Efficient Session Retrieval Using Topical Index Shards

Gijs Hendriksen, Djoerd Hiemstra, and Arjen P. de Vries

Radboud University, Nijmegen, The Netherlands
{gijs.hendriksen,djoerd.hiemstra,arjen.devries}@ru.nl

Abstract. Retrieval is often considered one query at a time. However, in practice, queries regularly come in the context of sessions with coherent topics. By dividing a collection into topical index shards and matching the topical context of a session with the right shards, we may reduce the amount of resources required for answering each query. We consider two alternatives: (1) starting with exhaustive search and pruning unnecessary shards after each session turn, and (2) applying a resource selection algorithm to pre-select shards at the start of the session.
We empirically evaluate our approaches on a conversational search dataset (CAsT), and compare effectiveness and resource usage against exhaustive retrieval. Our experiments show that both approaches reduce the number of postings necessary to fulfill a search request (by 50-80%), and in terms of effectiveness our systems are statistically indistinguishable from a system performing exhaustive retrieval.

Keywords: Session search · Efficiency · Resource selection

1 Introduction

Retrieval is often framed as an ad hoc task: given a query, formulated in keywords or natural language by a user, return the documents that are most likely to answer this query (i.e. are the most *relevant*). Often, though, user intent is unclear from a query (e.g. the query may be ambiguous) or the user's information need is multi-faceted. In such cases, a system may need multiple interactive turns to fully satisfy the user's information need. With the increasing capabilities of Large Language Models and the growing research interest in Retrieval Augmented Generation [27], such (conversational) systems have become more feasible to implement. However, to deploy them on large (Web-scale) collections, it is still critical to have a fast retrieval engine that can find and rank relevant documents given such session turns.

Despite all advancements in information retrieval technology in the last decades, traditional inverted file indexes are hard to beat in terms of efficiency. Recently, efficient dense retrieval has been made possible by the development of approximate nearest neighbors search [32]. However, both setups still require a lot of resources when deployed at massive scale with only a single index. A technique called *selective search* [25] aims to speed up large-scale retrieval by

dividing document collections into topical index shards and querying only the handful of shards that are most related to the query.

In this paper, we show that such topical index shards can also be used to speed up multi-turn search sessions. We show that we can start a session with exhaustive search and prune irrelevant shards after each query. We also apply standard resource selection algorithms (commonly used in selective search) to determine the most relevant shards at the beginning of the session (using the first query in the session), and use those shards for the entirety of the session. Both approaches can be used to reduce the required resources in later turns, or may even help in keeping the session grounded on the initial topic if later session turns do not contain the topical context. The selected shards may also be prefetched into memory (or cached [9]) to speed up query processing for later turns, though we do not investigate this use case in detail in this paper. Summarized, our contributions are as follows:

- We introduce a novel approach, which we call *shard pruning*, that uses the topical coherence of queries in a session, combined with the pseudo-relevance feedback of documents retrieved for a given set of queries, to select useful shards for subsequent queries in the same session.
- We show that, while it is challenging for existing resource selection algorithms to select the right shards using information available at the start of the session, it is possible to use only a small set of topical index shards to effectively answer all queries in a session.
- We empirically show that both techniques can greatly reduce the resources required for completing a full search session with only a minimal decrease in effectiveness.

The remainder of the paper is structured as follows. In Section 2, we discuss related work in session search and selective search. In Section 3, we explain how we implement our systems and evaluate our approaches on the TREC CAsT datasets, and Section 4 shows and discusses the corresponding experimental results. Finally, Section 5 discusses our findings and possible directions for future work, and Section 6 concludes the paper. All code used for the experiments in this paper is published to GitLab[1].

2 Related Work

2.1 Session-based retrieval

In ad hoc retrieval, queries are processed and evaluated individually. *Session search*, on the other hand, focuses on retrieval and evaluation of multi-turn search sessions. Most notable work in session search was done through the TREC Session Track [6], in which participants showed that using session information could increase a system's retrieval effectiveness. However, as far as we know, previous

[1] https://gitlab.science.ru.nl/informagus/efficient-session-search/

work on search sessions has mostly focused on increasing the *effectiveness* of retrieval over the later queries in the session. Our work, on the other hand, focuses mostly on the interplay between sessions and the *efficiency* of retrieval systems.

Conversational information seeking (CIS) is a subfield of (session-based) information retrieval in which users interact with the system in the form of natural language *utterances*. There are several challenges specific to conversation-based search, such as extensive natural language understanding, managing context, robust evaluation of highly variable and personal sessions, mixed-initiative, and different interfaces and modalities [45]. We focus on conversational search sessions (i.e. question answering) [3], in which the user's goal is to retrieve information about a specific topic or answer a certain question. The CAsT track at TREC [12,13,14,35] has been set up to evaluate such systems.

2.2 Selective search

Combining ideas from distributed and federated search, *selective search* [25] aims to make large-scale retrieval more efficient by only searching a small fraction of the index for each individual query. Collections are divided into topical index shards, and at query time a *resource selection* algorithm determines which shards are most likely to contain relevant documents for the given query. Retrieval uses the same global statistics and ranking model over all shards, so the scores from different shards are similar and results from different shards can easily be merged.

A large number of resource selection algorithms have been proposed. Some use a central sample index (CSI), which contains a sample (usually less than 5%) of documents from each shard [26,39,40,41]. The CSI is searched using the query, and the documents in the result list are used to rank the shards they originate from. Other approaches create a specific representation for each shard based on a set of summary statistics and use these representations to rank the shards for each query [2,5]. Other resource selection algorithms are based on decision theory [16] or learning to rank [10,15].

Assignment of documents to index shards (called a *shard map*) is usually done by applying unsupervised clustering algorithms (e.g. K-means) on the unigram language models of each document [11,24,43]. To evaluate the quality of a shard map (without setting up a full selective search system and collecting relevance assessments), the AUReC measure [20] uses documents retrieved by exhaustive search to create a recall curve per query and measures the area underneath it. The wAUReC [18] is a variant which also accounts for skew in shard sizes.

In selective search, queries are usually handled one at a time. We are not aware of any work that applies selective search to multiple queries at a time.

3 Experimental Setup

3.1 Dataset

For the evaluation of our approaches on realistic search sessions, we use the TREC Conversational Assistance Track (CAsT) datasets, which contain conversational search sessions with coherent topics. For CAsT 2019 and 2020 [12,13],

the goal was to respond to every utterance with a set of relevant documents. In later years, the task complexity was increased, e.g. by allowing mixed initiative from the conversational agent [14,35]. Since we are mostly interested in coherent retrieval throughout sessions, we only use the simpler CAsT 2019 and 2020 collections to evaluate our approach.

Conversational utterances may depend on context provided in earlier conversation turns. An important aspect of systems that participate in the CAsT tracks is thus query rewriting, e.g. through coreference resolution. However, the track organizers also released manually rewritten queries in which all references to previous turns were resolved. As our focus lies on retrieval effectiveness and the topical coherence of the session (and not on query rewriting), we use these manually resolved versions of the queries.

3.2 Shard Map Creation

To enable efficient, topic-based retrieval, we first need to partition our dataset into a set of shards. To do so, we apply the clustering algorithm `QKLD-QInit` [11] to the CAsT collection. We use the ORCAS query set [8] for initializing the query-biased weights, and GloVe word embeddings [36] to create the initial centroids with `QInit`. Similar to Kulkarni and Callan [24], we sample a subset of the passages to generate the clusters, after which we map the remaining passages to the resulting centroids. Since the passages are much shorter than the average web document, we cluster a 20% sample of the corpus instead of the 1% sample used in previous work. Following Dai et al. [11], we used the hyperparameters $\lambda = 0.1$, $\mu = 0.1$, and $b = 1/16$. Like Kulkarni [24], we choose the number of clusters K such that each shard contains 500k documents on average, resulting in ~77 shards for the CAsT collection. The actual number of shards might differ, as the algorithm splits large shards and merges small shards after clustering.

Because the selection of documents to be clustered (in our 20% sample) may influence the performance of the clustering algorithm, we run the `QKLD-QInit` algorithm three times in total. Then, we compute a session-based wAUReC [18,20] for each shard map as follows:

- We use BM25 to retrieve the top 1000 results for all queries in the first 1000 sessions of the MS MARCO Conversational Search Session dataset [12,34].
- For each session $S = \{q_1, \ldots, q_n\}$, we determine the set of pseudo-relevance documents for S to be the complete set of documents retrieved for all queries in the session:
$$D_S = \bigcup_{q \in S} D_q$$
- Finally, we compute the wAUReC per session using D_S (instead of computing it for each individual query using D_q).

This adaptation to use D_S instead of D_q measures how well (pseudo-)relevant documents for the full session are clustered in each shard. This more closely matches our setup – in which we want to select only a small set of shards for the

duration of the entire session. The three shard maps had session-based wAUReC scores between 0.926 and 0.927, indicating that `QKLD-QInit` produces shard maps of consistent quality. All experiments in this paper were performed with the shard map with the highest session-based wAUReC (0.927).

3.3 Resource Selection

A large variety of resource selection algorithms has been published previously, each with its own (dis)advantages. Most of these have been evaluated on Web document retrieval tasks, though, so it is unclear which algorithm is most suitable for the conversational question answering task (i.e. passage retrieval). We consider several popular algorithms that have proven to be effective in the past.

RBR (relevance-based ranking) [37] is an oracle method that ranks shards by the number of relevant documents they contain and selects the top k shards with the highest number of relevant documents.

SRBR (session-based relevance-based ranking) is our session-based variation on the RBR. Instead of counting the number of relevant documents per query, we group the relevant documents for all queries in a session together. SRBR ranks the shards per session based on these aggregate counts, and uses this rankingfor all queries in the session.

CORI [5] represents (the dictionary of) each shard as a single large document, and then performs retrieval (using INQUERY) over the resulting collection of large document representations.

ReDDE [40] uses a central sample index (CSI) to store a small subset of the documents from each shard. For a query, they rank the documents in the CSI and count the number of times each shard appears in the top $k(=1000)$ results, after which they scale the shard scores based on their size. Following Kulkarni [23], we sum the relevance scores of the returned documents (instead of just counting them) and do not include the size-based shard weights.

Taily [2] estimates the distribution of unigram language model scores per shard (with Dirichlet smoothing) by fitting gamma distributions on term-based score statistics, for each of the query terms. Taily then selects all shards that, according to this probability distribution, are likely to retrieve at least v documents out of the collection-wide top n_c documents for this query. We use the default parameters: $n_c = 400, v = 50$.

Rank-S [26] retrieves documents from a CSI (like ReDDE) and lets each document vote for the shard they originate from. The vote strength is based on the relevance score and decays exponentially with the document's rank. The parameter B controls the decay rate; we use the default $B = 50$.

L2R [10] is a resource selection algorithm based on learning-to-rank. We use the following features, also presented by Dai et al. [10]: shard popularity, champion list scores, and the (binned) scores returned by CORI, ReDDE, Rank-S and Taily. Following Arguello et al. [4] and Dai et al. [10], we use exhaustive retrieval results to generate the training data for the learning-to-rank model. Specifically, we use BM25 to retrieve the top 1000 results

for the first query of each of the first 1000 sessions of the MS MARCO Conversational Search Session dataset [12,34], and rank the shards based on the number of top 1000 results they contain. We use 750 queries for training and 250 queries for validation.

We use the same central sample index for both ReDDE and Rank-S, which is constructed by uniformly sampling 4% of the documents in the collection.

Note that most previous work on selective search has focused on achieving high precision in the top ranks. In our work, we also consider selective search as a first-stage retrieval system, for which we might need to select more shards on average. Instead of using default shard cutoffs for each of the resource selection algorithms, we empirically evaluate which shard cutoff is most suitable per resource selection algorithm and retrieval setting. We also do this for Taily and Rank-S, which normally apply score thresholds instead of static shard cutoffs.

3.4 Implementation

In order to easily experiment with different resource selection algorithms, shard representations, and retrieval models, we implemented each system in the analytical database engine DuckDB [38]. We followed the approach by Mühleisen et al. [33] to represent the inverted file and implement BM25 as main retrieval model. Most of the representations necessary for resource selection could then be easily expressed in the form of SQL queries: statistics for CORI and Taily are implemented through simple aggregate queries; creation of a CSI was done by sampling from the documents table; filtering on query terms and selected shards could be done through JOIN operators; and so forth.

For the selective search systems, we search all selected shards with BM25 using global term/document statistics (i.e. the same statistics as exhaustive search). This results in the exact same scores and rankings as on exhaustive search, but filtered to only contain documents from the selected shards. We remove stopwords according to the stopword list in DuckDB's full-text search extension. For all experiments, we use the BM25 parameters $k_1 = 0.9$ and $b = 0.4$ (the defaults of Anserini [44], which have been shown to work well on the MS Marco passage ranking task [28]).

We implemented PyTerrier [31] transformer interfaces for our systems and used PyTerrier and its `ir_datasets` [30] and `ir_measures` [29] bindings for running all our experiments. The learning-to-rank resource selection model was implemented using the XGBoost [7] implementation of LambdaMART [42].

3.5 Evaluation

To evaluate the effectiveness of different systems, we use two of the official CAsT 2019 and 2020 metrics – MAP@1000 and nDCG@3 – as well as R@1000. For MAP and R, which need binary relevance, we use a relevance cutoff of 2. Since conversational systems are often limited in the number of results they can present to the user, the CAsT organizers decided to focus on the top of the ranked list

and use nDCG@3 as their main metric. In our work, we consider R@1000 as well, since we also want to consider selective search as the first stage of a multi-stage pipeline; in such cases, the shallow metrics like nDCG@3 metrics would likely be improved by applying a strong re-ranker on the initial result set returned by our systems.

To measure the efficiency of a system, we use the Cost-in-Shards (CiS) and Cost-in-Postings (CiP) measures introduced by Kulkarni [23]. The CiS simply measures the number of shards selected by the resource selection algorithm. The CiP measures the number of postings accessed in the selected shards during *retrieval* (CiP_R), plus the number of postings used for *shard selection* (CiP_{SS}). For sample-based resource selection algorithms (ReDDE and Rank-S), CiP_{SS} is equal to the number of postings used from the CSI. For CORI, it is equal to the number of postings used from its index of large document representations. For Taily, it equals the number of term statistics accessed for all shard representations. For L2R, we sum the $\text{CiP}_S S$ of all input features. Finally, for RBR and exhaustive search, we assume no costs for resource selection (i.e. $\text{CiP}_{SS} = 0$).

While query latency might depend on a large number of other factors besides the total number of postings in the selected shards (e.g. load (im)balance [17,22], caching [9], and dynamic pruning [21]), previous work has shown the CiP is highly correlated with query latency for single-node systems [23, Chapter 3]. We use the CiP as a hardware-agnostic abstraction of the total load on the distributed system's resources. This allows us to compare the total system load of different search configurations in a simulated search environment, without having to worry about implementation details. In practice, a distributed selective search system would be set up with load balancing, caching and other runtime optimizations enabled.

To determine the statistical significance of our results, we use two approaches. First, Kim [19, Chapter 2] noticed that regular significance tests are not suitable for testing whether the effectiveness of a selective search system is equivalent to that of exhaustive search. If we fail to reject the null hypothesis $\mathcal{H}_0 : \mu_A = \mu_B$, it does not necessarily mean that the null hypothesis is true (and thus, the two systems perform equivalently). Instead, Kim [19, Chapter 5] uses a non-inferiority test (commonly used in the medical field) to determine whether selective search performs at most a small margin δ worse than exhaustive search. Following Kim, we set δ to 5% and 10% of the mean of the exhaustive search results. In some cases, selective search may outperform exhaustive search (e.g. if high-scoring irrelevant documents are part of shards not selected by the resource selection algorithm). We use a paired t-test to determine such cases. In all significance tests, we use a significance threshold of $\alpha = 0.01$ (i.e. a confidence level of 99%).

4 Results

In this section, we discuss our experimental results on the CAsT 2019 and 2020 datasets. First, we evaluate a setting in which we start with exhaustive search,

Table 1: Performance of systems that iteratively prune shards after each conversation turn, for CAsT 2019 and 2020. Best results per metric are marked in bold. Results that are significantly non-inferior to exhaustive retrieval (with $p < 0.01$) are marked with † for $\delta < 5\%$ and ‡ for $\delta < 10\%$.

(a) CAsT 2019

	MAP@1000	R@1000	nDCG@3	CiS	CiP ($\times 10^3$)	
Exhaustive	0.25	**0.84**	**0.35**	94.0	1197.5	
Shard pruning	**0.25**†‡	0.83†‡	0.35†‡	35.5	614.3	(–49%)

(b) CAsT 2020

	MAP@1000	R@1000	nDCG@3	CiS	CiP ($\times 10^3$)	
Exhaustive	**0.17**	**0.73**	**0.29**	94.0	1486.0	
Shard pruning	0.17‡	0.72†‡	0.28†‡	37.7	738.3	(–50%)

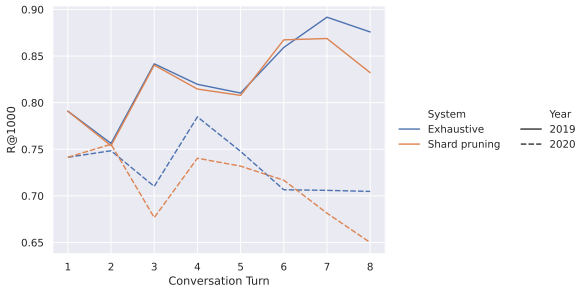
and iteratively prune shards with no retrieved documents to speed up query processing for later session turns. Then, we consider a setup in which we apply a resource selection algorithm to select the right shards at the start of the session.

4.1 Shard pruning

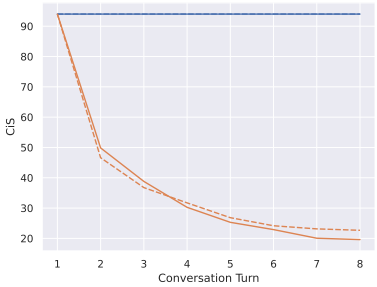
We start with a system that uses pseudo-relevance feedback of retrieved documents to select shards for later turns. This novel approach, which we call *shard pruning*, starts with exhaustive search for the first query in the session. We register which shards did not contribute any documents to the top 1500 retrieved documents, and remove them from consideration for subsequent session turns. In other words, each time we process a query, we prune the set of shards from which documents are retrieved. The rationale behind this approach is that the session topic should become more pronounced as the session proceeds, and thus the number of shards under consideration can be reduced as we go. The possible downside, of course, is that we prune shards that are not relevant to a specific query in the session but would be useful for a later one.

Table 1 shows the performance of a system incorporating this shard pruning approach. For both datasets, we clearly see that shard pruning can skip nearly half of the postings without a drop in quality when compared to exhaustive search. In some cases, the system even improves after pruning certain shards, because the pruned shards might contain irrelevant documents that are mistakenly ranked highly by exhaustive search.

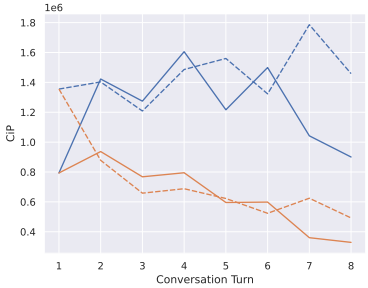
Figure 1 shows the R@1000, CiS, and CiP as functions of the conversation turn (up until turn eight). Figure 1b shows how quickly shards are eliminated after each successive query. In the later queries, we search (on average) less than a third of all shards. Figure 1c similarly shows how the difference in CiP between



(a) Performance (R@1000)



(b) Cost in Shards



(c) Cost in Postings

Fig. 1: Performance of shard pruning systems for different conversation depths, compared to exhaustive search.

exhaustive search and our shard pruning systems grows with the session depth. While we would expect the CiP to show a monotonic decrease (like the CiS), Figure 1c shows it sometimes fluctuates upwards as well. This is likely caused by the query term distribution: some queries are longer or contain terms with longer posting lists. As a result, the CiP might be higher for certain queries, even if we use fewer shards.

Finally, Figure 1a shows that the R@1000 of our system is almost identical to that of exhaustive search for CAsT 2019, until the end of the conversation. For CAsT 2020, there is a decrease in performance starting from turn three, but the difference is minimal.

Note that our shard pruning approach considers the top 1500 documents retrieved from the selected shards, while only the top 1000 documents are returned by the system (and evaluated in our metrics). This allows the shard pruning to be slightly more conservative in which shards it decides to prune. Early experiments showed that using only the top 1000 documents might make the system prune too aggressively and drop in performance for later queries in each session.

Since most shards are pruned near the beginning of the session, a system may also choose to stop pruning at a certain session length or when a certain number

of shards has been reached. This can prevent it from decreasing effectiveness too much in the later turns.

4.2 Shard pre-selection

We have also experimented with a variation on the classic selective search setting, where we select the shards to use throughout the session after receiving user input at the start of the session. Note the difference with selective search: instead of performing resource selection for every query individually, we perform resource selection only once, using the first query in the session. The resulting list of shards is used for all queries in the session.

Table 2: Performance of systems that select shards for the whole session using only the first query. For each system, we show the cheapest configuration such that its nDCG@3 matches that of exhaustive retrieval. Best results per metric are marked in bold. Results that are significantly non-inferior to exhaustive retrieval (with $p < 0.01$) are marked with † for $\delta < 5\%$ and ‡ for $\delta < 10\%$.

(a) CAsT 2019

	MAP@1000	R@1000	nDCG@3	CiS	CiP ($\times 10^3$)	
Exhaustive	**0.25**	**0.84**	0.35	94.0	1197.5	
SRBR	0.24‡	0.69	**0.36**†‡	2.0	102.5	(–91%)
CORI	0.25†‡	0.83†‡	0.34†‡	58.0	877.6	(–27%)
ReDDE	0.22	0.69	0.36†‡	7.0	228.4	(–81%)
Rank-S	0.23	0.75	0.35†‡	19.0	410.0	(–66%)
Taily	0.23	0.72	0.35†‡	15.0	324.3	(–73%)
L2R	0.22	0.69	0.36†‡	5.0	186.4	(–84%)

(b) CAsT 2020

	MAP@1000	R@1000	nDCG@3	CiS	CiP ($\times 10^3$)	
Exhaustive	0.17	**0.73**	0.29	94.0	1486.0	
SRBR	0.17‡	0.70‡	0.29†‡	3.9	161.5	(–89%)
CORI	0.17‡	0.70‡	0.29†‡	24.0	564.5	(–62%)
ReDDE	**0.18**†‡	0.72†‡	**0.29**†‡	27.9	672.9	(–55%)
Rank-S	0.18†‡	0.70‡	0.29†‡	25.0	590.0	(–60%)
Taily	0.17‡	0.72†‡	0.29†‡	57.0	1032.4	(–31%)
L2R	0.17†‡	0.69‡	0.29†‡	14.0	409.9	(–72%)

To ensure our system is comparable to exhaustive search in terms of effectiveness (but more efficient), we experiment with different shard cutoffs k for each resource selection algorithm, and show the results for the smallest k with

which the system is statistically significantly non-inferior (within a 5% margin) to exhaustive search. We do so in two settings: one with a focus on early precision and one that is more recall-oriented.

Early precision When focusing on early precision, we want the shard preselection system to be non-inferior to exhaustive search in terms of nDCG@3, following the official CAsT metrics. Table 2 shows the performance of each resource selection algorithm, in the setting where they reach similar nDCG as the exhaustive search baseline. Our session-based oracle method SRBR can reach adequate performance with only a very small number of shards: 2 for CAsT 2019 and 4 for CAsT 2020. This indicates that the relevant documents per search session are clustered together correctly in our shard map.

If we compare the different resource selection algorithms, L2R selects the fewest shards out of all of them. This is in line with previous work, which showed the learning-to-rank approach to outperform the other individual models. Even with the extra cost associated with feature extraction, the L2R model can drastically reduce the number of postings required for answering each query (by 84% and 72% for CAsT 2019 and 2020, respectively).

From the other resource selection algorithms, ReDDE performs best for CAsT 2019 and CORI performs best for CAsT 2020. Interestingly, CORI performs quite poorly on CAsT 2019, and similarly, Taily is underperforming for CAsT 2020. It is unclear why these systems perform well for one dataset and poorly for the other. On average, the required number of shards for CAsT 2020 is higher than for CAsT 2019. This seems to imply the topics for CAsT 2020 are broader (so they span more shards) and more complicated. The fact that all metrics are lower for CAsT 2020 than for CAsT 2019 confirms this.

Recall-oriented Our recall-oriented setting models selective search as a first-stage retrieval system, and assumes the performance in top ranks will be further improved in a later stage by applying a stronger reranker on the initial result set. As such, we optimize the R@1000 instead of the nDCG@3.

Table 3 shows the performance of each resource selection algorithm such that the R@1000 is non-inferior to that of exhaustive search. Again, our oracle method SRBR only needs to select a small number of shards in order to retrieve as many relevant documents as the exhaustive baseline. In fact, by selecting only 5.7 shards (on average) per session, it can outperform the exhaustive system on precision-oriented metrics MAP@1000 and nDCG@3 for CAsT 2019. This again highlights that topical clustering, when aligned with the session topic, can be used to remove highly ranked but irrelevant documents from the results.

The other resource selection algorithms behave similarly to the early precision setting. L2R performs best, followed by ReDDE and then the other systems. Interestingly, most systems are now able to select fewer shards for CAsT 2020 than for CAsT 2019 – contrary to what we saw in the early precision setting. Again, CORI and Taily seem to struggle to select the right shards per session in some cases. One possible explanation for this observation is the fact that

Table 3: Performance of systems that select shards for the whole session using only the first query. For each system, we show the cheapest configuration such that its R@1000 matches that of exhaustive retrieval. Best results per metric are marked in bold. Results that are significantly non-inferior to exhaustive retrieval (with $p < 0.01$) are marked with † for $\delta < 5\%$ and ‡ for $\delta < 10\%$. Results that significantly outperform exhaustive retrieval (with $p < 0.01$) are marked with +.

(a) CAsT 2019

	MAP@1000	R@1000	nDCG@3	CiS	CiP ($\times 10^3$)	
Exhaustive	0.25	**0.84**	0.35	94.0	1197.5	
SRBR	**0.26**$^{\dagger\ddagger+}$	0.81†‡	**0.37**$^{\dagger\ddagger+}$	5.7	206.9	(–83%)
CORI	0.25†‡	0.83†‡	0.34†‡	58.0	877.6	(–27%)
ReDDE	0.25†‡	0.82†‡	0.35†‡	32.0	596.4	(–50%)
Rank-S	0.24‡	0.82†‡	0.35†‡	36.8	642.1	(–46%)
Taily	0.24†‡	0.82†‡	0.35†‡	51.0	778.6	(–35%)
L2R	0.25†‡	0.82†‡	0.35†‡	25.0	516.7	(–57%)

(b) CAsT 2020

	MAP@1000	R@1000	nDCG@3	CiS	CiP ($\times 10^3$)	
Exhaustive	0.17	**0.73**	0.29	94.0	1486.0	
SRBR	**0.18**†‡	0.72†‡	**0.29**†‡	4.9	190.4	(–87%)
CORI	0.18†‡	0.72†‡	0.29†‡	32.0	700.8	(–53%)
ReDDE	0.18†‡	0.72†‡	0.29†‡	27.9	672.9	(–55%)
Rank-S	0.17†‡	0.72†‡	0.29†‡	33.7	749.8	(–50%)
Taily	0.17‡	0.72†‡	0.28‡	56.0	1018.4	(–31%)
L2R	0.18†‡	0.72†‡	0.29†‡	19.0	504.1	(–66%)

the CAsT dataset contains longer natural language queries, as opposed to the shorter keyword-style queries they have previously been evaluated on.

5 Discussion

From our experiments, we clearly see the viability of using topically partitioned document collections to make conversational question answering more efficient: high recall can still be achieved with a 50% reduction in costs. A system tuned for early precision requires even less resources. Starting with exhaustive search and pruning unused shards after each turn is apparently a viable strategy, and a combination of selective search and shard pruning seems a promising direction for further exploration.

Our setup in which we pre-select all shards for the full session is extremely effective when we use oracle resource selection. However, in practice, it turns

out to be difficult for existing resource selection algorithms to use information available in the beginning of the session to select the right shards straight away. Future work could explore new resource selection strategies to determine a session's topical context from the first query and select the right shards accordingly.

We also evaluated the standard selective search approach. Selecting shards for every individual query in the session leads to more fine-grained shard selection than taking this decision once at the start of the session. For instance, on the recall-oriented scenario, L2R needs to select only 13-14 shards per query to match the exhaustive system (compared to the 19-25 shown in Table 3; the RBR oracle method only needs 2-3). However, we envision a scenario in which accessing a shard for the first time is costly – too costly to perform at query time. For example, one could set up a local, personal search engine that downloads topical index shards based on user preferences, and uses those to search locally. In that context, it makes sense to pre-download the right shards before issuing a query, as downloading the shards at query time would result in a high query latency. This work, which shows the viability of selecting shards for sessions, serves as a stepping stone to explore this setup in future work; e.g. by selecting shards for full personal profiles instead of single search sessions.

6 Conclusion

We have argued that the topical coherence of queries in a search session should provide new opportunities to apply selective search and reduce query processing costs. We considered two approaches: one in which we start with exhaustive search and prune unnecessary shards after each query, and one in which we apply a resource selection algorithm at the start of the session to select shards for the full session duration. While it appears challenging to select the right set of shards at the very beginning of a session, we have shown empirically on the CAsT 2019 and 2020 datasets that it is possible to greatly improve the efficiency of (conversational) search systems, while reaching the same effectiveness as exhaustive search – both in early precision and in recall-oriented scenarios.

Our next step in this line of research is to further explore and improve resource selection for search sessions in more detail. Aside from improving session-based shard selection, we wish to extend our approach to longer sessions and user profiles. For instance, if one can determine a user's preferences ahead of time and download the right shards for their profile, they can create a personal, offline search engine that fits their interests. We wish to research this scenario in depth, starting with the personalized conversation-based iKAT collection [1].

Acknowledgments. This work has received funding from the European Union's Horizon Europe research and innovation programme under grant agreement No 101070014 (OpenWebSearch.EU, https://doi.org/10.3030/101070014).

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Aliannejadi, M., Abbasiantaeb, Z., Chatterjee, S., Dalton, J., Azzopardi, L.: TREC iKAT 2023: A Test Collection for Evaluating Conversational and Interactive Knowledge Assistants. In: Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval. pp. 819–829. SIGIR '24, Association for Computing Machinery, New York, NY, USA (Jul 2024). https://doi.org/10.1145/3626772.3657860
2. Aly, R., Hiemstra, D., Demeester, T.: 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. pp. 673–682. SIGIR '13, Association for Computing Machinery, New York, NY, USA (Jul 2013). https://doi.org/10.1145/2484028.2484033
3. Anand, A., Cavedon, L., Joho, H., Sanderson, M., Stein, B.: Conversational Search (Dagstuhl Seminar 19461). DROPS-IDN/v2/document/10.4230/DagRep.9.11.34 (2020). https://doi.org/10.4230/DagRep.9.11.34
4. Arguello, J., Callan, J., Diaz, F.: Classification-based resource selection. In: Proceedings of the 18th ACM Conference on Information and Knowledge Management. pp. 1277–1286. CIKM '09, Association for Computing Machinery, New York, NY, USA (Nov 2009). https://doi.org/10.1145/1645953.1646115
5. Callan, J.P., Lu, Z., Croft, W.B.: Searching distributed collections with inference networks. In: Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. pp. 21–28. SIGIR '95, Association for Computing Machinery, New York, NY, USA (Jul 1995). https://doi.org/10.1145/215206.215328
6. Carterette, B., Clough, P., Hall, M., Kanoulas, E., Sanderson, M.: Evaluating Retrieval over Sessions: The TREC Session Track 2011-2014. In: Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval. pp. 685–688. SIGIR '16, Association for Computing Machinery, New York, NY, USA (Jul 2016). https://doi.org/10.1145/2911451.2914675
7. Chen, T., Guestrin, C.: XGBoost: A Scalable Tree Boosting System. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 785–794. KDD '16, Association for Computing Machinery, New York, NY, USA (Aug 2016). https://doi.org/10.1145/2939672.2939785
8. Craswell, N., Campos, D., Mitra, B., Yilmaz, E., Billerbeck, B.: ORCAS: 18 Million Clicked Query-Document Pairs for Analyzing Search. In: Proceedings of the 29th ACM International Conference on Information & Knowledge Management. pp. 2983–2989. CIKM '20, Association for Computing Machinery, New York, NY, USA (Oct 2020). https://doi.org/10.1145/3340531.3412779
9. Dai, Z., Callan, J.: Inverted List Caching for Topical Index Shards. In: Pasi, G., Piwowarski, B., Azzopardi, L., Hanbury, A. (eds.) Advances in Information Retrieval, vol. 10772, pp. 577–583. Springer International Publishing, Cham (2018). https://doi.org/10.1007/978-3-319-76941-7_47
10. Dai, Z., Kim, Y., Callan, J.: Learning To Rank Resources. In: Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval. pp. 837–840. SIGIR '17, Association for Computing Machinery, New York, NY, USA (Aug 2017). https://doi.org/10.1145/3077136.3080657
11. Dai, Z., Xiong, C., Callan, J.: Query-Biased Partitioning for Selective Search. In: Proceedings of the 25th ACM International on Conference on Information and Knowledge Management. pp. 1119–1128. CIKM '16, Association for Computing

Machinery, New York, NY, USA (Oct 2016). https://doi.org/10.1145/2983323.2983706
12. Dalton, J., Xiong, C., Callan, J.: TREC CAsT 2019: The conversational assistance track overview. In: Voorhees, E.M., Ellis, A. (eds.) Proceedings of the Twenty-Eighth Text Retrieval Conference, TREC 2019, Gaithersburg, Maryland, USA, November 13-15, 2019. NIST Special Publication, vol. 1250. National Institute of Standards and Technology (NIST) (2019)
13. Dalton, J., Xiong, C., Callan, J.: CAsT 2020: The conversational assistance track overview. In: Voorhees, E.M., Ellis, A. (eds.) Proceedings of the Twenty-Ninth Text Retrieval Conference, TREC 2020, Virtual Event [Gaithersburg, Maryland, USA], November 16-20, 2020. NIST Special Publication, vol. 1266. National Institute of Standards and Technology (NIST) (2020)
14. Dalton, J., Xiong, C., Callan, J.: TREC CAsT 2021: The conversational assistance track overview. In: Soboroff, I., Ellis, A. (eds.) Proceedings of the Thirtieth Text Retrieval Conference, TREC 2021, Online, November 15-19, 2021. NIST Special Publication, vol. 500–335. National Institute of Standards and Technology (NIST) (2021)
15. Ergashev, U., Dragut, E., Meng, W.: Learning To Rank Resources with GNN. In: Proceedings of the ACM Web Conference 2023. pp. 3247–3256. WWW '23, Association for Computing Machinery, New York, NY, USA (Apr 2023). https://doi.org/10.1145/3543507.3583360
16. Fuhr, N.: A decision-theoretic approach to database selection in networked IR. ACM Trans. Inf. Syst. **17**(3), 229–249 (Jul 1999). https://doi.org/10.1145/314516.314517
17. Hafizoglu, F., Kucukoglu, E.C., Altingovde, I.S.: On the Efficiency of Selective Search. In: Jose, J.M., Hauff, C., Altıngovde, I.S., Song, D., Albakour, D., Watt, S., Tait, J. (eds.) Advances in Information Retrieval. pp. 705–712. Springer International Publishing, Cham (2017). https://doi.org/10.1007/978-3-319-56608-5_69
18. Hendriksen, G., Hiemstra, D., de Vries, A.P.: Weighted AUReC: Handling Skew in Shard Map Quality Estimation for Selective Search. In: Advances in Information Retrieval: 46th European Conference on Information Retrieval, ECIR 2024, Glasgow, UK, March 24–28, 2024, Proceedings, Part IV. pp. 87–96. Springer-Verlag, Berlin, Heidelberg (Mar 2024). https://doi.org/10.1007/978-3-031-56066-8_10
19. Kim, Y.: Robust Selective Search. Ph.D. thesis, Carnegie Mellon University (2019)
20. Kim, Y., Callan, J.: Measuring the Effectiveness of Selective Search Index Partitions without Supervision. In: Proceedings of the 2018 ACM SIGIR International Conference on Theory of Information Retrieval. pp. 91–98. ICTIR '18, Association for Computing Machinery, New York, NY, USA (Sep 2018). https://doi.org/10.1145/3234944.3234952
21. Kim, Y., Callan, J., Culpepper, J.S., Moffat, A.: Does Selective Search Benefit from WAND Optimization? In: Ferro, N., Crestani, F., Moens, M.F., Mothe, J., Silvestri, F., Di Nunzio, G.M., Hauff, C., Silvello, G. (eds.) Advances in Information Retrieval. pp. 145–158. Lecture Notes in Computer Science, Springer International Publishing, Cham (2016). https://doi.org/10.1007/978-3-319-30671-1_11
22. Kim, Y., Callan, J., Culpepper, J.S., Moffat, A.: Load-Balancing in Distributed Selective Search. In: Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval. pp. 905–908. SIGIR '16, Association for Computing Machinery, New York, NY, USA (Jul 2016). https://doi.org/10.1145/2911451.2914689
23. Kulkarni, A.: Efficient and Effective Large-scale Search. Ph.D. thesis, Carnegie Mellon University (Apr 2013)

24. Kulkarni, A., Callan, J.: Document allocation policies for selective searching of distributed indexes. In: Proceedings of the 19th ACM International Conference on Information and Knowledge Management. pp. 449–458. CIKM '10, Association for Computing Machinery, New York, NY, USA (Oct 2010). https://doi.org/10.1145/1871437.1871497
25. Kulkarni, A., Callan, J.: Selective Search: Efficient and Effective Search of Large Textual Collections. ACM Transactions on Information Systems **33**(4), 17:1–17:33 (Apr 2015). https://doi.org/10.1145/2738035
26. Kulkarni, A., Tigelaar, A.S., Hiemstra, D., Callan, J.: Shard ranking and cutoff estimation for topically partitioned collections. In: Proceedings of the 21st ACM International Conference on Information and Knowledge Management. pp. 555–564. ACM, Maui Hawaii USA (Oct 2012). https://doi.org/10.1145/2396761.2396833
27. Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.t., Rocktäschel, T., Riedel, S., Kiela, D.: Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks (Apr 2021). https://doi.org/10.48550/arXiv.2005.11401
28. Lin, J., Ma, X., Lin, S.C., Yang, J.H., Pradeep, R., Nogueira, R.: Pyserini: A Python Toolkit for Reproducible Information Retrieval Research with Sparse and Dense Representations. In: Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval. pp. 2356–2362. SIGIR '21, Association for Computing Machinery, New York, NY, USA (Jul 2021). https://doi.org/10.1145/3404835.3463238
29. MacAvaney, S., Macdonald, C., Ounis, I.: Streamlining Evaluation with ir-measures. In: Hagen, M., Verberne, S., Macdonald, C., Seifert, C., Balog, K., Nørvåg, K., Setty, V. (eds.) Advances in Information Retrieval. pp. 305–310. Springer International Publishing, Cham (2022). https://doi.org/10.1007/978-3-030-99739-7_38
30. MacAvaney, S., Yates, A., Feldman, S., Downey, D., Cohan, A., Goharian, N.: Simplified Data Wrangling with ir_datasets. In: Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval. pp. 2429–2436. SIGIR '21, Association for Computing Machinery, New York, NY, USA (Jul 2021). https://doi.org/10.1145/3404835.3463254
31. Macdonald, C., Tonellotto, N.: Declarative Experimentation in Information Retrieval using PyTerrier. In: Proceedings of the 2020 ACM SIGIR on International Conference on Theory of Information Retrieval. pp. 161–168 (Sep 2020). https://doi.org/10.1145/3409256.3409829
32. Malkov, Y.A., Yashunin, D.A.: Efficient and Robust Approximate Nearest Neighbor Search Using Hierarchical Navigable Small World Graphs. IEEE Transactions on Pattern Analysis and Machine Intelligence **42**(4), 824–836 (Apr 2020). https://doi.org/10.1109/TPAMI.2018.2889473
33. Mühleisen, H., Samar, T., Lin, J., de Vries, A.: Old Dogs Are Great at New Tricks: Column Stores for IR Prototyping. In: Proceedings of the 37th International ACM SIGIR Conference on Research & Development in Information Retrieval. pp. 863–866. SIGIR '14, Association for Computing Machinery, New York, NY, USA (Jul 2014). https://doi.org/10.1145/2600428.2609460
34. Nguyen, T., Rosenberg, M., Song, X., Gao, J., Tiwary, S., Majumder, R., Deng, L.: MS MARCO: A Human Generated MAchine Reading COmprehension Dataset. In: Besold, T.R., Bordes, A., d'Avila Garcez, A.S., Wayne, G. (eds.) Proceedings of the Workshop on Cognitive Computation: Integrating Neural and Symbolic Approaches 2016 Co-Located with the 30th Annual Conference on Neural Informa-

tion Processing Systems (NIPS 2016), Barcelona, Spain, December 9, 2016. CEUR Workshop Proceedings, vol. 1773. CEUR-WS.org (2016)
35. Owoicho, P., Dalton, J., Aliannejadi, M., Azzopardi, L., Trippas, J.R., Vakulenko, S.: TREC CAsT 2022: Going beyond user ask and system retrieve with initiative and response generation. In: TREC (2022)
36. Pennington, J., Socher, R., Manning, C.: GloVe: Global Vectors for Word Representation. In: Moschitti, A., Pang, B., Daelemans, W. (eds.) Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). pp. 1532–1543. Association for Computational Linguistics, Doha, Qatar (Oct 2014). https://doi.org/10.3115/v1/D14-1162
37. Powell, A.L., French, J.C.: Comparing the performance of collection selection algorithms. ACM Trans. Inf. Syst. **21**(4), 412–456 (Oct 2003). https://doi.org/10.1145/944012.944016
38. Raasveldt, M., Mühleisen, H.: DuckDB: An Embeddable Analytical Database. In: Proceedings of the 2019 International Conference on Management of Data. pp. 1981–1984. SIGMOD '19, Association for Computing Machinery, New York, NY, USA (Jun 2019). https://doi.org/10.1145/3299869.3320212
39. Shokouhi, M.: Central-Rank-Based Collection Selection in Uncooperative Distributed Information Retrieval. In: Amati, G., Carpineto, C., Romano, G. (eds.) Advances in Information Retrieval. pp. 160–172. Springer, Berlin, Heidelberg (2007). https://doi.org/10.1007/978-3-540-71496-5_17
40. Si, L., Callan, J.: 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. pp. 298–305. SIGIR '03, Association for Computing Machinery, New York, NY, USA (Jul 2003). https://doi.org/10.1145/860435.860490
41. Thomas, P., Shokouhi, M.: SUSHI: Scoring scaled samples for server selection. In: Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval. pp. 419–426. SIGIR '09, Association for Computing Machinery, New York, NY, USA (Jul 2009). https://doi.org/10.1145/1571941.1572014
42. Wu, Q., Burges, C.J.C., Svore, K.M., Gao, J.: Adapting boosting for information retrieval measures. Information Retrieval **13**(3), 254–270 (Jun 2010). https://doi.org/10.1007/s10791-009-9112-1
43. Xu, J., Croft, W.B.: Cluster-based language models for distributed retrieval. In: Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. pp. 254–261. SIGIR '99, Association for Computing Machinery, New York, NY, USA (Aug 1999). https://doi.org/10.1145/312624.312687
44. Yang, P., Fang, H., Lin, J.: Anserini: Enabling the Use of Lucene for Information Retrieval Research. In: Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval. pp. 1253–1256. SIGIR '17, Association for Computing Machinery, New York, NY, USA (Aug 2017). https://doi.org/10.1145/3077136.3080721
45. Zamani, H., Trippas, J.R., Dalton, J., Radlinski, F.: Conversational Information Seeking. Foundations and Trends in Information Retrieval **17**(3-4), 244–456 (2023). https://doi.org/10.1561/1500000081