Effective Adhoc Retrieval Through Traversal of a Query-Document Graph

Erlend Frayling(✉), Sean MacAvaney, Craig Macdonald, and Iadh Ounis

University of Glasgow, Glasgow, UK
{erlend.frayling,sean.macavaney,craig.macdonald,iadh.ounis}@glasgow.ac.uk

Abstract. Adhoc retrieval is the task of effectively retrieving information for an end-user's information need, usually expressed as a textual query. One of the most well-established retrieval frameworks is the two-stage retrieval pipeline, whereby an inexpensive retrieval algorithm retrieves a subset of candidate documents from a corpus, and a more sophisticated (but costly) model re-ranks these candidates. A notable limitation of this two-stage framework is that the second stage re-ranking model can only re-order documents, and any relevant documents not retrieved from the corpus in the first stage are entirely lost to the second stage. A recently-proposed Adaptive Re-Ranking technique has shown that extending the candidate pool by traversing a document similarity graph can overcome this recall problem. However, this traversal technique is agnostic of the user's query, which has the potential to waste compute resources by scoring documents that are not related to the query. In this work, we propose an alternative formulation of the document similarity graph. Rather than using document similarities, we propose a weighted bipartite graph that consists of both document nodes and query nodes. This overcomes the limitations of prior Adaptive Re-Ranking approaches because the bipartite graph can be navigated in a manner that explicitly acknowledges the original user query issued to the search pipeline. We evaluate the effectiveness of our proposed framework by experimenting with the TREC Deep Learning track in a standard adhoc retrieval setting. We find that our approach outperforms state-of-the-art two-stage re-ranking pipelines, improving the nDCG@10 metric by 5.8% on the DL19 test collection.

1 Introduction

Adhoc retrieval tasks aim to retrieve and rank information relevant to queries. Queries are typically provided by an end-user and represent some information need the end-user has. The most effective retrieval methods tend to use a multi-stage cascading approach, wherein an inexpensive first-stage retrieval algorithm (e.g., a lexical retriever like BM25 [29] or a dense retriever such as TAS-B [11]) returns a set of candidate documents, which are subsequently re-scored using a more costly and sophisticated model (such as a BERT-based cross-encoder) [18]. Although the second stage "re-ranker" can improve the precision of the top

N. Goharian et al. (Eds.): ECIR 2024, LNCS 14610, pp. 89–104, 2024.
https://doi.org/10.1007/978-3-031-56063-7_6

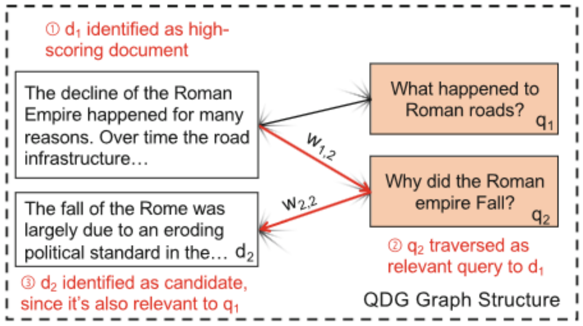

① d₁ identified as high-scoring document
The decline of the Roman Empire happened for many reasons. Over time the road infrastructure… d₁
The fall of the Rome was largely due to an eroding political standard in the… d₂
What happened to Roman roads? q₁
Why did the Roman empire Fall? q₂
w₁,₂
w₂,₂
② q₂ traversed as relevant query to d₁
③ d₂ identified as candidate, since it's also relevant to q₁
QDG Graph Structure

Fig. 1. Query-Document Graph (QDG): A candidate document (d_1) is likely to contain similar content to another document (d_2) if they both contain information relevant to a specific query (q_2).

results, it is ultimately limited by the recall of the first stage. No matter how well a second-stage model re-ranks the retrieved documents, the model can only score documents that were retrieved by the first stage, regardless of the retriever used. Adaptive Re-Ranking (ARR) [21], presents a solution to this second-stage recall limitation by progressively adding additional documents to the candidate pool, which are similar to the documents that were re-ranked highest. Intuitively, a document estimated to be highly relevant to a specific query is likely to be more similar to other highly ranked documents than those not relevant to the query. This intuition is formally known as the Cluster Hypothesis [13]: similar documents are likely to be relevant to the same query.

In this work, we propose an alternative mechanism for identifying related documents that is based on the *converse* of the Cluster Hypothesis. Simply put, we hypothesis that: if two documents are relevant to the same query, they are likely to be similar. Following this hypothesis, we propose that similar documents can be linked to one another through queries that the documents are both relevant to. To test this hypothesis, we propose building a bipartite graph structure that maps between documents and queries that are relevant to the documents, which we call a Query-Document Graph (QDG). The nodes of the QDG represent either queries or documents, and the edges are weighted by the estimated relevance between the document and the query. Figure 1 shows an example of a QDG. Further, we propose methods of exploring a QDG that (1) enables the estimation of document-document similarity, and (2) can be used in an ARR setting to help identify additional relevant documents. Specifically, we propose two ARR methods to use the QDG during retrieval, called (i) reverted adaptive retrieval (RAR) and (ii) resource selection.

Through experiments, we find that using QDG in the ARR framework can be used in retrieval tasks to find additional relevant documents as effectively as traditional nearest neighbour lexical graphs, and obtains similar performance in MS MARCO benchmark evaluation tests. In addition, we find that our approaches outperform dense retrieval approaches in terms of the nDCG metric.

Our contributions can be summarised as follows: (1) We present a new document-similarity method based on the hypothesis that documents relevant to the same query are similar; (2) We demonstrate how this new document-similarity method can be utilised with the Cluster Hypothesis in the context of information retrieval and how it can be constructed as a bipartite graph using query generation models; (3) We demonstrate how we can use the weighted document-query bipartite graph in an Adaptive Re-ranking setting to perform retrieval in a variety of ways; (4) We demonstrate that we can use several methods of graph exploration with Adaptive Re-ranking to significantly improve upon typical re-ranking pipelines, improving the nDCG@10 metric by 5.8% compared to a state-of the art two-stage re-ranking pipeline.

2 Related Work

In adhoc retrieval tasks, neural models such as BERT, T5 and XLNET [7,28,37] can be fine-tuned to determine the relevance between queries and documents by using learned semantic embedding representation of text [18], typically outperforming retrieval models that rely on lexical term matches, such as BM25. Neural models are typically too costly to run on entire document collections due to their large size [10,14]. Therefore, neural ranking models are commonly deployed as second-stage re-rankers - they reorder a candidate subset of the whole corpus retrieved by an inexpensive (but less effective) first-stage model. An emergent limitation of such a two-stage pipeline design is that recall cannot be improved after the first stage ranking. Ind eed, the neural re-ranker can only reorder the documents retrieved by the first-stage retriever, and any relevant documents that were not retrieved in the first stage are 'lost'. This results in an efficiency-effectiveness trade-off, whereby retrieving a higher number of documents in the first stage of retrieval improves recall, but increases the computational cost of re-ranking. As a whole, the two-stage pipeline tries to approximate the performance of an end-to-end neural ranker. Indeed, there have been several attempts to reduce the computational costs using a standalone neural ranker, i.e. dense retrieval [15,19,20,36]. However, it is still generally considered an expensive procedure and comes with its own trade-offs, especially in terms of the computational requirements at indexing time and the storage requirements of the pre-computed representations.

Another technique that is typically used to improve recall after re-ranking is Pseudo-Relevance Feedback (PRF) [31]. Traditional PRF reformulates the original query with additional terms taken from the top ranked documents assumed to be relevant to the query. The first-stage ranker then uses this enhanced query to retrieve more candidate documents [1,2,12]. PRF has also been applied to improve dense retrieval [17,35,38], where feedback of the top ranked documents is expressed in the form of the embeddings [16]. Recent work in transformer-based solutions like Doc2Query [25] and DocT5Query [24] have been used to enrich the initial document representation by applying sequence-to-sequence text generation on documents to generate queries that may be relevant to the document's content. Further pre-processing steps - in particular, relevance filtering - can improve the quality of the document expansion terms [8]. Other learned sparse retrieval systems combine both document and query expansion, alongside term re-weighting [23].

An earlier idea for improving the two-stage pipeline uses generated queries specifically for PRF. Pickens *et al.* [26] proposed Reverted Indexing (RI), which builds an index of which terms or queries are most indicative of each document in a corpus. After a first pass retrieval, the terms associated to the retrieved documents can then be used as an expanded query. One of our proposed Adaptive Retrieval methods is inspired by Reverted Indexing.

In ARR [21], documents from the first-stage retrieval are re-ranked in batches. Each time a candidate batch is re-ranked, an additional discovery task is executed to find similar documents to the most relevant, highest re-ranked documents. These similar documents are obtained using a document-similarity graph, and any identified similar documents not already present in the first-stage retrieval document collection are queued to be re-ranked alongside the documents from the first-stage retrieved documents. This process is repeated, alternating between re-ranking batches from the first stage retrieval and re-ranking batches of documents similar to those already re-ranked highly, which were discovered with the similarity graph until a re-ranking budget, in terms of number of documents, has been consumed. This process is based on the principle of the Cluster Hypothesis [9,34], which states that similar documents will be relevant to the same queries. Using ARR, it is possible to identify additional documents similar to those that the re-ranker has already assessed as relevant to the user's query. This process, therefore, presents an effective method of overcoming the first stage-recall limitation as additional similar documents may not have been present in the first-stage retrieval candidate collection and shows significant improvements in recall over typical two-stage pipelines [21].

One notable limitation of the ARR framework is that the document-similarity graph is agnostic to the user's original query. The graph identifies documents similar to those already predicted as highly relevant to the query by the re-ranker. However, it does not explicitly account for the user's query in any way. In our work, we instead propose to use a bipartite graph of generated queries and documents to discover similar documents that also consider document relevance to the actual user's query. To our knowledge, only one other work [32] has attempted to directly map documents to a space of their generated queries, however they do not consider a graph-based approach. In our approach, neighbouring documents linked by generated queries can be prioritised for adaptive retrieval based on how similar their linking query in the bipartite graph is to the original user query. We present our work as a reformulation of ARR, accounting for the user query with this new graph mechanism.

3 Query-Document Graph

In this section, we introduce our idea as to how similar documents can be found through queries that documents are relevant to. We also propose that such query-document links can be developed into a bipartite graph structure where the nearest neighbour documents can be identified through connecting query nodes.

The Cluster Hypothesis can be stated using notation as follows:

$$Rel(d_1, q) \wedge Sim(d_1, d_2) \implies Rel(d_2, q) \tag{1}$$

In other words, if a document d_1 is relevant to a query q, and d_1 is similar to another document d_2, then d_2 is also likely to be relevant to q. Central to our idea of QDG, we believe that the converse of the Cluster Hypothesis statement also indicates the similarity between two documents. Specifically, if two documents are relevant to the same query, then those documents are likely to be similar:

$$Rel(d_1, q) \wedge Rel(d_2, q) \implies Sim(d_1, d_2) \tag{2}$$

More broadly, we envisage these relations between documents and queries as a bipartite graph structure, where multiple documents may be relevant to the same queries - forming multi-hop connections in the graph. Our proposed logic for this graph connecting documents and queries, and for how similar documents can be identified using the graph, which we call a Query-Document Graph (QDG), is illustrated in Fig. 1.

The sets of unique documents and unique queries of a corpus can represent two disjointed sets of vertices in a bipartite graph such that no two documents are directly connected, instead always connecting through a shared query vertex. When building such a structure, two key challenges arise. Firstly, obtaining a large set of associated relevant queries for each document in a corpus may be challenging. Secondly, we require a method of using the weighted edges of the graph so that the order of nearest neighbour documents can be calculated when a document is linked to multiple other documents through query vertices. We describe in detail how these two challenges are approached in the following two sections to build a weighted bipartite graph and how this can be used to identify nearest neighbour documents. Finally, we note that other works have also modeled query-document relationships as a bipartite graph [3,6]. However, to the best of our knowledge, we are the first to make use of this graph in an adaptive re-ranking setting.

3.1 Query Population

To build QDG for a corpus, it is necessary to first obtain a set of queries for each document and an associated relevance score is necessary to represent edge weights. For our work, we generate queries for each document using a Doc2Query generative model - generating T queries for each document in the collection [24]. We use a neural re-ranking model to provide a relevance estimate of how well-suited each generated query is to its source document. We remove duplicate queries per document and, inspired by work on Reverted Indexing [26], invert that list, such that we obtain a two-way look-up structure, i.e. document-to-query and query-to-document, with associated relevance scores that map traversals across the bipartite graph from any document or query vertex.

3.2 Ranking Neighbours

A mapping of documents and queries with associated relevance scores for each document makes it possible to explore documents connected through queries as a bipartite graph. The weights of the edges between documents and queries are associated with estimated relevance scores between the documents and queries. Indeed, between a tuple of documents $\langle d_1, d_2 \rangle$, there exists two aspects of relevance connecting them via a common generated query, q'. Hence, we can estimate the similarity between d_1 and d_2 as the product of the relevance scores along each edge, as follows:

$$Sim(d_1, d_2) = Rel(q', d_1) * Rel(q', d_2) \tag{3}$$

where $Rel(\cdot)$ is a relevance estimation function, such as a cross-encoding neural re-ranker. The intuition is that for two documents to be similar to one another, they should be similar to the same query. By scoring all documents that are two hops from a source document (d_1), we can compute a weighted ranking of the most similar documents to the source document d_1.

Equation (3) is sufficient to determine the similarity between two documents in isolation. However, in practice, we often have an additional important signal available: the user's original query q_u. In this case, it becomes more important to emphasise edges that are similar to the actual user's information need, rather than relevance to our source document. We, therefore, can further condition the similarity on the user's query, and replace the $Rel(q', d)$ component with a query similarity measure as follows:

$$Sim(d_1, d_2 \mid q_u) = Sim(q_u, q') * Rel(q', d_2) \tag{4}$$

Note that in this formulation, the source document (d_1) is ignored entirely; instead, it is only used insofar as to identify potentially-relevant queries in the graph to traverse.

4 QDG in an Adaptive Retrieval Framework

We now describe different ways a QDG can be utilised in a retrieval pipeline. To begin, we identify the Adaptive Re-ranking pipeline as a natural fit for the QDG due to its use of a corpus graph component. In the original ARR work, this pipeline retrieves a candidate set of documents with a first stage retriever, and then re-ranks batches of documents from this candidate pool. Using a corpus graph component (originally a lexical nearest neighbour graph) the pipeline identifies documents that similar to the documents in the batch to be re-ranked are added to a special pool called the frontier. The re-ranking process then continues, but batches of documents are chosen alternatively from the original candidate pool or the frontier pool, adding new documents to the frontier for every batch that is re-ranked, until a re-ranking budget is reached. This frontier helps to find documents not retrieved by the first-stage retriever and helps overcome the first-stage retriever.

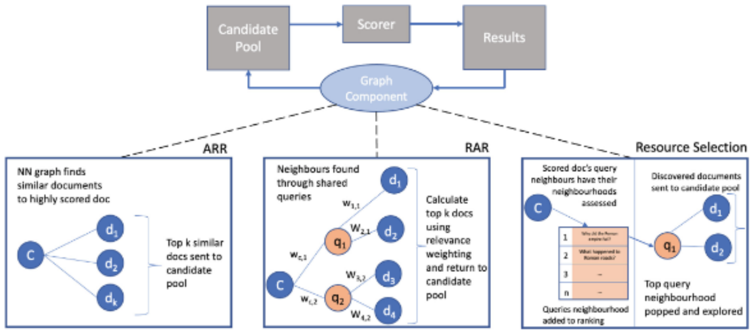

Candidate Pool
Scorer
Results
Graph Component
ARR
RAR
Resource Selection
NN graph finds similar documents to highly scored doc
Top k similar docs sent to candidate pool
Neighbours found through shared queries
Calculate top k docs using relevance weighting and return to candidate pool
Scored doc's query neighbours have their neighbourhoods assessed
Discovered documents sent to candidate pool
Queries neighbourhood added to ranking
Top query neighbourhood popped and explored

Fig. 2. The fundamental Adaptive Re-Ranking pipeline remains while the graph-based component is altered. (Left) Original Adaptive Re-Ranking approach. (Centre) Reverted Adaptive Retrieval (Right) The Resource Selection approach.

As a first idea, we propose to directly use a QDG as the corpus graph component in the Adaptive Re-ranking framework. Using the logic described in Sect. 3.2, similar documents to those already re-ranked in the adaptive re-ranking pipeline can be found with the QDG to add to the frontier, with the added benefit that the similar documents will also be identified using the original user query (as portrayed in Eq. (4)).

Secondly, we again replace the corpus graph component in the adaptive re-ranking pipeline with the QDG graph. However in this implementation, we treat each query vertex as a neighborhood of documents, i.e. all document vertices one hop away from a specific query vertex make up a neighbourhood. For a re-ranked document in the Adaptive Re-ranking pipeline, we then score, using the neural re-ranker, a small number of documents from each query vertex neighbourhood to prioritise the full query vertex neighbourhood for re-ranking in the Adaptive Re-ranking system. In this way, we can estimate the relevance of the neighbourhood of a retrieved document directly using the pipeline's re-ranker and can prioritise specific neighbourhoods of documents.

Figure 2 provides a graphical summary of our contributions. It first shows the original ARR approach using a typical nearest neighbour graph in the Adaptive Re-ranking pipeline, alongside our two proposed adaptations. As seen in Fig. 2, we essentially change the graph-based part of the ARR framework - the component that provides similar documents to those already re-ranked highly to the frontier - the rest of the Adaptive Re-ranking framework remains the same. In the remainder of this section, we describe in detail these two adaptations, Reverted Adaptive Retrieval (Sect. 4.1) and Resource Selection (Sect. 4.2).

4.1 Reverted Adaptive Retrieval

In incorporating QDG directly as the graph component of ARR, a new logic emerges for finding additional relevant documents - we call this new process

Reverted Adaptive Retrieval (RAR), inspired by the Reverted Indexing approach of Pickens *et al.* [26].

To explain RAR, we resort back to the Cluster Hypothesis (Eq. (1)) and the equation for the converse of the Cluster Hypothesis (Eq. (2)). In the case of RAR, the QDG is responsible for suggesting additional documents to score based on their similarity to the top-scoring documents seen so far. By instantiating Eq. (2) into Eq. (1) for the original Cluster Hypothesis, we can now discern the central intuition behind RAR. In particular, our intuition is that, given a document d_1 that is scored highly to the query q, both d_1 and another document d_2 are relevant to another query q', then there is a high probability that d_2 may also be relevant to the original query q, i.e.:

$$Rel(d_1, q) \wedge Rel(d_1, q') \wedge Rel(d_2, q') \implies Rel(d_2, q) \tag{5}$$

We envisage several advantages that are associated with RAR. Firstly, RAR aims to search the graph for documents that are closely related to the user's original query and not just those documents similar to the candidate document which is the case with the original ARR. Secondly, documents found using a reverted approach are directly connected to the candidate document through a related query. This allows to inspect the path in the graph that has been used to identify a document as potentially relevant, thereby providing a potential explanation for why a document was retrieved.

4.2 Resource Selection

In our first proposed retrieval framework, RAR, there is no guarantee that the neighbours added to the frontier will be relevant to the query - there is no process to identify which neighbourhoods should be explored and which neighbourhoods should be ignored, as a whole. We hypothesise that some neighbourhoods may contain no relevant documents at all and thus, a potential efficiency problem can occur where resources are used to process non-relevant documents that have been added to the frontier from low-quality neighbourhoods.

To tackle this problem, we propose a resource selection approach that prioritises entire neighbourhoods in the frontier based on an estimate of whether or not they are likely to contain relevant documents. This is schematically visualised in the right-hand box of Fig. 2. The new frontier consists of prioritised *neighbourhoods*, rather than individual documents. The process remains similar to ARR. A batch of documents is re-ranked, and then for each re-ranked document, a subset of neighbour documents for each neighbour query will also be scored. The average score of this subset of documents, per neighbour query, will be used to give each query neighbourhood a priority ranking. The central idea of this framework is that a neighbourhood with very few documents predicted to be relevant will receive a low average score from the subset of nearest neighbours. In contrast, we predict that a neighbourhood with many documents that are estimated to be relevant will score high for the subset. By prioritising the neighbourhoods based on a small scored subset, we can prioritise whole neighbourhoods that likely have many additional relevant documents. We argue this leads to a more informed selection of neighbourhoods and neighbouring documents to re-rank, and should reduce the resources used to re-rank documents that are not really relevant to the original query.

5 Experiments

We examine the effectiveness of our RAR and Resource Selection methods using the QDG by performing experiments to answer three research questions:

- **RQ1:** Does RAR bring additional relevant documents during adaptive re-ranking?
- **RQ2:** Does Resource Selection bring additional relevant documents during adaptive ranking?
- **RQ3:** Do our approaches enable earlier discovery of additional documents? i.e. can more relevant documents be found with a smaller re-ranking budget?

5.1 Experimental Setup

We address our research questions by performing retrieval experiments using the TREC Deep Learning 2019 (DL19) [4] and 2020 (DL20) [5] test collections. Our methods were developed on DL19, with DL20 serving as a held-out evaluation set. Both of these datasets use the MSMARCO passage ranking corpus consisting of 8.8 million passages. In the evaluation of our experiments, we are concerned with both precision and recall metrics, so we use nDCG, reporting the official task measure of nDCG with a rank cutoff of 10 (nDCG@10) to provide meaningful comparisons with other works, and also with cutoff 100 (nDCG@100). We also compute Mean Average Precision (MAP), and Recall at cutoff 100 (R@100).

To build a QDG for the MSMARCO corpus, we obtain a collection of 80 generated queries for each passage in MSMARCO, which were originally generated by a Doc2Query generative model [24]. We then follow the process of high-quality query generation described in Sect. 3, by using the ELECTRA cross-encoder re-ranker model [27] to estimate the relevance between a document and each generated query - inspired by the work of Gospodinov *et al.* [8] who use the same cross-encoder model for relevance assessment to identify poor quality queries generated by the Doc2query model.

We compare our methods to the original ARR approach, and with a two-stage retrieval pipeline that uses BM25 [30] – a classical lexical (sparse) retrieval model – for the first stage retrieval, and an ELECTRA cross-encoder as the second stage re-ranker, due to its superior performance to BERT-based re-rankers [27]. We also compare to two end-to-end dense retrieval approaches, ColBERT [15] & TAS-B [11]. In total, we apply six retrieval pipeline configurations:

- **MonoELECTRA**: A two-stage retrieval pipeline, where a BM25 first-stage retrieval is re-ranked by the MonoELECTRA cross-encoder, with no adaptive component.

- **ColBERT-E2E**: A state-of-the-art end-to-end dense neural ranking model that uses multiple representations per query and document.
- **TAS-B**: A state-of-the-art end-to-end dense neural ranking model that uses a single representation per query and document.
- **ARR**: This applies the original ARR framework that re-ranks documents from a BM25 initial first-stage ranking, uses a BM25 lexical graph, and Mono-ELECTRA as the re-ranker.
- **RAR**: This applies RAR (Sect. 4.1), using the QDG as the graph component in the ARR framework and using the BM25 retrieval model for first stage retrieval.
- **Resource Selection**: As above, but applying the Resource Selection (Sect. 4.2).

All retrieval pipeline configurations are implemented using PyTerrier [22], which provides pre-indexed retrieval models for both BM25 on the MSMARCO collection and a framework for building neural re-ranking pipelines. We obtain the original BM25 corpus graph structure used by the original ARR paper from their Github repository [21]. To measure query-query similarity, we use the `neeva/query2query` model obtained from the Huggingface model repository.

Finally, we follow the original ARR work when setting operational parameters. We use a re-ranking batch size of 16 for the re-ranker in all pipelines. We therefore set an overall re-ranking budget of 100, and evaluate performance with metrics at a rank cutoff of 100 or below. In line with the ARR work, we set a limit of 8 on the number of k-nearest neighbours to be returned in the both ARR and RAR pipelines. In the case of the Resource Selection pipeline, we assess the top 4 closest neighbour documents to each query neighbour documents to rank each query vertex, and retrieve the next top 8, after discounting the top 4 that were already scored in the assessment phase. The source code of our Adaptive Re-ranking pipelines and the generated queries are available at https://github.com/terrierteam/ecir2024_rar.

5.2 Results

Table 1 presents the results of different retrieval pipeline configurations evaluated on the DL19 and DL20 datasets. We first analyse the results compared to the MonoELECTRA baseline and the original ARR approach, and then compare the performance of our methods to the dense retrieval methods.

From the DL19 results, we see that the Resource Selection method performs significantly better than the MonoELECTRA baseline in terms of nDCG@10 - the only approach to do so on DL19 - achieving a 5.8% improvement. RAR and ARR also show improvements over the baseline for nDCG@10. In terms of nDCG@100 and MAP, the Resource Selection algorithm is also the best-performing pipeline. R@100 is the only metric where the Resource Selection approach is not the highest-performing for DL19. There, the ARR pipeline is the best model, achieving the best score of 0.559, a 14.5% improvement over the baseline. It is noteworthy, however, that although ARR retrieves more relevant documents, it does not necessarily identify documents it ranks highly, as

Table 1. Results for the DL19 and DL20 displaying RAR alongside ARR and other retrieval pipelines. **Bold** text indicates highest performing score in a given metric. * indicates statistical significance in the corresponding metric w.r.t. the baseline Mono-ELECTRA approach (paired t-test, $p < 0.05$).

Pipeline	TREC DL 2019				TREC DL 2020			
	NDCG@10	NDCG@100	MAP	R@100	NDCG@10	NDCG@100	MAP	R@100
MonoELECTRA	0.706	0.565	0.365	0.488	0.698	0.588	0.417	0.584
w/ ARR	0.724	0.607*	0.405*	**0.559***	0.724*	0.620*	0.440*	0.615
w/ RAR	0.740	0.608*	0.400	0.543*	**0.744***	**0.642***	**0.481***	0.665*
w/ Resource Selection	**0.747***	0.623*	0.417*	0.551*	0.726*	0.625*	0.446*	0.622*
Dense Retrieval Baselines								
ColBERT E2E	0.693	0.602	**0.431**	0.578	0.687	0.626	0.465	0.710
TAS-B	0.716	**0.636**	0.405	0.609	0.684	0.632	0.449	**0.713**

evidenced by its lower nDCG scores. Both RAR and Resource selection also improve recall significantly compared to the baseline, improving by 11.3% and 12.9%, respectively.

For DL20, the RAR pipeline achieves the best performance in all metrics: for the nDCG@10 metric, it achieves a 6.5% improvement over the baseline; for nDCG@100 a 9.2% improvement; for MAP a 15.3%; and finally for R@100, RAR achieves a 13.8% improvement - all of which are statistically significant. Likewise, our Resource Selection pipeline also shows statistical significance over the MonoELECTRA baseline on all metrics. The ARR pipeline shows a significant improvement in the nDCG@10, nDCG@100 and MAP metrics, however, it is not statistically significant in the R@100 metric w.r.t. the baseline - though it does offer improved performance in this metric.

Across both tasks, DL19 and DL20, the RAR and Resource selection pipelines consistently achieve higher recall than the MonoELECTRA baseline method - though only our two methods are significantly better than the baseline for R@100 on the DL20 queryset. This indicates that the Adaptive Re-Ranking approaches are capable of overcoming the recall problem associated with the first stage retrieval of the baseline pipeline.

Comparing our methods to the dense retrieval methods, we observe that for DL19, both Resource selection and RAR outperform both of the dense retrieval methods in terms of nDCG@10. Our best performing approach, Resource Selection, improves on the best performing dense approach, TAS-B, by 4.3%. Both our approaches also beat ColBERT-E2E for nDCG@100 (but TAS-B performs best overall for that measure). In terms of the recall, the dense retrieval approaches outperform all others. This is also the case for the recall metric on the DL20 dataset. However, in DL20, for every other metric, we observe that RAR outperforms the dense retrieval methods. Here, for the nDCG@10 metric, Resource Selection also outperforms the dense retrieval methods. To summarise, while the dense retrieval approaches recall more relevant documents, they do not necessarily rank these documents highly. Our approaches provide a better trade-off in this regard, as indicated by the observed nDCG values.

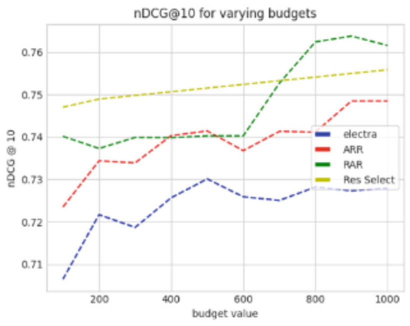


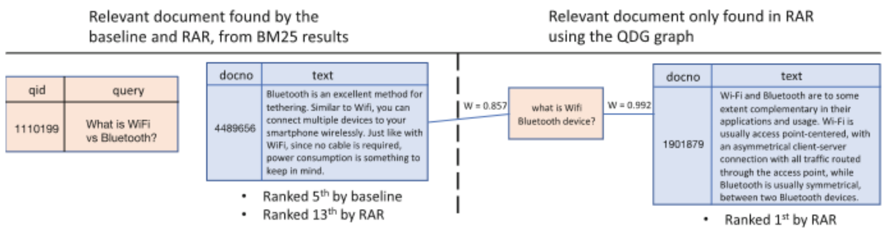
Fig. 3. nDCG@10 scores of each retrieval pipeline performing the DL19 search task across at different re-ranking budget values ranging from 100 to 1000.

We can now answer RQ1 and RQ2; both RAR and Resource Selection find additional relevant documents during retrieval, compared to the baseline MonoELECTRA two-stage re-ranker pipeline. We find that our methods perform better than ARR and the MonoELECTRA baseline for nDCG@10 in DL19, and better in all metrics than ARR in DL20.

Figure 3 shows a line graph for four of the pipelines we experimented with. It shows how nDCG@10 varies (y-axis) as the re-ranking budget (x-axis) varies from 100 to 1000 in increments of 100 for each pipeline. The values to the far left of each plot are those already displayed in Table 1. From Fig. 3, we see that both of our approaches, RAR and Resource Selection, achieve the highest nDCG@10 scores at a re-ranking budget of 100. This indicates that more relevant documents are found earlier in the retrieval process when using our approaches. Notably, our Resource Selection approach (denoted Res Select in Fig. 3) achieves the highest nDCG@10 for most re-ranking budgets below 1000, up to a budget of 700. From this point, RAR achieves the best performance from budgets of 700 to 1000. In answering RQ3, our approaches showed improved nDCG@10 scores at low re-ranking budgets compared to the MonoELECTRA re-ranking baseline and the original ARR approach. RAR and Resource selection can indeed find more relevant documents than other approaches sooner, providing the re-ranker component with more relevant documents to score.

6 Qualitative Analysis

As mentioned in Sect. 4.1, since the QDG connects document nodes through related queries, it is possible to trace back the path to determine why a document was retrieved. In this section, we demonstrate this through a qualitative example. Fig. 4 proves an example where RAR improves retrieval performance over the baseline approach in the DL19 task. One relevant document (4489656) was retrieved by both the RAR and baseline retrieval pipelines, while a second document (1901879) was only retrieved by the RAR pipeline (i.e., it was not retrieved by BM25). Instead, it was retrieved using the QDG, since it is


Relevant document found by the baseline and RAR, from BM25 results
Relevant document only found in RAR using the QDG graph
W = 0.857
what is Wifi Bluetooth device?
W = 0.992
• Ranked 5th by baseline
• Ranked 13th by RAR
• Ranked 1st by RAR

qid	query
1110199	What is WiFi vs Bluetooth?

docno	text
4489656	Bluetooth is an excellent method for tethering. Similar to Wifi, you can connect multiple devices to your smartphone wirelessly. Just like with WiFi, since no cable is required, power consumption is something to keep in mind.

docno	text
1901879	Wi-Fi and Bluetooth are to some extent complementary in their applications and usage. Wi-Fi is usually access point-centered, with an asymmetrical client-server connection with all traffic routed through the access point, while Bluetooth is usually symmetrical, between two Bluetooth devices.

Fig. 4. An example RAR graph traversal that resulted in improved effectiveness.

closely connected to the first document through a query ("what is Wifi Bluetooth device?") which both documents are estimated to be highly relevant to. This trace could help explain to a user or researcher why the document was retrieved, for instance, by stating that the document is relevant to a query that is similar to that entered by the user.

7 Conclusions

Overcoming the recall problem is a central issue for re-ranking systems. Although the recently-proposed Adaptive Re-Ranking approach helps address the issue, its document similarity measure is agnostic of the query, which can lead to related-yet-non-relevant documents being scored in the re-ranking process. Drawing on intuitions based on the converse of the Cluster Hypothesis, we proposed two alternative approaches for identifying potentially similar documents that is conditioned on the user's original query by leveraging a new index structure – a query-document graph – that links documents and their generated queries. Through experimentation, we successfully used these approaches in two Adaptive Re-Ranking settings, which ultimately yielded improved precision of the top results and improved overall recall. We also found that the process identifies relevant documents earlier in the re-ranking process, potentially reducing the number of documents that need to be re-ranked to achieve high effectiveness.

Our approach has several limitations to address in future work, however. First, it relies on a query generation process during indexing time, which is known to be computationally expensive [33]. Future work could explore how to reduce this burden through techniques like relevance filtering [8] or by using latent (rather than text-based) query representations. The approach also adds computational overhead to the neighbourhood lookup process that cannot be pre-computed, since it relies on the user's current query. We expect that clustering or quantisation approaches could reduce this computational burden. Nevertheless, this work fills an important research gap in both index-time modelling of query-document relationships and overcoming limitations of Adaptive Re-Ranking systems.

Acknowledgement. We acknowledge EPSRC grant EP/R018634/1: Closed-Loop Data Science for Complex, Computationally- & Data-Intensive Analytics. We thank the anonymous reviewers for their helpful feedback on this manuscript.

References

1. Amati, G., Carpineto, C., Romano, G.: Query difficulty, robustness, and selective application of query expansion. In: Advances in Information Retrieval - 26th European Conference on Information Retrieval, pp. 127–137 (2004)
2. Amati, G., Van Rijsbergen, C.J.: Probabilistic models of information retrieval based on measuring the divergence from randomness. ACM Trans. Inf. Syst. (TOIS) **20**(4), 357–389 (2002)
3. Boldi, P., Bonchi, F., Castillo, C., Donato, D., Gionis, A., Vigna, S.: The query-flow graph: model and applications. In: Proceedings of the 17th ACM Conference on Information and Knowledge Management, pp. 609–618 (2008)
4. Craswell, N., Mitra, B., Yilmaz, E., Campos, D., Voorhees, E.M.: Overview of the TREC 2019 deep learning track. In: Proceedings of the Twenty-Eighth Text REtrieval Conference (2019)
5. Craswell, N., Mitra, B., Yilmaz, E., Campos, D., Voorhees, E.M., Soboroff, I.: TREC deep learning track: Reusable test collections in the large data regime. In: Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 2369–2375 (2021)
6. Craswell, N., Szummer, M.: Random walks on the click graph. In: Proceedings of the 30th International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 239–246 (2007)
7. Devlin, J., Chang, M., Lee, K., Toutanova, K.: BERT: pre-training of deep bidirectional transformers for language understanding. In: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT, pp. 4171–4186 (2019)
8. Gospodinov, M., MacAvaney, S., Macdonald, C.: Doc2Query–: when less is more. In: Advances in Information Retrieval - 45th European Conference on Information Retrieval, pp. 414–422 (2023)
9. Hearst, M.A., Pedersen, J.O.: Reexamining the cluster hypothesis: scatter/gather on retrieval results. In: Proceedings of the 19th International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 76–84 (1996)
10. Hofstätter, S., Hanbury, A.: Let's measure run time! extending the IR replicability infrastructure to include performance aspects. In: Proceedings of the Open-Source IR Replicability Challenge co-located with 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 12–16 (2019)
11. Hofstätter, S., Lin, S., Yang, J., Lin, J., Hanbury, A.: Efficiently teaching an effective dense retriever with balanced topic aware sampling. In: Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 113–122 (2021)
12. Jaleel, N.A., et al.: UMass at TREC 2004: novelty and HARD. In: Proceedings of the Thirteenth Text REtrieval Conference (2004)
13. Jardine, N., van Rijsbergen, C.J.: The use of hierarchic clustering in information retrieval. Inf. Storage Retr. **7**(5), 217–240 (1971)
14. Johnson, J., Douze, M., Jégou, H.: Billion-scale similarity search with GPUs. IEEE Trans. Big Data **7**(3), 535–547 (2019)

15. Khattab, O., Zaharia, M.: ColBERT: efficient and effective passage search via contextualized late interaction over BERT. In: Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 39–48 (2020)
16. Li, C., et al.: NPRF: a neural pseudo relevance feedback framework for ad-hoc information retrieval. In: Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pp. 4482–4491 (2018)
17. Li, H., Zhuang, S., Mourad, A., Ma, X., Lin, J., Zuccon, G.: Improving query representations for dense retrieval with pseudo relevance feedback: a reproducibility study. In: Advances in Information Retrieval - 44th European Conference on Information Retrieval, pp. 599–612 (2022)
18. Lin, J., Nogueira, R.F., Yates, A.: Pretrained Transformers for Text Ranking: BERT and Beyond. Morgan & Claypool Publishers, San Rafael (2021)
19. MacAvaney, S., Nardini, F.M., Perego, R., Tonellotto, N., Goharian, N., Frieder, O.: Efficient document re-ranking for transformers by precomputing term representations. In: Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 49–58 (2020)
20. MacAvaney, S., Nardini, F.M., Perego, R., Tonellotto, N., Goharian, N., Frieder, O.: Expansion via prediction of importance with contextualization. In: Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 1573–1576 (2020)
21. MacAvaney, S., Tonellotto, N., Macdonald, C.: Adaptive re-ranking with a corpus graph. In: Proceedings of the 31st ACM International Conference on Information & Knowledge Management, pp. 1491–1500 (2022)
22. Macdonald, C., Tonellotto, N.: Declarative experimentation in information retrieval using PyTerrier. In: Proceedings of the 2020 ACM SIGIR International Conference on the Theory of Information Retrieval, pp. 161–168 (2020)
23. Nguyen, T., MacAvaney, S., Yates, A.: A unified framework for learned sparse retrieval. In: Advances in Information Retrieval - 45th European Conference on Information Retrieval, pp. 101–116 (2023)
24. Nogueira, R., Lin, J.: From doc2query to docTTTTTquery (2019). https://cs.uwaterloo.ca/~jimmylin/publications/Nogueira_Lin_2019_docTTTTTquery-v2.pdf
25. Nogueira, R.F., Yang, W., Lin, J., Cho, K.: Document expansion by query prediction. CoRR abs/1904.08375 (2019)
26. Pickens, J., Cooper, M., Golovchinsky, G.: Reverted indexing for feedback and expansion. In: Proceedings of the 19th ACM Conference on Information and Knowledge Management, pp. 1049–1058 (2010)
27. Pradeep, R., Liu, Y., Zhang, X., Li, Y., Yates, A., Lin, J.: Squeezing water from a stone: a bag of tricks for further improving cross-encoder effectiveness for reranking. In: Advances in Information Retrieval - 44th European Conference on Information Retrieval, pp. 655–670 (2022)
28. Raffel, C.: Exploring the limits of transfer learning with a unified text-to-text transformer. J. Mach. Learn. Res. **21**(1), 5485–5551 (2020)
29. Robertson, S.E., Walker, S., Jones, S., Hancock-Beaulieu, M., Gatford, M.: Okapi at TREC-3. In: Proceedings of the Third Text REtrieval Conference, pp. 109–126 (1994)
30. Robertson, S.E., Zaragoza, H.: The probabilistic relevance framework: BM25 and beyond. Found. Trends Inf. Retr. **3**(4), 333–389 (2009)
31. Rocchio Jr, J.J.: Relevance feedback in information retrieval. The SMART retrieval system: experiments in automatic document processing (1971)

32. Salamat, S., Arabzadeh, N., Zarrinkalam, F., Zihayat, M., Bagheri, E.: Learning query-space document representations for high-recall retrieval. In: Advances in Information Retrieval - 45th European Conference on Information Retrieval, pp. 599–607 (2023)
33. Scells, H., Zhuang, S., Zuccon, G.: Reduce, reuse, recycle: green information retrieval research. In: Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 2825–2837 (2022)
34. Voorhees, E.M.: The cluster hypothesis revisited. In: Proceedings of the 8th International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 188–196 (1985)
35. Wang, X., Macdonald, C., Tonellotto, N., Ounis, I.: Pseudo-relevance feedback for multiple representation dense retrieval. In: Proceedings of the 2021 ACM SIGIR International Conference on the Theory of Information Retrieval, pp. 297–306 (2021)
36. Xiong, L., et al.: Approximate nearest neighbor negative contrastive learning for dense text retrieval. In: 9th International Conference on Learning Representations (2021)
37. Yang, Z., Dai, Z., Yang, Y., Carbonell, J.G., Salakhutdinov, R., Le, Q.V.: XLNet: generalized autoregressive pretraining for language understanding. In: Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, pp. 5754–5764 (2019)
38. Yu, H., Xiong, C., Callan, J.: Improving query representations for dense retrieval with pseudo relevance feedback. In: Proceedings of the 30th ACM International Conference on Information & Knowledge Management, pp. 3592–3596 (2021)