an initial set of documents using Indri.[2] Then the documents are re-ranked by their similarity to the user feedback in all previous iterations. It is a variant of the relevance feedback (RF) model (Robertson and Jones 1976).

Results Diversification (DIV) (Zhang et al. 2017): This approach expands queries based on previous user feedback. The documents retrieved with solr[3] are then re-ranked with the xQuAD result diversification algorithm (Santos et al. 2010).

CE3: The proposed method in this paper. For comparison, we also implement a variant, **CE3 (doc2vec)**, which uses doc2vec (Le and Mikolov 2014) to compress the feature vector for each segment. The embeddings are trained on more than 1.8 million documents. Other settings are identical between CE3 and CE3 (doc2vec).

Parameters We construct a collection for each search topic Q by mixing relevant documents and irrelevant documents at a ratio of 1:1 to simulate a common re-ranking scenario. The corpus size C ranges from tens to thousands. Among all the parameter combinations, the following configuration yields the best performance: The dimension of t-SNE's output n is set to 3. The number of segments per document B is set to 20. Coefficients c_1 and c_2 in Eq. 4 are 0.5 and 0, respectively. Both the policy and value networks have 2 layers of CNNs and 1 MLP. The first CNN consists of eight 2×2 kernels and the second consists of 16. The hidden layer of MLP consists of 32 units and is the same for both networks. The output layer of MLP has 3 units for the policy network and 1 for the value network.

[2]https://www.lemurproject.org/indri/

[3]http://lucene.apache.org/solr/

Results

From Figure 3, we observe that CE3 outperforms all others in recall (Fig. 3c) and aspect recall (Fig. 3c) at all time. It suggests that our RL agent is able to explore more areas in the state and action spaces than the rest. While other algorithms also manage to achieve a high aspect recall (> 0.9), they do not perform as well at recall. It shows that although traditional diversification methods can find a few relevant documents for each aspect, it is hard for them to continue the investigation on a visited aspect. This indicates their less effective exploration. Instead, CE3's ranking function enables end-to-end optimization, which allows the agent to effectively explore at all different directions in the state and action spaces. It thus works very well on recall-oriented measures.

CE3 performs very impressive in precision (Fig. 3b), too. As the search episode develops, all other approaches show declined precision; however, CE3 stays strong at all iterations. We think it is because other methods could not easily recover from early mistakes while CE3's global representation allows it to explore elsewhere for ne opportunities when a bad decision happens.

Moreover, even not specifically designed for rank-sensitive metrics, CE3 performs very well on nsDCG, too. Results (Fig. 3a) reveal that at the beginning CE3 does not score as high as other methods; however, at the end of the episode, CE3 largely outperforms the rest. We believe the initial successes of other methods are caused by that they are well-tuned to be ranking-sensitive, which is what existing retrieval functions address. However, they seem not to be able to adapt well when the number of interactions increases.

In addition, it comes to our attention that CE3 (doc2vec) is left far behind by CE3. We know that they only differ in their choices to dimension reduction. In a follow-up investigation, we discover that CE3 retrieves much less duplicated documents than CE3 (doc2vec) does. Table 3 reports $\frac{|\mathcal{D}_t \cap (\mathcal{D}_1 \cup \mathcal{D}_2 \cup \ldots \cup \mathcal{D}_{t-1})|}{|\mathcal{D}_t|}$, the percentage of duplicate documents being retrieved, for the two CE3 variants. We believe it is due to how they compress the feature vectors in a segment. Doc2vec makes no assumption about the data distribution after compression. Vectors trained by doc2vec are probably crowded together and yield more duplicated results. On the contrary, t-SNE helps CE3 separate relevant documents from irrelevant documents, which makes it contribution to CE's success.

Visualize the Exploration

We are interested in observing the dynamics during a DS process. Figure 4 illustrates the first 8 steps for a search task with 3 subtopics. Based on the ground truth, we arrange the relevant documents at the top and irrelevant documents at the bottom. Among the relevant documents, those belong to the same subtopic are grouped together and placed in the order of subtopics 1 to 3. The turquoise dotted lines are added to highlight where each subtopic's are. The white

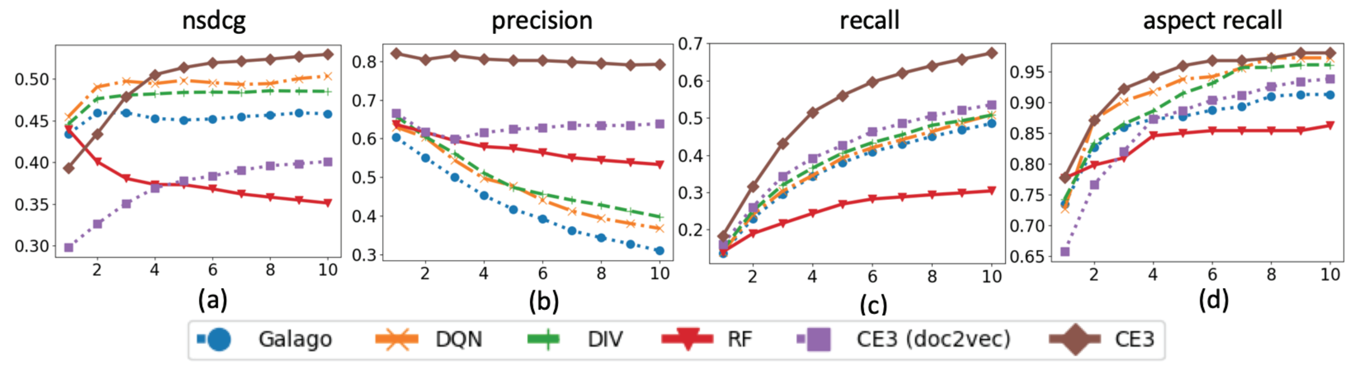

Figure 3: Experiment results in the first 10 search iterations.

Time step	t=1	t=2	t=3	t= 4	t=5	t=6	t=7	t=8	t=9	t=10
CE3 (doc2vec)	0.0%	11.7%	18.0%	30.0%	35.0%	34.3%	30.0%	25.0%	25.7%	25.0%
CE3	0.0%	3.0%	1.7%	1.0%	3.0%	0.0%	1.0%	0.3%	3.0%	0.0%

Table 3: Percentage of duplicate documents.

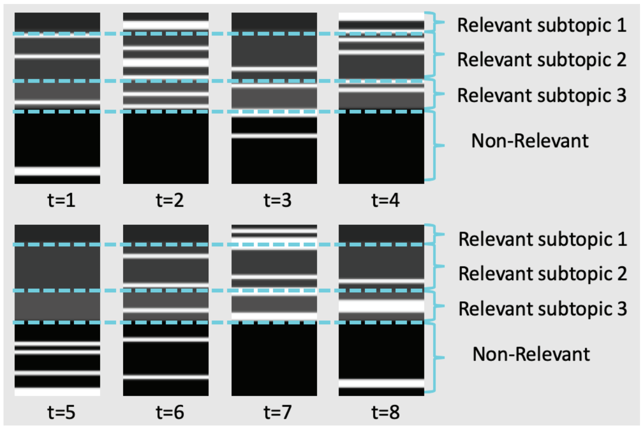

Figure 4: Visualization of Exploration (Topic DD17-3). White bars mark documents selected by the agent. Successful retrieval means more white in the top half.

color does not indicate relevance but show the visitations. It highlights which documents the agent returns at time t. A thicker white bar indicates more selected documents in the same subtopic. In the case a selected document is relevant to multiple subtopics, it is highlighted in multiple places. The effectiveness of the DS agent is jointly told by the white highlights and their positions: *successful retrieval means more white in the top half of this picture*.

We observe that at $t = 4$, the DS agent explores at subtopics 1 and 2; while at $t = 7$, it changes to subtopics 1 and 3. At the 5^{th} iteration, the agent seems to enter into a wrong path since the visualization shows that its current selection is in the lower irrelevant portion. However, the agent quickly corrects its actions and improves at $t = 7$ and $t = 8$. It confirms that CE3 is able to recover from bad actions.

6 Conclusion

Using Dynamic Search (DS) as an illustrating example, this paper presents a new deep reinforcement learning framework for retrieval-based interactive AI systems. To allow an agent to explore a space fully and freely, we propose to maintain a global representation of the entire corpus at all time. We achieve corpus-level compression by t-SNE dimension reduction. We also propose a novel differentiable ranking function to ensure user feedback can truly control what documents to return. The experimental results demonstrate that our method's performance is superior to state-of-the-art DS systems. Given the fundamental issues we address in this paper, we believe CE3's success can be extended to other interactive AI systems.

Acknowledgements

This research was supported by U.S. National Science Foundation IIS-145374. Any opinions, findings, conclusions, or recommendations expressed in this paper are of the authors, and do not necessarily reflect those of the sponsor.

References

Aissa, W.; Soulier, L.; and Denoyer, L. 2018. A reinforcement learning-driven translation model for search-oriented conversational systems. In *The 2nd International Workshop on Search-Oriented Conversational AI, SCAI@EMNLP '18*.

Barnes, J., and Hut, P. 1986. A hierarchical o (n log n) force-calculation algorithm. *Nature* 324(6096):446.

Belkin, N. J., et al. 1993. Interaction with texts: Information retrieval as information seeking behavior. *Information Retrieval* 93:55–66.

Buccio, E. D., and Melucci, M. 2016. Evaluation of a feedback algorithm inspired by quantum detection for dynamic search tasks. In *TREC '16*.

Cook, J.; Sutskever, I.; Mnih, A.; and Hinton, G. E. 2007. Visualizing similarity data with a mixture of maps. In *AISTATS '07*.

Croft, W. B.; Metzler, D.; and Strohman, T. 2009. *Search Engines - Information Retrieval in Practice*. Pearson Education.

Daronnat, S.; Azzopardi, L.; Halvey, M.; and Dubiel, M. 2019. Human-agent collaborations: trust in negotiating control. In *CHI '19*.

Dhingra, B.; Li, L.; Li, X.; Gao, J.; Chen, Y.; Ahmed, F.; and Deng, L. 2017. Towards end-to-end reinforcement learning of dialogue agents for information access. In *ACL '17*.

Fan, Y.; Guo, J.; Lan, Y.; Xu, J.; Zhai, C.; and Cheng, X. 2018. Modeling diverse relevance patterns in ad-hoc retrieval. In *SIGIR '18*.

Hearst, M. A. 1995. Tilebars: visualization of term distribution information in full text information access. In *CHI '95*.

Hu, Y.; Da, Q.; Zeng, A.; Yu, Y.; and Xu, Y. 2018. Reinforcement learning to rank in e-commerce search engine: Formalization, analysis, and application. In *SIGKDD '18*.

Huang, J., and Efthimiadis, E. N. 2009. Analyzing and evaluating query reformulation strategies in web search logs. In *CIKM '09*.

Järvelin, K.; Price, S. L.; Delcambre, L. M. L.; and Nielsen, M. L. 2008. Discounted cumulated gain based evaluation of multiple-query IR sessions. In *ECIR '08*.

Lagergren, E., and Over, P. 1998. Comparing interactive information retrieval systems across sites: The TREC-6 interactive track matrix experiment. In *SIGIR '98*.

Le, Q. V., and Mikolov, T. 2014. Distributed representations of sentences and documents. In *ICML '14*.

Li, J.; Chen, X.; Hovy, E. H.; and Jurafsky, D. 2016. Visualizing and understanding neural models in NLP. In *NAACL '16*.

Li, C.; Resnick, P.; and Mei, Q. 2016. Multiple queries as bandit arms. In *CIKM '16*.

Liu, X., and Croft, W. B. 2004. Cluster-based retrieval using language models. In *SIGIR '04*.

Luo, J.; Dong, X.; and Yang, H. 2015. Session search by direct policy learning. In *ICTIR '15*.

Luo, J.; Zhang, S.; and Yang, H. 2014. Win-win search: dual-agent stochastic game in session search. In *SIGIR '14*.

Maaten, L. v. d., and Hinton, G. 2008. Visualizing data using t-sne. *JMLR* 9(Nov):2579–2605.

Marchionini, G. 2006. Exploratory search: From finding to understanding. *ACM Communications* 49(4):41–46.

Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G. S.; and Dean, J. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS '13*.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M.; Fidjeland, A. K.; Ostrovski, G.; et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518(7540):529.

Moraes, F.; Santos, R. L. T.; and Ziviani, N. 2016. UFMG at the TREC 2016 dynamic domain track. In *TREC '16*.

Reddy, S.; Chen, D.; and Manning, C. D. 2019. Coqa: A conversational question answering challenge. *TACL* 7:249–266.

Robertson, S. E., and Jones, K. S. 1976. Relevance weighting of search terms. *JASIS* 27(3):129–146.

Robertson, S.; Zaragoza, H.; et al. 2009. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends® in Information Retrieval* 3(4):333–389.

Rogers, K., and Oard, D. W. 2017. Umd_clip: Using relevance feedback to find diverse documents for TREC dynamic domain 2017. In *TREC '17*.

Salton, G., and Buckley, C. 1988. Term-weighting approaches in automatic text retrieval. *Inf. Process. Manage.* 24(5):513–523.

Sandhaus, E. 2008. The new york times annotated corpus. *Linguistic Data Consortium, Philadelphia* 6(12):e26752.

Sankar, C.; Subramanian, S.; Pal, C.; Chandar, S.; and Bengio, Y. 2019. Do neural dialog systems use the conversation history effectively? an empirical study. In *ACL '19*.

Santos, R. L. T.; Peng, J.; Macdonald, C.; and Ounis, I. 2010. Explicit search result diversification through sub-queries. In *ECIR '10*.

Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; and Klimov, O. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Sutton, R. S., and Barto, A. G. 2018. *Reinforcement Learning: An Introduction*. The MIT Press, second edition.

Tang, Z., and Yang, G. H. 2017. A reinforcement learning approach for dynamic search. In *TREC '17*.

Tang, Z., and Yang, G. H. 2019. Deeptilebars: Visualizing term distribution for neural information retrieval. In *AAAI '19*.

Wei, X., and Croft, W. B. 2006. Lda-based document models for ad-hoc retrieval. In *SIGIR '06*.

White, R. W., and Roth, R. A. 2009. Exploratory search: Beyond the query-response paradigm. *Synthesis lectures on information concepts, retrieval, and services* 1(1):1–98.

Xue, Y.; Cui, G.; Yu, X.; Liu, Y.; and Cheng, X. 2014. ICTNET at session track TREC2014. In *TREC '14*.

Yang, G. H.; Sloan, M.; and Wang, J. 2016. Dynamic information retrieval modeling. *Synthesis Lectures on Information Concepts, Retrieval, and Services* 8(3):1–144.

Yang, G. H.; Tang, Z.; and Soboroff, I. 2017. TREC 2017 dynamic domain track overview. In *TREC '17*.

Zhang, W.; Hu, Y.; Jia, R.; Wang, X.; Zhang, L.; Feng, Y.; Yu, S.; Xue, Y.; Yu, X.; Liu, Y.; and Cheng, X. 2017. ICTNET at TREC 2017 dynamic domain track. In *TREC '17*.