Resource2Box: Learning To Rank Resources in Distributed Search Using Box Embedding

Ulugbek Ergashev*, Geon Lee†, Kijung Shin†, Eduard C. Dragut‡, Weiyi Meng*
*Binghamton University, Binghamton, USA
†KAIST, Seoul, Republic of Korea
‡Temple University, Philadelphia, USA
uergash1@binghamton.edu, geonlee0325@kaist.ac.kr, kijungs@kaist.ac.kr, edragut@temple.edu, meng@binghamton.edu

***Abstract*—The rapid and continuous growth of internet content poses significant challenges to conventional web search engines. Distributed Search (DS) offers a solution by integrating multiple information sources into a unified search system. When a user submits a query, the DS system selects relevant resources and ranks the documents within these selected resources. Recently, representation learning of queries and resources has been employed to enhance DS performance. However, existing methods that represent resources as vector embeddings may not sufficiently capture the semantic diversity within each resource.**

To address this limitation, we propose Resource2Box, a novel representation learning method for DS that models resources as boxes (i.e., hypercubes) in the latent space. Resource2Box more effectively captures the diverse and intricate information of documents within resources compared to single-point vector embeddings. It learns a box embedding for each resource, characterized by a center and offset, through two key processes: (1) aggregating document information within each resource using attentive pooling and (2) propagating information across resources. These box embeddings are learned to reflect the semantic relationships with training queries, utilizing a unique box-vector distance metric. Comprehensive experimentation on benchmark datasets demonstrates that Resource2Box significantly enhances resource selection, improving ranking performance by up to 24.7% across various metrics.

***Index Terms*—Distributed Search, Federated Search, Learning to Rank, Representation Learning, Box Embedding**

I. Introduction

The World Wide Web is expanding with a diverse and scattered assortment of data. Traditional web search engines and centralized information retrieval systems may not be able to keep up with the rapid changes and additions to web content [1], [2], leading to decreased retrieval quality and negatively impacting user experience.

Distributed Search (DS) [3], also referred to as Federated Search [4]–[7], has emerged as a promising solution to this challenge. DS systems integrate multiple distributed information sources, or *resources*, each of which organizes relevant *documents*, into a unified search system. When a user submits a query, instead of directly ranking a vast number of documents, a DS system selects a subset of resources most likely to meet the user's needs. Each selected resource then ranks its relevant documents [8]. These ranked lists are then combined into a single, unified ranking, which is presented to the user [9]–[11]. As a result, the DS system enhances the quality of search results by selectively ranking documents within the specialized resources.

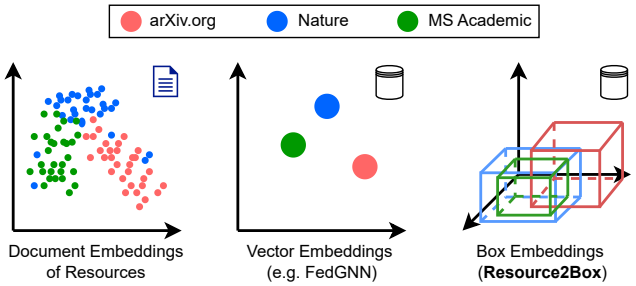


Fig. 1: Acquiring informative and representative resource embeddings is essential for accurate ranking algorithms. Our proposed method, Resource2Box, represents three resources in the Academic domain in FedWeb14 as boxes to effectively capture the information from the extensive and diverse set of documents within each resource.

In light of the need for effective DS, various methods have been proposed for ranking resources and selecting relevant ones for users' queries. In recent years, with advancements in deep learning, a popular approach for DS has been the representation learning of queries and items (i.e., documents and/or resources) as embeddings in a low-dimensional latent space. In the latent space, resources can be ranked according to their proximity to query embeddings. *This paper focuses on resource representation and selection in DS.*

Most existing representation learning methods for DS represent resources as *vector embeddings* [12]. However, a vector embedding, which is essentially a single point in the embedding space, may be suboptimal for capturing the semantic diversity and uncertainty of documents within each resource. For example, the ACM Digital Library [1] contains about 560,000 full-text articles (i.e., documents) spanning various subjects (i.e., resources) including data mining, information retrieval, data mining, and computer vision. Each subject (resource) encompasses numerous articles (documents) that, while relevant, can be very diverse and complex. For example, the data mining subject can include articles on clustering, anomaly detection, and pattern mining. Due to the semantic diversity within each resource, representing them as a single

[1]https://libraries.acm.org/digital-library

point in the latent space may not be sufficient to capture the full range of information and context, as shown in Figure 1.

Thus, a richer representation method for resources is needed. To this end, we propose Resource2Box, a novel representation method of resources via hypercubes (i.e., boxes), as shown in Figure 1. Unlike vector embeddings, which represent a single point in the latent space, boxes represent a bounded range in the latent space. Specifically, while vectors capture only the *position* of the embedding in the latent space, boxes also capture the *range* of the embedding in each dimension. This approach is geometrically more expressive, allowing it to more effectively capture the diverse and intricate information of documents contained in resources. While box embeddings have received significant research attention in ranking and recommendation tasks [12]–[15], their application to resource ranking remains unexplored.

Specifically, Resource2Box begins with data preprocessing, which involves sampling representative documents from each resource and extracting features using a pre-trained language model. Each resource is then represented as a box in the latent space, characterized by a center and an offset, to effectively encapsulate the diverse information contained in the documents. To this end, we aggregate document features through attentive pooling (intra-resource) and refine them by incorporating resource-wise relationships using a graph neural network (inter-resource). The box embeddings are learned for each resource, and they are subsequently ranked based on the distance to the user's query vector using a novel box-vector distance metric. Resource2Box addresses the limitations of representing a resource with a single vector, as empirically demonstrated by our experiments, leading to more accurate and effective resource ranking.

In summary, the contributions of our work are:

- We propose a novel resource representation method in DS. To our knowledge, this is the first study to adapt boxes to more accurately capture and model diverse data in each resource in the latent space.
- We conduct extensive experiments on three real-world benchmark datasets. The empirical results (specifically, improving ranking performance up to 24.7%) demonstrate that our proposed approach outperforms state-of-the-art baselines. We also give a detailed analysis of box showing its added benefits to resource representation compared to single vector representation.

The rest of this paper is organized as follows. Section II discusses related work. Sections III and IV present preliminaries and the details of the proposed method, respectively. Section V describes the experimental results. The conclusion to this study and future work are given in Section VI.

II. Related Work

We give an overview of the related literature in this section.

A. Distributed Search

We group existing resource selection methods broadly into four groups: lexicon-based, sample-based, combination-based, and supervised methods [4], [16], [17].

Lexicon-based methods treat each resource as a bag of words, and many initial studies view each resource as a big document, meaning the confines of each document within a resource are combined to form one extensive document consisting only of a collection of words [3], [18]. When a user's query is received, the broker calculates the similarity of the query with the lexical statistics of each information resource and ranks them according to their relevance score [19]. However, these methods cannot capture semantics at both the word and sentence granularity, which diminishes the quality of the resource representation [20].

Sample-based methods aim to avoid the practice of merging document boundaries as seen in the big document method. Instead, they view each resource as a collection of documents, and the importance of a resource is assessed, in aggregate, based on the significance of its collection of documents. Many methods in this group, such as ReDDE [21], CRCS [22] and SUSHI [23], utilize various voting algorithms over documents to infer a ranking of the resources. The latest method in this category, KBCS [20], characterizes resources as a set of entities weighted according to context- and structure-based metrics, and orders resources based on the similarity between the query and the resource. Lexicon-based and sample-based methods have been demonstrated to work well when resources are largely uniform [16], however, they are not fully capable of exploiting the structural attributes of text inputs.

Combination-based methods combine the approaches mentioned above. ECOMSVZ [24] combines a number of strategies to rank resources according to a given query, namely i) topical closeness between a query and resources, ii) estimation of resource popularity from online sources, and iii) query expansion. The method presented in [25] aggregates all three strategies proposed in [24] and ranks resources by giving more weight to small (specialized) resources.

Supervised methods employ machine learning approaches to build a model to tackle the resource selection problem. There are three types of supervised methods: query classification, resource classification, and learning to rank methods. Dai et al. [26] presented SVMrank model where they learn a ranking function based on three different sets of features, query-independent, term-based, and sampled-documents. Another method is LTRRS [27], which combines all the features proposed in [26] and topic relevance feature. They used LambdaMART model which directly optimizes nDCG metric to rank resources. The most recent method, FedGNN [28], which is a graph neural network (GNN) based approach to learning-to-rank that models resource-query and resource-resource relationships. FedGNN leverages pre-trained language model and GNN to extract both semantic and structural relationships of resources, and rank them for the given query. The primary disadvantages of these types of methods are that they either require extensive feature engineering or resources are represented by a single vector embedding which is less capable of capturing the diversity of documents in a resource in the latent space [29].

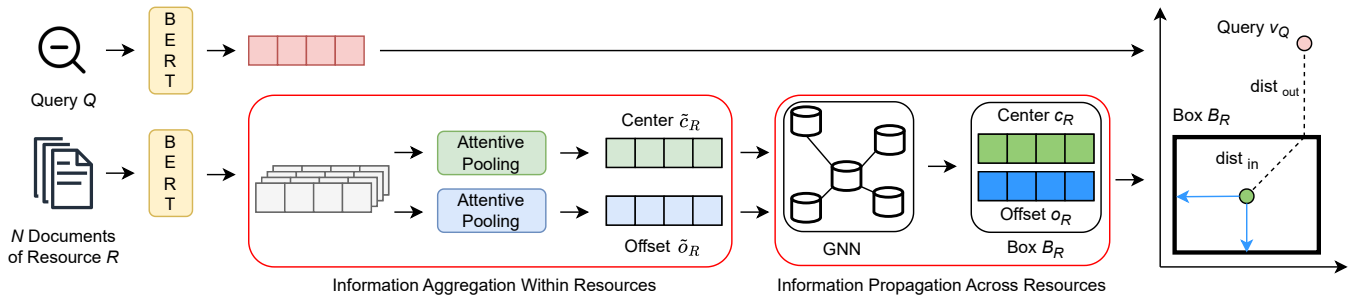


Fig. 2: The overall architecture of Resource2Box. Top N documents, with the highest relevance scores per each resource, and query are fed into pre-trained large language model BERT. Then, attentive pooling is used to aggregate document embeddings to obtain intermediate box centers and offsets for the resource. Finally, after employing GNN to capture the dependencies between resources, we generate final resource box embeddings. This gives us a richer resource representation, which in turn allows us to better identify the relevant resources for a query.

It is essential to distinguish our work within the Federated Search paradigm from other forms of distributed tasks. Federated Search, also known as Distributed Information Retrieval (DIR), focuses on integrating and querying multiple, heterogeneous information sources without centralized access to their entire content. This is distinct from other domains like Distributed Computing [30] and Distributed Databases [31], which often involve distributed processing and storage with a unified control. Our work contributes to the Federated Search field by proposing Resource2Box, a method that specifically addresses the weakness of single vector embedding representations of resources by using boxes to better capture the semantic diversity and uncertainty within resources.

B. Box Embeddings

Box embeddings provide effective abstractions for capturing complex, higher-order information within the data. Their expressiveness has supported the success of applications across diverse domains, such as in knowledge base [32]–[38] and recommender systems [13]–[15]. Moreover, box embeddings have been demonstrated to enhance accurate representation learning for words [39], images [40], and sets [41]. For example, Query2Box [32] embeds queries as boxes in the knowledge graph, including a set of answer entities within the box, enabling more diverse logical reasoning. In CubeRec [13], groups of users with multi-faceted preferences are represented as boxes. Set2Box [41] represents sets as boxes for accurate and versatile preservation of similarities across sets of varying sizes. These examples motivate our adoption of the box paradigm in our DS scenario, where resources consist of documents encompassing a diverse range of topics. Specifically, our approach involves representing resources as boxes, facilitating a more accurate representation not only of the resources themselves but also their relationships with user queries.

III. Preliminaries & Problem Definition

In this section, we introduce the problem setting with notations and define the problem of learning to rank resources.

A. Problem Setting.

Consider a set $\mathcal{R} = \{R_1, \cdots, R_N\}$ of *resources* and a set $\mathcal{D} = \{D_1, \cdots, D_M\}$ of *documents* where $N \ll M$ holds. Each resource $R_i \in \mathcal{R}$ consists of an arbitrary number of documents, forming a non-empty subset of $\mathcal{D}$, i.e., $R_i \subseteq \mathcal{D}$. Notably, the same documents can be shared across resources. Given a user *query* Q, a predefined *query-document* relevance score $s(Q, D_k)$ is associated with each document $D_k \in \mathcal{D}$.

B. Problem Definition

Given a user query Q, the objective is to rank the resources in $\mathcal{R}$ in such a way that it prioritizes those containing a significant number of documents relevant to the query. Specifically, our goal is to estimate the relevance score (or the inverse/negative distance) between the query Q and each resource $R_i \in \mathcal{R}$. These relevance scores are then used to rank the resources. Subsequently, a few high-ranked resources are selected, and the documents within them are ranked to be presented to the user.

IV. Our Method

In this section, we present Resource2Box (Figure 2), our proposed method for learning to rank resources. We begin by detailing the preprocessing step applied to the dataset. Then, we explain how we represent resources in the latent space.

A. Data Preprocessing

As a preliminary stage, we preprocess the dataset to prepare it for training Resource2Box. This involves two substeps: (1) sample selection and (2) feature extraction.

*1) **Sample Selection**:* Given that each resource can contain thousands or even millions of documents, we sample a subset of representative documents from each resource for representation learning to enhance efficiency. Following prior work [28], for each resource $R \in \mathcal{R}$, we compute the maximum relevance score with respect to the training queries for each document it contains, i.e., $\max_{Q \in \mathcal{Q}_{\text{train}}} s(Q, D)$ for each document $D \in R$ where $\mathcal{Q}_{\text{train}}$ is the set of training queries. Then, we select the top-N unique documents from the resource R with the

highest computed scores, represented by a subset $\tilde{R} \subseteq R$ where $|\tilde{R}| = N$; we use $N = 100$ throughout this paper. It is important to note that while the subsets of documents are used for training, during inference, we search the entire collection of documents from the filtered resources.

*2) **Feature Extraction**:* We extract features from documents and queries to use as input for Resource2Box. For documents, we concatenate the title with the body text [42]. For queries, we use the query text directly. Each text (document or query) is tokenized into a sequence of tokens and fed into the pre-trained Sentence-BERT model [43]. We then compute the mean of the output token embeddings to form a single feature vector for the input text. Formally, we denote the text feature obtained for a document D or a query Q as $F_D \in \mathbb{R}^{d'}$ and $F_Q \in \mathbb{R}^{d'}$, respectively, where d' is the dimension of the output token embeddings.

B. Resources as Boxes

Now that we have prepared the data, we move our attention to representing resources within the latent space. Each preprocessed resource $\tilde{R}$ consists of a set of N documents, each potentially covering distinct aspects of information. Thus, it is essential to represent resources to effectively aggregate the information from the documents in the latent space.

While the common approach is to represent resources as *vectors*, this reduces the resource to a single point in the latent space, which can inadequately represent the range of topics and context present in the documents. To address this limitation, we propose using *boxes* to represent resources, as they are geometrically more effective at capturing the diverse information within each resource. We will first discuss the concept of box embedding and then present our method for representing resources as boxes.

A *box* is a d-dimensional hyper-rectangle, characterized by its center (a vector) and an offset (also a vector) in the latent space. The center describes the location of the box, while the offset represents the length of each edge of the box in every dimension, defining its shape and size. Formally, for a given box $B = (c, o)$, where the center $c \in \mathbb{R}^d$ and offset $o \in \mathbb{R}^d_+$ both reside in the same latent space, this box is defined as a bounded region:

$$B \equiv \{p \in \mathbb{R}^d : c - o \preceq p \preceq c + o\}.$$

where p denotes any point within the box.

C. Resource2Box: Learning Box Embeddings for Resources

To derive a box embedding $B_R = (c_R, o_R)$ of resource R, we employ a two-step process. First, we aggregate the document information *within* each resource. Then, we enhance this aggregated embedding by considering relations *across* resources. Formally, this can be expressed as:

$$B_R = f_{\text{res}}\left(f_{\text{doc}}\left(\{D : D \in \tilde{R}\}\right), \mathcal{R}\right)$$

where f_{doc} is a function that aggregates document features within a resource, and f_{res} is a function that enhances the resource embedding by considering resource-level relations. This approach allows us to derive box embeddings that capture both the internal document-level information within a resource and the relational context among resources.

*1) **Information Aggregation Within Resources**:* We first summarize each resource by aggregating the documents within it. Instead of employing simple pooling functions like mean or max, we use attentive pooling, which allows us to highlight documents within each resource that are important for obtaining the center and offset of the box. Specifically, for a resource $\tilde{R}$ consisting of N sampled documents, let $F_{\tilde{R}}$ denote the embedding matrix consisting of the features of the N documents in $\tilde{R}$, i.e., $F_{\tilde{R}} = \|_{D \in \tilde{R}} F_D \in \mathbb{R}^{N \times d'}$, where $\|$ is the stack operation. We compute the intermediate center $\tilde{c}_R$ and offset $\tilde{o}_R$ as:

$$\tilde{c}_R = \sigma\left(\frac{(F_{\tilde{R}}\mathbf{W}^c_K) \cdot \mathbf{q}^c}{\sqrt{d}}\right) F_{\tilde{R}}, \quad \tilde{o}_R = \sigma\left(\frac{(F_{\tilde{R}}\mathbf{W}^o_K) \cdot \mathbf{q}^o}{\sqrt{d}}\right) F_{\tilde{R}},$$

respectively, where $\sigma(\cdot)$ is a Softmax function. Here, $\mathbf{W}^c_K, \mathbf{W}^o_K \in \mathbb{R}^{d' \times d}$ are the learnable key projection matrices for the center and offset, and $\mathbf{q}^c, \mathbf{q}^o \in \mathbb{R}^d$ are the learnable query vectors for the center and offset. The resulting intermediate center $\tilde{c}_R \in \mathbb{R}^{d'}$ and offset $\tilde{o}_R \in \mathbb{R}^{d'}$ are obtained by aggregating document features, each highlighted differently using their respective attention parameters. They are then utilized to construct the box embedding $B_R = (c_R, o_R)$ for the resource R, as described below.

*2) **Information Propagation Across Resources**:* While we have computed the intermediate center and offset by aggregating documents within each resource, it is important to note that resources often share similar documents, indicating inherent relationships between resources. Integrating these resource-resource relationships into our representation learning can potentially enhance the box embeddings for resources. To this end, we model the relationships between resources as a graph and leverage its structural information.

We construct a graph $G = (\mathcal{R}, \mathcal{E}, \omega)$, where $\mathcal{R}$ denotes the nodes corresponding to resources, $\mathcal{E}$ denotes the edges connecting these resources, and $\omega : \mathcal{E} \to \mathbb{R}$ is a function assigning weights to the edges. To construct a weighted edge between a pair $\{R_i, R_j\}$ of resources, we compute their weight (i.e., similarity) $\omega(R_i, R_j)$ as follows:

$$\omega(R_i, R_j) = \frac{\sum_{D \in R_i, D' \in R_j} \mathbb{1}[\cos(F_D, F_{D'}) > \tau]}{|R_i| \cdot |R_j|}$$

where $\cos(F_D, F_{D'})$ is the cosine similarity between F_D and $F_{D'}$, and τ is the threshold, which is a hyperparameter. That is, we measure the proportion of document pairs between the two resources that exhibit significant similarity. A positive $\omega(R_i, R_j)$ results in an edge between resources R_i and R_j with a weight $\omega(R_i, R_j)$; otherwise, we do not create an edge. Intuitively, a larger threshold τ results in fewer edges, leading to a sparser graph, and vice versa.

To effectively capture the dependencies and relationships between resources using the constructed graph G, we employ a Graph Neural Network (GNN). Specifically, we utilize

LightGCN [44] to refine the centers and offsets for resources. More precisely, at the ℓ-th propagation layer, the center and offset of resource R are updated as follows:

$$c_R^{(\ell+1)} = \sum_{R' \in \mathcal{N}_R} \frac{\omega(R, R')}{\sqrt{|\mathcal{N}_R| \cdot |\mathcal{N}_{R'}|}} c_{R'}^{(\ell)}, \quad o_R^{(\ell+1)} = \sum_{R' \in \mathcal{N}_R} \frac{\omega(R, R')}{\sqrt{|\mathcal{N}_R| \cdot |\mathcal{N}_{R'}|}} o_{R'}^{(\ell)}$$

respectively; $c_R^{(\ell)}$ and $o_R^{(\ell)}$ respectively denote the refined center and offset of the resource R after ℓ layers of propagation, and $\mathcal{N}_R$ denotes the neighboring resources, i.e., the set of resources that has an edge with resource R in G. The initial embeddings $c_R^{(0)}$ and $o_R^{(0)}$ are set to the intermediate embeddings $\tilde{c}_R$ and $\tilde{o}$, respectively. The center $\bar{c}_R$ and offset $\bar{o}_R$ are obtained by averaging those obtained at $\ell = 0, 1, \cdots, L$ steps:

$$\bar{c}_R = \frac{1}{L+1} \sum_{\ell=0}^{L} c_R^{(\ell)}, \quad \bar{o}_R = \frac{1}{L+1} \sum_{\ell=0}^{L} o_R^{(\ell)}.$$

*3) **Linear Projection**:* Finally, the resulting center $\bar{c}_R \in \mathbb{R}^{d'}$ and offset $\bar{o}_R \in \mathbb{R}^{d'}$ are passed through linear layers to be projected into the d-dimensional latent space:

$$c_R = \bar{c}_R \mathbf{W}_p^c + \mathbf{b}_p^c, \quad o_R = \max\left(\bar{o}_R \mathbf{W}_p^o + \mathbf{b}_p^o, 0\right)$$

where $\mathbf{W}_p^c, \mathbf{W}_p^o \in \mathbb{R}^{d' \times d}$ are the linear projection matrices, and $\mathbf{b}_p^c, \mathbf{b}_p^o \in \mathbb{R}^d$ are the intercept parameters, for the center and offset, respectively. Note that the offset o_R is constrained to be nonnegative. This completes the box embedding $B_R = (c_r, o_r)$ of the resource R. Similarly, we utilize the parameters $\mathbf{W}_p^c$ and $\mathbf{b}_p^c$ to project the feature $F_Q \in \mathbb{R}^{d'}$ of the query Q into the same latent space, i.e., $v_Q = F_Q \mathbf{W}_p^c + \mathbf{b}_p^c$.

D. Training Procedures

We now discuss the training procedure of Resource2Box. Ideally, given a user query Q, its embedding v_Q should be close to the embeddings of the resources containing relevant documents and far from those with irrelevant documents. To measure the *closeness* between the query vector and the resource box embeddings, we first introduce the distance metric used to measure the distance between vectors and boxes. Then, we discuss our objective function.

*1) **Distance Metric**:* Measuring distances between vectors and boxes differs from conventional distance metrics for vectors due to the unique geometric properties of boxes. Specifically, given a box $B = (c, o)$ and a vector v, we find the nearest point p on the boundary of the box to the vector, which can be expressed as [15]:

$$p = \min\left(c + o, \max(c - o, v)\right)$$

where min and max are the element-wise minimum and maximum functions, respectively. The *outside distance* $\text{dist}_{\text{out}}(v, B)$ and the *inside distance* $\text{dist}_{\text{in}}(v, B)$ are defined as:

$$\text{dist}_{\text{out}}(v, B) = \|p - v\|_2^2, \quad \text{dist}_{\text{in}}(v, B) = \|p - c\|_2^2,$$

respectively. The outside distance dist_{out} measures the distance between the vector and box in the outer region of the box, while dist_{in} measures the distance within the region of the box. Note that if the vector v is inside the box (i.e., $c - o \preceq v \preceq c + o$), then $p = v$, and thus the outside distance $\text{dist}_{\text{out}}(v, B)$ is zero, and the inside distance $\text{dist}_{\text{in}}(v, B)$ is the distance between v and the center c of the box. Finally, these two distances are combined to formulate the overall distance between the box and the vector:

$$\text{dist}(v, B) = \text{dist}_{\text{out}}(v, B) + \gamma \cdot \text{dist}_{\text{in}}(v, B)$$

where γ is a hyperparameter that controls the weights of the two distances. Using this distance metric allows us to model complex relationships between resources and queries, as illustrated in Figure 3. For instance, for two queries, Q_i and Q_j, with equal distances from the centers of resource boxes R_1 and R_2, their overall distances may be different after taking into account the offsets of the boxes and γ.

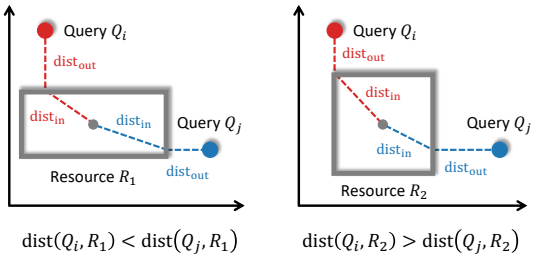


Fig. 3: Box embeddings for resources allow us to model complex relationships between resources and queries. Even when two queries, Q_i and Q_j, have the same distance from the centers of the box embeddings of resources R_1 and R_2, their distances may still be different, depending on the shapes and sizes of the respective boxes.

*2) **Loss Function**:* As the final step, we design our loss function for Resource2Box to rank resources. Specifically, we minimize the following distance-based hinge loss:

$$\mathcal{L} = \sum_{(Q, R^+, R^-) \in \mathcal{T}} \max\left(\text{dist}(v_Q, B_{R^+}) + \lambda - \text{dist}(v_Q, B_{R^-}), 0\right)$$

where $\mathcal{T}$ is the set of triplets consisting of a query, a positive resource, and a negative resource. In addition, λ is a margin that separates the two resources, which is a hyperparameter. We omit the L2 regularization term for brevity. Minimizing the above loss function encourages the query embeddings to be closer to relevant resources and farther from irrelevant ones.

To construct a training triplet of a given training query Q, we sample a *positive* resource $R^+ \in \mathcal{R}$ that contains an adequate number of relevant documents and a *negative* resource $R^- \in \mathcal{R}$ that lacks relevant documents. Specifically, we measure the relevance score $\hat{s}(Q, R)$ between the query Q and a resource R by summing the positive relevance scores of the documents in the resource and the query, i.e., $\hat{s}(Q, R) = \sum_{D_k \in R} \max(s(Q, D_k), 0)$ where $s(Q, D)$ for any $D \in \mathcal{D}$ is provided (see Section III-A). Positive resources are sampled with probability proportional to their relevance scores, while negative resources are uniformly sampled from those with a relevance score of zero.

E. Ranking Resources

Once Resource2Box is trained, we are prepared to rank the resources using the learned parameters. For a given test query Q', we compute its vector representation $v_{Q'}$ and measure its distance $\text{dist}(v_{Q'}, B_R)$ from the precomputed box embedding B_R of each resource $R \in \mathcal{R}$. Then, we rank the resources based on these distances, giving higher ranks to resources with box embeddings that are closer to the query embedding vector.

It is important to note that the resources are predefined, allowing their learned box embeddings to be stored without requiring additional computation each time a user submits a query. The process involves computing the embedding vector $v_{Q'}$ of the user's query Q' through a simple linear projection, followed by calculating the distance between this embedding and each resource's box embedding.

V. Experiments

In this section, we evaluate the effectiveness of our proposed approach for resource selection. We perform a validation of our approach and compare it with various baselines. We first describe baseline models and real-world public datasets that we employ for our extensive experiments. We then present experimental results with discussions and analysis.

A. Baselines

To evaluate the performance of our proposed model, we use existing methods from all four categories of resource selection approaches as baselines, including lexicon-based (Taily [19]), sample-based (Rank-S [45], ReDDE [21], CRCS [22] and KBCS [20]), combination-based (SSLTS [25] and ECOMSVZ [24]), and supervised (FedGNN [28], L2R [26] and Jnt [46]) methods. Notably, there are no existing methods in Federated Search that utilize box embeddings.

Taily [19] calculates the similarity of the query with the lexical statistics of each resource and ranks them according to their relevance score.
Rank-S [45] decays the scores of retrieved documents from the *centralized sample index* (CSI) exponentially, and treats as votes for the resources the documents were sampled from.
ReDDE [21] estimates the distribution of relevant documents across various resources by accounting for both content similarity and the size of the resource.
CRCS [22] is a sample-based method that, similar to ReDDE, runs a query on a CSI. However, in contrast to ReDDE, it ranks resources based on the positions of retrieved documents within the CSI ranking.
KBCS [20] uses a weighted entity set approach that models collections based on semantic relationships between entities and integrates DBpedia-based query expansion to enhance query-collection similarity metrics.
SSLTS [25] aggregates all the strategies proposed in [24] and ranks resources by assigning greater weight to smaller, specialized resources.
ECOMSVZ [24] combines SEIF, vertical selection, tf-idf, and semantic similarity features for resource selection, achieving top performance in the 2014 TREC FedWeb track.
FedGNN [28] is a recent supervised-based approach. It is a GNN-based approach to learning-to-rank that is capable of modeling resource-query and resource-resource relationships.
L2R [26] trains an SVMrank model where it learns a ranking function based on various sets of features, query-independent, term-based, and sampled-documents.
Jnt [46] uses a joint probabilistic classification model that estimates the probabilities of relevance in a joint manner by considering relationships among resources.

B. Comparison Methodology

For a fair comparison, we contrast our empirical results directly with those documented in prior literature, utilizing the same benchmark datasets and evaluation metrics. In particular, if an experimental result for a baseline B was reported using dataset D and evaluation metric M, our comparison with B also utilizes the same D and M. A clear advantage of this approach is that it eliminates the need for us to implement the baseline methods and at the same time avoid inaccuracies that often arise when implementing others' methods. A limitation of this approach is that we are not able to compare all baselines across different datasets using the same metrics. This is a trade-off that future work will need to address.

C. Evaluation Metrics

We will evaluate Resource2Box using the following metrics:
Precision at k or P@k. Precision is the ratio of the number of relevant retrieved items to the total number of retrieved items. P@10 is used to compare with baselines FedGNN [28], L2R [26], KBCS [20], ReDDE [21], CRCS [22], Rank-S [45], Taily [19] and Jnt [46] at document level.
Normalized precision at k or nP@k. This metric measures the relevance scores of the top-ranked k resources, normalized by the relevance scores of the best possible k resources for the given query. We use nP@$\{1,5\}$ to compare with baselines FedGNN [28], SSLTS [25] and ECOMSVZ [24] on the resource level evaluation.
Normalized Discounted Cumulated Gain at k or nDCG@k. This metric is a measure of ranking quality. We use nDCG@30 to compare with FedGNN [28], L2R [26], ReDDE [21], Rank-S [45], Taily [19] and Jnt [46] at document level.

Metrics such as P@k and nDCG@k are commonly used in the evaluation of ranking algorithms. Precision-focused metrics at lower rankings, like P@10 and nDCG@30, tend to outperform recall-oriented metrics at higher ranks like P@100 and nDCG@100, especially in resource selection problems where only a small fraction of resources are returned [16]. This is because users are typically interested in the top 10 results, and thus, highly ranked results are anticipated to meet user satisfaction directly.

D. Datasets

Our experiments are conducted using three widely used benchmark datasets, recognized as the latest publicly available

TABLE I: Dataset statistics

Dataset	# of resources	Total # of docs (K)	Resource (K)			Compared with baselines
			Min	Max	Avg	
GOV2	199	25,205	0.319	550.7	126.6	L2R, ReDDE, Rank-S, Taily, Jnt
ClueWeb09-B	100	50,220	63	1,691	502	FedGNN, KBCS, ReDDE, CRCS, Rank-S
ClueWeb09-B	123	50,220	32	734	408	FedGNN, L2R, ReDDE, Rank-S, Taily, Jnt
FedWeb14	149	187.7	.022	2.7	1.2	FedGNN, SSLTS, ECOMSVZ

TABLE II: Query set statistics

Query set	Dataset	# of queries	Folds	Compared with baselines
Web Track 2009	CW100	50	5	FedGNN, KBCS, ReDDE, CRCS, Rank-S
Terabyte Track 2004-2006	GOV2	150	10	L2R, ReDDE, Rank-S, Taily, Jnt
Web Track 2009-2012	CW123	200	10	FedGNN, L2R, ReDDE, Rank-S, Taily, Jnt
FedWeb14	FW14	50	5	FedGNN, SSLTS, ECOMSVZ

for Federated Search tasks. These datasets were also used to evaluate the baselines.

GOV2[2] is 25 million web pages crawled from the US government web domains. The collection is partitioned into 199 resources. We obtain 150 queries and relevance scores for document-query pairs from TREC 2004-2006 Terabyte Track[3].

ClueWeb09-B[4] is a portion of the larger web collection known as ClueWeb09. This dataset is developed to assist research in the fields of ranking tasks and similar language-based technologies, consisting of approximately 50 million English web pages. We partition the collection into two sets of resources: 100 and 123 resources. We refer to these two datasets as CW100 and CW123, respectively. For CW100 dataset, we obtain 50 queries and relevance scores for query-document pairs from TREC 2009 Web Track[3]. For CW123, 200 queries and query-document pair relevance scores are obtained from TREC 2009-2012 Web Track[3].

FedWeb Greatest Hits [47] is a large test collection designed to support research in web ranking tasks including resource selection in DS. The collection contains two datasets FedWeb13 and FedWeb14. We use the most recent, FedWeb14, dataset which consists of 149 resources and 50 queries. Relevance scores for query-document pairs are given in the dataset. We refer to this dataset as FW14 for the rest of the paper.

Table I summarizes the statistics of the datasets. The last column lists the baselines that are used to compare with our method using a given dataset.

Table II gives the query sets used to train and evaluate the proposed model. It also shows the cross-validation settings we followed for each query set. Our study adheres to the same cross-validation settings, including the number of folds, as specified in the baseline papers for each dataset, to ensure a fair and rigorous comparison. In cross-validation, we partition queries. For example, Web Track 2009-2012 has 200 queries. For ten-fold cross-validation, the 200 queries are partitioned into ten subsets; in each run, nine subsets are used for training and one subset is used for testing. The results are then averaged over the ten runs.

E. Implementation Details

To extract query and document features, we utilize a pre-trained Sentence-BERT[5] model that encodes input texts into 768-dimensional dense vectors. We implemented Resource2Box in PyTorch and trained it using the Adam optimizer. Each dataset is trained and evaluated following the cross-validation settings shown in Table II. For the other hyperparameters, we used a hidden dimension d of 512, a batch size of 256, and the number L of GNN layers of 2 by default. The learning rate was searched from $\{0.00001, 0.00005\}$, the regularization coefficient from $\{0.001, 0.0001, 0.00001\}$, the γ for the box-vector distance from $\{0.1, 0.5, 1.0, 5.0\}$, the margin λ from $\{0.5, 1.0, 5.0, 10.0\}$, and the threshold τ for graph construction from $\{0.1, 0.5, 0.9\}$. Experiments were conducted on a machine with RTX 8000 D6 GPUs where it takes less than 10 minutes to train on average. We followed the same search engine setting given in the baseline papers to simulate a distributed information retrieval environment. Once the resource selection algorithm picks top N highly scored resources, top-1000 documents are retrieved from the resources and merged directly.

F. Experimental Results

We conducted three sets of experiments to validate the effectiveness of Resource2Box. The first set compares our method with FedGNN and the other baselines L2R, Taily, Rank-S, ReDDE, and Jnt at document level (Table III). Note that L2R reports its empirical results for top-4 and top-8 resources for the CW123 dataset. For GOV2, it reports for top-6 and top-12 resources. We follow the same settings here. Table IV shows comparisons with FedGNN as well as the baselines KBCS, ReDDE, CRCS, and Rank-S, at document level. Baseline KBCS reports its results from 1 to 10 top resources on the CW100 dataset; we follow the same settings in our experiments as well. The final set of experiments reveals the results at the resource level and compares them with FedGNN and the baselines SSLTS and ECOMSVZ.

We observe that our approach consistently outperforms all baselines at document level, suggesting that modeling resources as boxes in latent space provides a representation that better captures the diverse data contained within a resource, which in turn improves resource selection. In particular, according to the results shown in Table III, Resource2Box outperforms the best baseline FedGNN on the CW123 dataset by 5.8% and by 3.7% on P@10 for top-4 and top-8 resources, respectively. For the same dataset and the same numbers of top resources, Resource2Box outperforms FedGNN by 0.7% and by 0.7% on nDCG@30, respectively. Table III also shows the results for GOV2 dataset, and our method outperforms

[2]https://ir.dcs.gla.ac.uk/test_collections/gov2-summary.htm
[3]https://trec.nist.gov/data/webmain.html
[4]http://www.lemurproject.org/clueweb09
[5]https://www.sbert.net/

TABLE III: P@10 and nDCG@30 metrics on CW123 and GOV2 datasets

Methods	CW123				GOV2			
	T=4		T=8		T=6		T=12	
	P@10	nDCG@30	P@10	nDCG@30	P@10	nDCG@30	P@10	nDCG@30
ReDDe	0.355	0.262	0.363	0.275	0.580	0.445	0.587	0.460
Rank-S	0.350	0.259	0.360	0.268	0.570	0.440	0.585	0.461
Taily	0.346	0.260	0.346	0.260	0.518	0.403	0.530	0.418
Jnt	0.370	0.269	0.367	0.277	0.582	0.459	0.588	0.465
L2R	0.374	0.281	0.377	0.286	0.593	0.469	0.591	**0.475**
FedGNN	0.398	0.282	0.407	0.286	0.595	0.469	0.592	**0.475**
Resource2Box	**0.421 ± 0.008**	**0.284 ± 0.005**	**0.422 ± 0.006**	**0.288 ± 0.001**	**0.602 ± 0.001**	**0.471 ± 0.002**	**0.598 ± 0.004**	**0.475 ± 0.004**

TABLE IV: P@10 comparison on CW100 dataset

T	Methods					
	ReDDe	CRCS	Rank-S	KBCS	FedGNN	**Resource2Box**
1	0.176	0.212	0.263	0.340	0.361	**0.435 ± 0.018**
2	0.236	0.212	0.272	0.368	0.382	**0.458 ± 0.015**
3	0.244	0.208	0.316	0.380	0.391	**0.466 ± 0.012**
4	0.244	0.236	0.316	0.370	0.418	**0.473 ± 0.016**
5	0.244	0.252	0.357	0.420	0.422	**0.479 ± 0.008**
6	0.252	0.252	0.365	0.400	0.429	**0.483 ± 0.009**
7	0.276	0.252	0.353	0.394	0.431	**0.484 ± 0.008**
8	0.300	0.260	0.348	0.422	0.430	**0.483 ± 0.009**
9	0.336	0.248	0.357	0.414	0.425	**0.482 ± 0.008**
10	0.356	0.232	0.361	0.420	0.421	**0.480 ± 0.008**

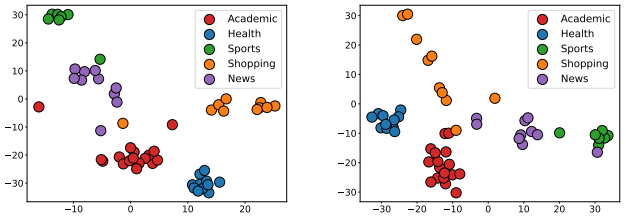
TABLE V: Search accuracy comparison on FW14 dataset

Methods	nP@1	nP@5
SSLTS	0.380	0.480
ECOMSVZ	0.535	0.604
FedGNN	0.760	**0.788**
Resource2Box	**0.809 ± 0.012**	0.786 ± 0.015

FedGNN by 1.2% and 1.0% on P@10 at top-6 and top-12, respectively. Table III also reveals that our method outperforms FedGNN for the same dataset by 0.4% on nDCG@30 at top-6. Table IV further demonstrates the superiority of our approach, showing that Resource2Box outperforms the best baselines, FedGNN and KBCS, from top-1 to top-10 resources on the CW100 dataset by 12.3% to 20.5%, respectively. On FW14, as reported in Table V, our method outperforms by 6.4% on nP@1 and slightly falls behind by 0.2% on nP@5 compared with FedGNN at the resource level comparison.

A contributing factor to the above noticeable enhancements is that modeling resources as boxes in the latent space possesses a better capability of capturing the semantic diversity of documents in each resource, thus ensuring a superior efficacy compared to the current baselines [12]. In contrast, FedGNN represents a resource as a single vector in the low-dimensional space, which is less capable of capturing the diversity of documents in each resource.

Another reason contributing to the performance improvement is that Resource2Box does not require a centralized sample index (CSI). Most of the baselines such as ReDDE, CRCS, and KBCS are designed to operate in uncooperative environments [22], where a broker has to send random queries to a resource on the Web to collect thousands of documents as a resource representation. CSI-based approaches depend heavily on this sampling process, consequently, both over-sampling and under-sampling may lead to biased resource representation [48]. Moreover, it is expensive to build a CSI [28]. Different from the sample-based methods, our approach



(a) Center (b) Offset

Fig. 4: Resource2Box learns meaningful box embeddings for resources in the 2D space, where both the centers and offsets are well-clustered based on the categories in the FW14 dataset.

is capable of representing a resource with fewer documents as a box embedding in the low-dimensional latent space.

*1) **Information Aggregation Within Resources**:* We begin by examining the effectiveness of attentive pooling for aggregating information from documents within each resource. To achieve this, we compare the performance of Resource2Box with two of its variants, Resource2Box$_{\text{Max Pooling}}$ and Resource2Box$_{\text{Mean Pooling}}$, which use max pooling and mean pooling of document features, respectively, to aggregate information. From Table VI, we can observe that Resource2Box consistently outperforms both Resource2Box$_{\text{Max Pooling}}$ and Resource2Box$_{\text{Mean Pooling}}$ by 0.2% to 19.5% on nP@1 on all datasets at the resource level. While at the document level, Resource2Box outperforms both variants by 6.5% to 18.6% on P@1 on CW100, CW123, and GOV2, confirming the effectiveness of attentive pooling applied in Resource2Box. This observation implies that by distinctively highlighting documents within each resource, we can enhance the aggregation of resource representations and thus resource retrieval [49].

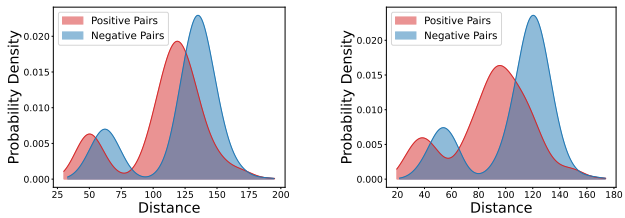
*2) **Information Propagation Across Resources**:* Table VI also demonstrates the superiority of Resource2Box by 0.5% to 24.7% on nP@1 on all datasets over Resource2Box$_{w/o\ \text{GNN}}$ at the resource level retrievals. In document-level retrievals, Resource2Box outperforms by 3.6% to 12% on P@1 on the CW100, CW123, and GOV2 datasets. The reason for the superiority of Resource2Box over the method without GNN is that some resources may share inherent similarities (i.e. identical or similar documents) between each other and GNN helps to integrate a structural message passing between resource nodes in the graph which leads to better ranking results [28]. This demonstrates the effectiveness of the GNN module, enabling Resource2Box to capture complex relationships between resources, leading to better box representations.

TABLE VI: Analysis of Resource2Box

Methods	Resource Level (nP@1)				Document Level (P@1)		
	FW14	CW100	CW123	GOV2	CW100	CW123	GOV2
Resource2Box$_{\text{Max Pooling}}$	0.778 ± 0.023	0.733 ± 0.027	0.630 ± 0.016	0.585 ± 0.023	0.475 ± 0.031	0.360 ± 0.024	0.564 ± 0.035
Resource2Box$_{\text{Mean Pooling}}$	0.806 ± 0.004	0.802 ± 0.038	0.706 ± 0.018	0.680 ± 0.006	0.482 ± 0.042	0.401 ± 0.017	0.609 ± 0.021
Resource2Box$_{w/o\ \text{GNN}}$	0.642 ± 0.015	0.881 ± 0.021	0.718 ± 0.019	0.682 ± 0.010	0.466 ± 0.035	0.412 ± 0.016	0.611 ± 0.026
Resource2Vec	0.755 ± 0.019	0.883 ± 0.025	0.672 ± 0.026	0.676 ± 0.009	0.470 ± 0.033	0.383 ± 0.013	0.609 ± 0.024
Resource2Box	**0.808 ± 0.011**	**0.885 ± 0.019**	**0.726 ± 0.008**	**0.699 ± 0.010**	**0.522 ± 0.034**	**0.427 ± 0.012**	**0.651 ± 0.024**

*3) **Box Compared to Vector Embedding**:* We observe in Table VI that Resource2Box outperforms Resource2Vec by 7%, 2.6%, 8% and 2.9% on nP@1 at the resource level on datasets FW14, CW100, CW123, and GOV2, respectively. At the document level, Resource2Box outperforms Resource2Vec by 11%, 11.5% and 6.9% on P@1 on the CW100, CW123 and GOV2 datasets, respectively. This indicates that boxes, which are geometrically richer than single vectors, can more effectively represent resources comprising documents of a wide range of topics, in the latent space.

We visually analyze this observation in Figure 4, where we utilize t-SNE [50] to project the centers and offsets of the box embeddings of the resources in the FW14 dataset into a 2D space. Notably, both the center (i.e., location of the box) and the offset (i.e., shape and size of the box) are well-clustered based on their respective resource categories. This implies that the learned box embeddings effectively capture the distinctive characteristics associated with each resource. We further investigate the reason behind the superiority of box embeddings over vector embeddings for resource representation. In Figure 5, we illustrate density functions for the distances between queries and their positive and negative resources. We compare the cases where resources are represented as vectors (Figure 5a) and boxes (Figure 5b) in the FW14 dataset. Visually, we can confirm that representing resources as boxes increases the distinction between positive and negative pairs, potentially contributing to more effective retrievals. Numerically, we compute the KL divergences between the two distributions to quantify how effectively positive and negative pairs are distinguished with respect to a query in the latent space. In particular, the KL divergences are computed as 2.425 and 3.794 for the vector embedding and box embedding, respectively, confirming our analysis.



(a) Vector Embedding (KL Divergence = 2.245)
(b) Box Embedding (KL Divergence = 3.794)

Fig. 5: The density functions of distances between query vector embeddings and (a) vector embeddings and (b) box embeddings of resources in FW14 dataset. The box embedding offers a superior differentiation between positive and negative pairs of queries and resources. This enhanced distinction contributes to more effective retrieval.

VI. Conclusion

To learn high-quality resource representations that effectively capture the diverse information within a resource, we move beyond conventional vector representations and propose a novel approach: learning box (hypercube) resource representations. Resource2Box represents resources as boxes in a low-dimensional space. Document information is aggregated through attentive pooling (intra-resource) and refined by incorporating resource-wise relationships using a graph neural network (inter-resource). By employing a unique box-vector distance metric, Resource2Box learns box embeddings for each resource to accurately rank resources based on user queries. This method addresses the limitations of single-vector representations, as demonstrated by its empirical superiority over vector-based methods.

References

[1] F. Hafizoğlu, "Improving the efficiency of distributed information retrieval using hybrid index partitioning," Master's thesis, Middle East Technical University, 2018.

[2] A. N. Pouamoun and İlker Kocabaş, "Multi-agent-based hybrid peer-to-peer system for distributed information retrieval," *Journal of Information Science*, vol. 0, no. 0, p. 01655515211010392, 0.

[3] J. Callan, *Distributed Information Retrieval*. Boston, MA: Springer US, 2000, pp. 127–150.

[4] M. Shokouhi and L. Si, "Federated search," *Foundations and Trends® in Information Retrieval*, vol. 5, no. 1, pp. 1–102, 2011.

[5] K.-L. Liu, W. Meng, J. Qiu, C. Yu, V. Raghavan, Z. Wu, Y. Lu, H. He, and H. Zhao, "Allinonenews: Development and evaluation of a large-scale news metasearch engine," in *Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '07. New York, NY, USA: Association for Computing Machinery, 2007, p. 1017–1028.

[6] Y. Lu, Z. Wu, H. Zhao, W. Meng, K.-L. Liu, V. Raghavan, and C. Yu, "Mysearchview: A customized metasearch engine generator," in *Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '07. New York, NY, USA: Association for Computing Machinery, 2007, p. 1113–1115.

[7] R. Desai, Q. Yang, Z. Wu, W. Meng, and C. Yu, "Identifying redundant search engines in a very large scale metasearch engine context," in *Proceedings of the 8th Annual ACM International Workshop on Web Information and Data Management*, ser. WIDM '06. New York, NY, USA: Association for Computing Machinery, 2006, p. 51–58.

[8] C. Yu, G. Philip, and W. Meng, "Distributed top-n query processing with possibly uncooperative local systems," in *Proceedings of the 29th International Conference on Very Large Data Bases - Volume 29*, ser. VLDB '03. VLDB Endowment, 2003, p. 117–128.

[9] J. Yuan, L. He, E. C. Dragut, W. Meng, and C. Yu, "Result merging for structured queries on the deep web with active relevance weight estimation," *Information Systems*, vol. 64, pp. 93–103, 2017.

[10] E. Dragut, B. Gupta, B. Beirne, A. Neyestani, B. Atassi, C. Yu, and W. Meng, "Merging query results from local search engines for georeferenced objects," *ACM Transactions on the Web*, vol. 8, 10 2014.

[11] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun, "Graph neural networks: A review of methods and applications," *AI Open*, vol. 1, pp. 57–81, 2020.

[12] L. Mei, J. Mao, G. Guo, and J.-R. Wen, "Learning probabilistic box embeddings for effective and efficient ranking," in *Proceedings of the ACM Web Conference 2022*, ser. WWW '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 473–482.

[13] T. Chen, H. Yin, J. Long, Q. V. H. Nguyen, Y. Wang, and M. Wang, "Thinking inside the box: learning hypercube representations for group recommendation," in *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2022, pp. 1664–1673.

[14] K. Deng, J. Huang, and J. Qin, "Box4rec: Box embedding for sequential recommendation," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 2021, pp. 537–548.

[15] S. Zhang, H. Liu, A. Zhang, Y. Hu, C. Zhang, Y. Li, T. Zhu, S. He, and W. Ou, "Learning user representations with hypercuboids for recommender systems," in *Proceedings of the 14th ACM international conference on web search and data mining*, 2021, pp. 716–724.

[16] Y. Kim, "Robust selective search," *SIGIR Forum*, vol. 52, pp. 170–171, 2019.

[17] Q. Ai, J. Mao, Y. Liu, and W. B. Croft, "Unbiased learning to rank: Theory and practice," in *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, ser. CIKM '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 2305–2306.

[18] J. Xu and W. Croft, "Cluster-based language models for distributed retrieval," 02 2000.

[19] R. Aly, D. Hiemstra, and T. Demeester, "Taily: shard selection using the tail of score distributions," in *Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2013*. United States: Association for Computing Machinery (ACM), Jul. 2013, pp. 673–682, 10.1145/2484028.2484033 ; null ; Conference date: 29-07-2013 Through 01-08-2013.

[20] B. Han, L. Chen, and X. Tian, "Knowledge based collection selection for distributed information retrieval," *Inf. Process. Manage.*, vol. 54, no. 1, p. 116–128, jan 2018.

[21] L. Si and J. Callan, "Relevant document distribution estimation method for resource selection," in *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval*, ser. SIGIR '03. New York, NY, USA: Association for Computing Machinery, 2003, p. 298–305.

[22] M. Shokouhi, "Central-rank-based collection selection in uncooperative distributed information retrieval," vol. 4425, 04 2007, pp. 160–172.

[23] P. Thomas and M. Shokouhi, "Sushi: Scoring scaled samples for server selection," 01 2009, pp. 419–426.

[24] S. Jin and M. Lan, "Simple may be best - A simple and effective method for federated web search via search engine impact factor estimation," in *Proceedings of The Twenty-Third Text REtrieval Conference, TREC 2014, Gaithersburg, Maryland, USA, November 19-21, 2014*, ser. NIST Special Publication, E. M. Voorhees and A. Ellis, Eds., vol. 500-308. National Institute of Standards and Technology (NIST), 2014. [Online]. Available: http://trec.nist.gov/pubs/trec23/papers/pro-ECNU_federated.pdf

[25] G. Urak, H. Ziak, and R. Kern, "Source selection of long tail sources for federated search in an uncooperative setting," in *Proceedings of the 33rd Annual ACM Symposium on Applied Computing*, ser. SAC '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 720–727.

[26] Z. Dai, Y. Kim, and J. Callan, "Learning to rank resources," in *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, ser. SIGIR '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 837–840.

[27] T. Wu, X. Liu, and S. Dong, *LTRRS: A Learning to Rank Based Algorithm for Resource Selection in Distributed Information Retrieval*, 09 2019, pp. 52–63.

[28] U. Ergashev, E. Dragut, and W. Meng, "Learning to rank resources with gnn," in *Proceedings of the ACM Web Conference 2023*, ser. WWW '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 3247–3256.

[29] S. Wang and S. Zhuang, "Resllm: Large language models are strong resource selectors for federated search," https://synthical.com/article/134ba7f9-cceb-4234-9e5f-3193329e812b, 0 2024.

[30] A. Kshemkalyani and M. Singhal, *Distributed Computing: Principles, Algorithms, and Systems*. Cambridge University Press, 2011.

[31] V. Garg, *Principles of Distributed Systems*, ser. Lecture notes in computer science. Springer US, 2012.

[32] H. Ren, W. Hu, and J. Leskovec, "Query2box: Reasoning over knowledge graphs in vector space using box embeddings," in *International Conference on Learning Representations*, 2019.

[33] L. Vilnis, X. Li, S. Murty, and A. McCallum, "Probabilistic embedding of knowledge graphs with box lattice measures," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2018, pp. 263–272.

[34] R. Abboud, I. Ceylan, T. Lukasiewicz, and T. Salvatori, "Boxe: A box embedding model for knowledge base completion," *Advances in Neural Information Processing Systems*, vol. 33, pp. 9649–9661, 2020.

[35] L. Liu, B. Du, H. Ji, C. Zhai, and H. Tong, "Neural-answering logical queries on knowledge graphs," in *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*, 2021, pp. 1087–1097.

[36] Y. Onoe, M. Boratko, A. McCallum, and G. Durrett, "Modeling fine-grained entity types with box embeddings," in *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, 2021, pp. 2051–2064.

[37] X. Chen, M. Boratko, M. Chen, S. S. Dasgupta, X. L. Li, and A. McCallum, "Probabilistic box embeddings for uncertain knowledge graph reasoning," in *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2021, pp. 882–893.

[38] D. Patel and S. Sankar, "Representing joint hierarchies with box embeddings," *Automated Knowledge Base Construction*, 2020.

[39] S. S. Dasgupta, M. Boratko, S. Atmakuri, X. L. Li, D. Patel, and A. McCallum, "Word2box: Learning word representation using box embeddings," *arXiv preprint arXiv:2106.14361*, 2021.

[40] A. Rau, G. Garcia-Hernando, D. Stoyanov, G. J. Brostow, and D. Turmukhambetov, "Predicting visual overlap of images through interpretable non-metric box embeddings," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part V 16*. Springer, 2020, pp. 629–646.

[41] G. Lee, C. Park, and K. Shin, "Set2box: Similarity preserving representation learning for sets," in *2022 IEEE International Conference on Data Mining (ICDM)*. IEEE, 2022, pp. 1023–1028.

[42] B. Mitra and N. Craswell, 2018.

[43] N. Reimers and I. Gurevych, "Sentence-bert: Sentence embeddings using siamese bert-networks," *arXiv preprint arXiv:1908.10084*, 2019.

[44] X. He, K. Deng, X. Wang, Y. Li, Y. Zhang, and M. Wang, "Lightgcn: Simplifying and powering graph convolution network for recommendation," in *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*, 2020, pp. 639–648.

[45] A. Kulkarni, A. S. Tigelaar, D. Hiemstra, and J. Callan, "Shard ranking and cutoff estimation for topically partitioned collections," in *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, ser. CIKM '12. New York, NY, USA: Association for Computing Machinery, 2012, p. 555–564.

[46] D. Hong, L. Si, P. Bracke, M. Witt, and T. Juchcinski, "A joint probabilistic classification model for resource selection," in *Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, ser. SIGIR '10. New York, NY, USA: Association for Computing Machinery, 2010, p. 98–105.

[47] T. Demeester, D. Trieschnigg, D. Nguyen, D. Hiemstra, and K. Zhou, "Fedweb greatest hits: Presenting the new test collection for federated web search," in *Proceedings of the 24th International Conference on World Wide Web*, ser. WWW '15 Companion. New York, NY, USA: Association for Computing Machinery, 2015, p. 27–28.

[48] A. Garba, S. Wu, and S. Khalid, "Federated search techniques: an overview of the trends and state of the art," *Knowledge and Information Systems*, vol. 65, pp. 1–31, 07 2023.

[49] S. Jiang, Q. Yao, Q. Wang, and Y. Sun, "A single vector is not enough: Taxonomy expansion via box embeddings," in *Proceedings of the ACM Web Conference 2023*, ser. WWW '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 2467–2476.

[50] L. Van der Maaten and G. Hinton, "Visualizing data using t-sne." *Journal of machine learning research*, vol. 9, no. 11, 2008.