RE-PACRR: A Context and Density-Aware Neural Information Retrieval Model

Kai Hui
Max Planck Institute for Informatics
Saarbrücken Graduate School of Computer Science
khui@mpi-inf.mpg.de

Andrew Yates
Max Planck Institute for Informatics
ayates@mpi-inf.mpg.de

Klaus Berberich
Max Planck Institute for Informatics
htw saar
kberberi@mpi-inf.mpg.de

Gerard de Melo
Rutgers University
New Brunswick
gdm@demelo.org

ABSTRACT

Ad-hoc retrieval models can benefit from considering different patterns in the interactions between a query and a document, effectively assessing the relevance of a document for a given user query. Factors to be considered in this interaction include (i) the matching of unigrams and ngrams, (ii) the proximity of the matched query terms, (iii) their position in the document, and (iv) how the different relevance signals are combined over different query terms. While previous work has successfully modeled some of these factors, not all aspects have been fully explored. In this work, we close this gap by proposing different neural components and incorporating them into a single architecture, leading to a novel neural IR model called RE-PACRR. Extensive comparisons with established models on Trec Web Track data confirm that the proposed model yields promising search results.

CCS CONCEPTS

•**Information systems** →**Retrieval models and ranking;** *Web searching and information discovery;*

1 INTRODUCTION

Deep learning has enjoyed enormous success in the last few years, completely transforming fields such as natural language processing and computer vision. While state-of-the-art neural models for ad hoc information retrieval (IR) have been making significant advances and now perform quite well, there have been decades of prior research on information retrieval, and many of the resulting domain insights are yet to be incorporated into the neural IR paradigm. Deep learning allows us to learn powerful models that can capture relevance matching at fine-grained levels. For example, when encoding local matching, the filters in a convolutional neural network (CNN) enable a model to learn different patterns of matching beyond exact matches as in [9, 14]. The state-of-the-art neural models have demonstrated the utility of such capabilities. We argue that, however, there is still room to further exploit such insights and knowledge of the task in order to develop more powerful neural models.

To close this gap, we highlight several promising aspects based on past research within the IR community, and propose several deep neural components to address them accordingly. These components are integrated to form a novel neural model called RE-PACRR, short for Relevance Enhanced PACRR, which builds on PACRR [9], a state-of-the-art neural model. However, the insights we identify and components we propose are not specific to PACRR, and we expect them to be similarly relevant to other state-of-the-art neural models, such as DUET [11], DRMM [5], and MatchPyramid [14]. An overview of RE-PACRR is given in Figure 1, incorporating the following domain insights.

Disambiguation refers to distinguishing different senses of the same term. For retrieval models such as BM25 that rely on exact term matching, some of the matches may actually result from different senses of the same term in a query and a document. Such sense mismatches are incorrectly considered as relevance matches. For example, given the query "*Jaguar SUV price*", the term "Jaguar" refers to a car brand, but the term "jaguar" could also refer to an animal, as in the following sentence.

> *In a safari wild park near London, an attack happened when a jaguar was irritated by a horn from a tourist SUV, which, fortunately, caused no harm.*

Therefore, when evaluating the relevance between this sentence-query pair, the exact term match of "jaguar" should not be counted as a relevance match. Models such as DUET (in its distributed component) may be able to account for this by encoding a query and a document into a dense vector space. Most neural IR models do not explicitly account for disambiguation. In this work, a context checking module is proposed and plugged into the PACRR model, allowing the model to consider sense matching as well as term matching.

Proximity takes into account information about where query terms occur in a document and how they depend on each other – as exploited by retrieval models aware of term proximity [17]. Term proximity emphasizes the frequent co-occurrence of different query terms within a small text window, which are not necessarily successive. Intuitively, a small text window including multiple different query terms is more likely to be relevant in the sense that it covers more relevant information to the query. For example,

SIGIR 2017 Workshop on Neural Information Retrieval (Neu-IR'17), August 7–11, 2017, Shinjuku, Tokyo, Japan

DOI: 10.1145/nnnnnnn.nnnnnnn

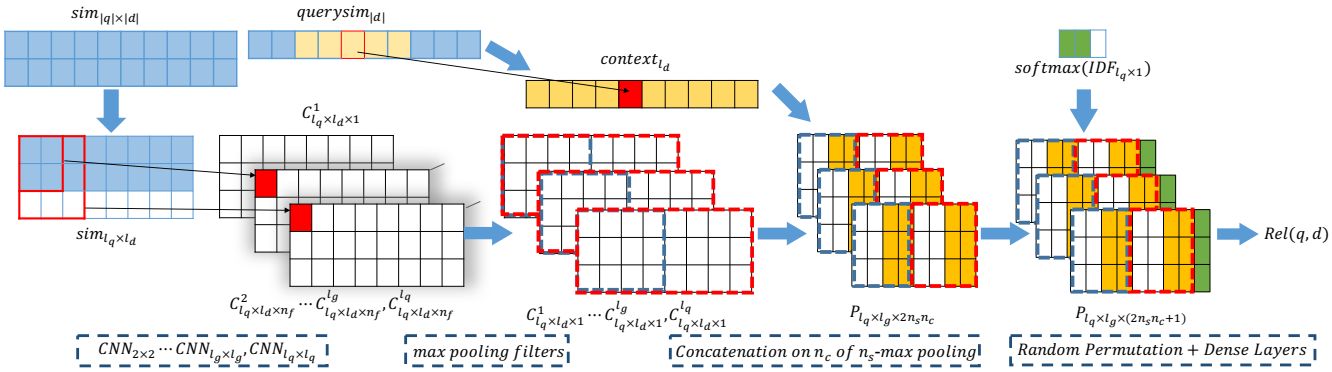

Figure 1: The pipeline of RE-PACRR. The inputs include two matrices, namely, $sim_{|q|\times|d|}$ and $querysim_{|d|}$, recording the similarity of different terms in a document relative to each query term and the whole query respectively. All these similarity matrices are truncated/zero-padded to the dimensions governed by l_q and l_d. Several 2D convolutional kernels are first applied to the similarity matrices, one for each $l_g \in [2, l_g]$, and an extra one for query proximity with kernel size $l_q \times l_q$. Next, max pooling is applied to pool out the filters, ending up with $l_g + 1$ matrices, namely, $C^1 \cdots C^{l_g}, C^{l_q}$. Following this, n_s-max pooling captures the strongest n_s signals on each C, at all n_c positions from $cpos$. In the meantime, the context similarity corresponding to each term in top-n_s from $context_{l_d}$ is also appended, leading to $P_{l_q\times l_g\times(2n_sn_c)}$. Finally, the query terms' normalized IDFs are appended, and a feedforward network is applied, after permuting the rows in $P_{l_q\times l_g\times(2n_sn_c+1)}$, getting a query-document relevance score $rel(q,d)$. In this plot, the situation when $l_d = 8$, $l_q = 3$, $l_g = 3$, $n_c = 2$, $n_s = 2$ and $cpos = [50\%, 100\%]$ is displayed.

given the query "*Jaguar SUV price*", a text window including all three terms is more likely to be relevant, since both "Jaguar SUV" and "price" are included. To consider proximity, a model needs to be capable of capturing local matching information beyond a single query term. In this regard, it has been partially considered in MatchPyramid [14] and PACRR [9], but these models do not consider proximity of all query terms at once. In this work, we further investigate a new building block to model proximity by including a sufficiently large kernel to cover all query terms, which makes it possible to fully model proximity.

Query coverage aims at comprehensively catering to all relevant concepts in the query, to avoid biasing the results towards only individual concepts occurring in the query. Consider, for instance, a verbose query such as the following [6]:

> *Provide information on all kinds of material international support provided to either side in the Spanish Civil War.*

Relevant documents should likely mention both "Spanish" and "civil war" in this case. More generally, a relevant document to a query should, to some degree, cover all of the query's most salient concepts. Existing neural IR models satisfy query coverage when combining the matching signals from different query terms. We argue that such a combination should be position-independent. Put differently, the way to combine the relevance signals and the weights allocated to different query terms should only depend on the inherent properties of the query terms and their relevance signals. Such matching signals already cover both unigrams and ngrams (e.g., in PACRR the matching signal for the term "civil" includes both the unigram signal and bigram matching signal for "civil war"), so the relative positions of these matching signals need not be considered when the signals are combined. This is especially true when the inputs are zero-padded in the tail of a query, because a model is not supposed to learn combination weights based on the occurrences of zeros in the tail positions. In this regard, we argue that position-independence is not fully implemented in established model architectures. To address this, we introduce a random permutation phase before combining signals, removing the position-dependent factors and thereby improving the generalization ability of the model.

Cascade reading model. In addition to the amount of relevant information in a document, a model should also consider the density of the relevant information. An extreme example is the comparison between one relevant document and a pseudo-document composed of several concatenated non-relevant documents with the same relevant document appended to the end. Although the same amount of relevance matches are included in the original relevant document and the pseudo-document, one would expect that the original document is preferred, due to the relevant information being presented earlier and overall more densely. Beyond this, according to the cascade model from [2], a document's relevance gains should be degraded when enough relevant information has already been observed, since a user may already stop reading after collecting enough relevant information from the current document. Such signals might be learned implicitly by a $1 \times doclen$ kernel in the local matching of DUET. However, to the best of our knowledge, it has not been explicitly modeled in established neural IR models. To address this, we propose a cascade component to model the position of the relevant signals in addition to the total amount of relevant signals.

Contributions. Inspired by these insights, we propose the novel RE-PACCRR model that incorporates all of the aforementioned building blocks: the context checking module, the term proximity

kernel, the signal combination regularizer, and the cascade component. The proposed model is examined through extensive experiments based on relevance judgments from TREC, and compared with multiple state-of-the-art models including MatchPyramid, DRMM, the local model in DUET, and the PACRR model. We find that RE-PACRR compares favorably against the baseline models. Remarkably, when re-ranking the search results from a naïve ranker, namely a query-likelihood ranking model, the re-ranked runs are ranked top-1 under both ERR20 and nDCG20 in the Web Track 2012–14, improving both evaluation measures by 100% on average. We release our implementation of RE-PACRR to support further comparisons.[1]

Organization. The rest of this paper unfolds as follows. Section 2 recaps the basic neural-IR model PACRR, and thereafter Section 3 describes the proposed building components in detail. The setup and results of our extensive experimental evaluation can be found in Section 4. We discuss related work in Section 5, before concluding in Section 6.

2 BACKGROUND

In this section we introduce the basic PACRR model following the notation from [9]. In general, PACRR takes a similarity matrix between a query q and a document d as input, and the output of the model is a scalar, namely, $rel(d, q)$, indicating the relevance of document d to query q. PACRR attempts to model query-document interactions based on these similarity matrices. At training time, the relevance scores for one relevant and one non-relevant document, denoted as d^+ and d^-, respectively, are fed into a max-margin loss as in Eq. 1.

$$\mathcal{L}(q, d^+, d^-; \Theta) = max(0, 1 - rel(q, d^+) + rel(q, d^-)) \quad (1)$$

At inference time, given a query, documents are ranked based on their $rel(d, q)$. In the following, PACRR is introduced component-by-component.

(1) **Input**: the similarity matrix $sim_{l_q \times l_d}$, where both l_q and l_d are hyper-parameters unifying the dimensions of the input similarity matrices. l_q is set to the longest query length, and the maximum document length hyper-parameter l_d is tuned on the validation dataset. Given the settings for both l_q and l_d, a similarity matrix between a query and a document is truncated if the document is too long or zero-padded if it is too short.

(2) **CNN kernels and max-pooling layers for the filters**: multiple CNN kernels with l_f filters are introduced to capture the local matching information, like n-gram matching, for different text window lengths, namely $2, 3, \cdots, l_g$. The hyper-parameters l_g and l_f govern the longest text window under consideration and the number of filters, respectively. These CNN kernels are followed by a max-pooling layer to retain the strongest signals per kernel, leading to l_g matrices, denoted as

$$C^1_{l_q \times l_d \times 1} \cdots C^{l_g}_{l_q \times l_d \times 1} ,$$

(3) **k-max pooling**: subsequently, the matching signals in $C^1, \cdots, C^{l_g}$ from these kernels are further pooled with n_s-max pooling layers, only keeping the top-n_s strongest signals over individual query terms, ending up with

$$P^1_{l_q \times n_s}, \cdots, P^{l_g}_{l_q \times n_s} ,$$

which are further concatenated for individual query terms, resulting in a matrix $P_{l_q \times (l_g n_s)}$;

(4) **combinations of signals from different query terms**: the signals in $P_{l_q \times (l_g n_s)}$, together with the inverse document frequency for individual query terms, are fed into an LSTM layer to generate the final query-document relevance score $rel(d, q)$.

Tweaks. Before moving on, we make two changes to the original PACRR model to cater to our study. This revised model is employed and simply denoted as PACRR in the following sections. First, according to our pilot experiments, the performance of the model does not change significantly when replacing the LSTM layer with a stack of dense layers, which have been demonstrated to be able to simulate an arbitrary function [4]. Such dense layers can easily learn in parallel, leading to faster training [4], whereas back-propagation through an LSTM layer is much more expensive due to its sequential structures. From Section 4, it can be seen that efficiency is very important for this study, due to the huge amount of model variants to be trained and the limited availability of hardware at our disposal. Finally, another tweak is to switch the max-margin loss in [9] to a cross-entropy loss as in Eq. 2, according to [3], where it has been demonstrated that a cross-entropy loss may lead to better performance.

$$\mathcal{L}(q, d^+, d^-; \Theta) = -log \frac{\exp(rel(q, d^+))}{\exp(rel(q, d^+)) + \exp(rel(q, d^-))} \quad (2)$$

3 METHOD

In this section, we describe the novel components in the RE-PACRR model. The architecture of this model is summarized in Figure 1.

Context checking module: disambiguation of the relevance matching. Beyond the $sim_{l_q \times l_d}$ from PACRR, we introduce an input vector denoted as $querysim_{|d|}$, encoding the similarity between each document term and the entire query. In particular, the vector of a query is computed by averaging the word vectors of all query terms, and cosine similarity is computed between each term vector in the document relative to this query vector. As in [9], we use pre-trained word2vec[2] embeddings to compute term similarities due to its availability. In the future, one may desire to replace this with other embeddings such as the dual embedding from [12] or the relevance-based embedding from [18]. Thereafter, given a term at position i, namely, $querysim[i]$, the similarity for its context is computed by averaging the similarity in its surrounding windows, resulting in $context_{l_d}$ after truncating or zero-padding in accordance with l_d. Thus, for a term at position i in a document, its context similarity $context[i]$ is computed as

$$context[i] = \frac{\sum_{j \in [i-w, i+w]} querysim[i]}{2 * w + 1} .$$

In the model, to avoid relevance matching due to ambiguity as in the "jaguar" example from Section 1, intuitively, only when

[1] https://github.com/khui/repacrr

[2] https://code.google.com/archive/p/word2vec/

both the term and its context are highly similar with a query term (or n-gram in the query) do matching signals represent a correct relevance match. Thus, when combining the top-n_s signals from individual query terms, the corresponding similarities for these top-n_s signals from $context_{l_d}$ are also concatenated, making the matrices $P_{l_q \times (l_g n_s)}$ become $P_{l_q \times (2 l_g n_s)}$. This enables the aggregating model, namely, a feed-forward network, to take the ambiguity into account when determining the ultimate score. For example, in the "jaguar" example in Section 1, in the context of "jaguar" there are "safari" and "wild park", which have low similarity with query terms such as "SUV" and "price", effectively making the model understand that the two "jaguar" occurrences in the query and in the sentence are referring to different senses.

Model proximity with a larger CNN kernel. As discussed in Section 1, we assume that proximity could be modeled over the entire query, instead of over multiple small text pieces in the query, as in MatchPyramid and PACRR. Thus we propose to implement an extra kernel with size $l_q \times l_q$, adding it along with other CNN kernels. This leads to an extra matrix after the CNN layer, namely $C^{l_q}_{l_q \times l_d \times l_f}$, and ultimately an extra matrix after the max-pooling of filters, namely, $C^{l_q}_{l_q \times l_d \times 1}$. Thereafter, such proximity signals are passed on to the follow-up pipeline in the model, resulting in a matrix $P_{l_q \times ((l_g+1) n_s)}$ before the combination.

Cascade k-max: simulating the cascade reading approach. As discussed in Section 1, not only the strength but also the offsets of relevance signals matter. We propose to encode such cascade factors by conducting k-max pooling at multiple offsets in a document, instead of pooling only on the entire document. For example, one could conduct multiple k-max pooling at 25%, 50%, 75%, and 100% of a document, ending up with $P_{l_q \times (4 l_g n_s)}$. This corresponds to when a user sifts through a document and evaluates its relevance after finishing the first, second, third, or fourth quarter of the document. The list of offsets at which cascade k-max pooling is conducted is governed by an array *cpos*, e.g., $cpos = [25\%, 50\%, 75\%, 100\%]$ in the above example, and the length of this array is denoted as n_c, which are both hyper-parameters.

Randomly permute query terms before feed-forward layer: improving generalization. As mentioned in Section 1, the combination of relevance signals among different queries is position-independent. In light of this, we propose to randomly permute rows in $P_{l_q \times (l_g n_s)}$ before aggregating them. Note that each row contains signals for multiple n-gram lengths; permuting the rows does not prevent the model from recognizing n-grams. We argue that, taking advantage of this independence, the permutation can effectively improve the generalization ability of the model, making the computation of the relevance scores depend solely on the importance of a query term (idf) and the relevance signals aggregated on it. This should be particularly helpful when training on short queries ($|q| < l_q$), where padded zeros are in the tail of $sim_{l_q \times l_d}$. Without permutation, a model might remember that the relevance signals at the tail of a query, e.g., the several final rows in $sim_{l_q \times l_d}$, contribute very little and are mostly zero, leading to it mistakenly degrade the contribution from terms at tail positions when inferring relevance scores for longer queries.

4 EVALUATION

In this section, we empirically compare the proposed RE-PACRR with multiple state-of-the-art neural IR models using manual relevance judgments from six years of the Trec Web Track. Akin to [9], the comparison is based on three benchmarks, namely, re-ranking search results from a simple initial ranker, coined RerankSimple, re-ranking all runs from the Trec Web Track, coined RerankALL, and further examining the classification accuracy of the neural IR models in determining the order of document pairs, coined PairAccuracy.

We compare our model with multiple state-of-the-art neural IR models including DRMM [5], DUET [11], MatchPyramid [14], and the basic PACRR model [9]. As our focus is on the deep relevance matching model, we only compare against DUET's local model, denoted as DUETL. Additionally, the Trec benchmarks contain much less data than that used to train DUET's distributed model in [11].

4.1 Experimental Setup

We rely on the 2009–2014 Trec Web Track ad-hoc task benchmarks[3], which are based on the ClueWeb09 and ClueWeb12 datasets as document collections. In total, there are 300 queries and more than 100k judgments (qrels). Three years (2012–14) of query-likelihood baselines (*QL*) (Terrier [13] version without filtering spam) provided by Trec[4], and the search results from runs submitted by participants from each year are both employed as initial rankings in the RerankSimple and RerankALL benchmarks, respectively. In total, there are 71 (2009), 55 (2010), 62 (2011), 48 (2012), 50 (2013), and 27 (2014) runs. ERR@20 [1] and nDCG@20 [10] are employed as evaluation measures, and both are computed with the script from Trec[5].

Training. Models are trained and tested in a round-robin manner, using individual years as training, validation and test data. Specifically, the available judgments are considered in accordance with the individual years of the Web Track, with 50 queries per year. Proceeding in a round-robin manner, we report test results on one year by using combinations of every four years and the left-out year in the remaining years for training and validation, respectively. Model parameters and the number of training epochs are chosen by maximizing the ERR20 on the validation set for each training/validation combination separately. Thereafter, the selected model is used to make predictions on the test data. Hence, for each test year, there are five different predictions each from a training and validation combination. Akin to the procedure in cross-validation, we report the average of these five test results as the ultimate result for individual test years, and conduct a Student's t-test over them to determine whether there is a statistically significant difference between different methods. For example, a significant difference between two evaluated methods on a particular test year is claimed if there exists a significant difference between the two vectors with five scores for individual methods. We argue that, this way, the effects of the choice of training and validation data are minimized. This was motivated by an observation that the closeness of the

[3] http://trec.nist.gov/tracks.html
[4] https://github.com/trec-web/trec-web-2014
[5] http://trec.nist.gov/data/web/12/gdeval.pl

subsets for training and for validation can adversely influence the model selection. For example, when using a validation set containing queries from the same Trec years as in the training set, a model could sometimes over-fit both the training and validation sets at the same time, leading to poor results on the test set.

Choice of the hyper-parameters. For the new components in RE-PACRR, we fix the size of the context window as 4 on both sides, leading to a context vector computed by averaging 9, namely, 4+4+1, surrounding similarity values in *querysim*. For the cascade component, we conduct k-max pooling with $cpos = [25\%, 50\%, 75\%, 100\%]$. Apart from the two modifications mentioned in Section 2, we fixed the model choices for PACRR, focusing on the proposed novel model variants. In particular, the PACRR-firstk variant is employed, fixing the unified dimensions $l_d = 800$ and $l_q = 16$, the k-max pooling size $n_s = 3$, the maximum n-gram size $l_g = 3$, and the number of filters used in convolutional layers $n_f = 32$. Beyond that, we fix the batch size to 16 and we train individual models to at most 150 epochs. We train on servers with multiple CPU cores; training RE-PACRR takes approximately 90 seconds per epoch. DRMM ($\text{DRMM}_{LCH\times IDF}$), DUETL, and MatchPyramid [14] are trained similarly. The size of the dense layer for DRMM is fixed to 5 following the setting in [5]; the document dimension for both DUETL and MatchPyramid is also set to 800. All these methods are trained with a cross-entropy loss as summarized in Eq. 2.

4.2 Results for RE-PACRR

RerankSimple. We first examine how well the proposed model performs when re-ranking the search results from a basic initial ranker, the query-likelihood model. The ultimate quality of the re-ranked search results depends on both the strength of the initial ranker and the quality of the re-ranker. The query-likelihood model, as one of the most widely used retrieval models, is used due to its efficiency and practical availability, given that it is included in most retrieval toolkits as a default model. The results are summarized in Table 1. The absolute scores in terms of ERR@20 and nDCG@20 are reported, together with the improvements relative to the measures on search results from the query-likelihood model. The ranks of the re-ranked runs are also reported when sorting the re-ranked search results together with other competing runs from the same year according to ERR20 or nDCG20. The characters in brackets indicate a significant difference compared with other methods, marked using uppercase or lowercase characters, namely P/p for PACRR, D/d for DRMM, L/l for DUETL and M/m for MatchPyramid, representing the two-tailed significance at 95% or 90% confidence levels, respectively.

It can be seen that, by simply re-ranking the search results from the query-likelihood method, the model can already achieve the best runs in all three years. Beyond that, compared with the query-likelihood results, the average improvement in terms of ERR@20 is more than 99% and the average improvement in terms of nDCG@20 is 104%.

RerankALL. Given that the search results from *QL* only account for a small subset of all judged documents and different retrieval models return different subsets, we evaluate our re-ranker's performance when re-ranking all submitted runs from the Trec Web Track 2009–14. This evaluation focuses on two aspects: how many different runs we can improve upon and by how much we improve. The former aspect is about the adaptability of a neural IR model, investigating whether it can make improvements based on different kinds of retrieval models, while the latter aspect focuses on the magnitude of improvements. Tables 2 summarizes the percentages of systems that see improvements based on either ERR@20 and nDCG@20 out of the total number of systems in each year. In Table 3, we further report the average percentage of improvements.

Table 2 demonstrates that on average more than 95% of runs are improved by the RE-PACRR in terms of both measures. When compared with other models, RE-PACRR significantly outperforms all baseline models on five years out of six in terms of ERR@20. It is only significantly better than the results from PACRR in 2013 and 14 when measured with nDCG@20, but outperforms the other baselines in all six years. Note that, compared with nDCG@20, ERR@20 emphasizes the quality of the top-ranked documents and heavily penalizes relevant documents that are ranked lower by a model when enough relevant documents have been observed earlier [1]. This means that the improvement of the ERR for a model mainly comes from improvements on queries for which search results at the top are not good enough from an initial ranker, and hardly from the queries where an initial ranker already performs well and a re-ranker makes the search results even better by adjusting the positions of documents in lower positions. This could explain the differences between results in terms of ERR@20 and nDCG@20, indicating that the two measures actually reflect different types of improvements. Furthermore, in Table 3, the average improvements on different runs are summarized, and on average the initial runs get improved by 54% and 67% of ERR@20 and nDCG@20, respectively. Under both measures, RE-PACRR performs significantly better than all baselines in five years out of six years.

PairAccuracy. Ideally, a re-ranking model should make correct decisions when ranking all documents. Therefore, we further rely on a pairwise ranking task to compare different models in this regard. In particular, given a query and a set of documents, different models assign a score to each document according to their inferred relevance relative to the given query. Thereafter, all pairs of documents are examined and the pairs that are ranked in concordance with the ground-truth judgments from Trec are deemed correct, based on which an aggregated accuracy is reported on all such document pairs in different years. For example, given query q and two documents d_1 and d_2, along with their ground-truth judgments $label(d_1)$ and $label(d_2)$, a re-ranking model provides their relevance scores as $rel(q, d_1)$ and $rel(q, d_2)$. The re-ranking model is correct when it predicts these two documents in the same order as in the ranking from the ground-truth label, e.g., $rel(q, d_1) > rel(q, d_2)$ and $label(d_1) > label(d_2)$. The relevance judgments in the Trec Web Track include up to six relevance levels: junk pages (Junk), non-relevant (NRel), relevant (Rel), highly relevant (HRel), key pages (Key), and navigational pages (Nav), corresponding to six graded levels, i.e., -2, 0, 1, 2, 3, 4. As suggested in [9], a navigational document is different from other relevant documents in terms of the user intent it satisfies, where links to other relevant documents for the same query are provided. Thus, documents labeled with Nav are not considered in this task. Moreover, documents labeled as Junk and NRel, and documents labeled as HRel and Key are merged into NRel and HRel, respectively, due to the limited number of

Table 1: Err@20 and nDCG@20 on Trec Web Track 2012–14 when re-ranking search results from *QL*. The comparisons are conducted between RE-PACRR and PACRR (P/p), DRMM (D/d), DUETL (L/l), as well as MatchPyramid (M/m). The upper/lower-case characters in brackets indicate a significant difference under two-tailed paired Student's t-tests at 95% or 90% confidence levels relative to the corresponding approach. In addition, the relative improvements (%) and ranks among all runs within the respective years according to Err@20 and nDCG@20 are also reported after the absolute scores.

Measures	Year	RE-PACRR	PACRR	MatchPyramid	DUETL	DRMM
ERR@20	wt12	0.390 (p↑D↑L↑M↑) 121% 1	0.347 (d↑L↑M↑) 96% 1	0.309 (P↓L↑) 74% 5	0.218 (P↓D↓M↓) 23% 18	0.314 (p↓L↑) 78% 4
	wt13	0.190 (p↑D↑L↑M↑) 89% 1	0.175 (D↑L↑M↑) 74% 3	0.135 (P↓D↓) 34% 15	0.137 (P↓D↓) 36% 14	0.155 (P↓L↑M↑) 54% 7
	wt14	0.246 (P↑D↑L↑M↑) 88% 1	0.223 (D↑L↑M↑) 70% 1	0.183 (P↓) 40% 12	0.174 (P↓D↓) 33% 16	0.195 (P↓L↑) 49% 8
nDCG@20	wt12	0.281 (P↑D↑L↑M↑) 163% 1	0.248 (D↑L↑M↑) 133% 2	0.212 (P↓L↑) 99% 5	0.155 (P↓D↓M↓) 45% 18	0.205 (P↓L↑) 93% 6
	wt13	0.345 (p↑D↑L↑M↑) 82% 1	0.324 (D↑L↑M↑) 70% 2	0.263 (P↓L↑) 39% 6	0.239 (P↓D↓M↓) 26% 15	0.269 (P↓L↑) 41% 6
	wt14	0.385 (P↑D↑L↑M↑) 67% 1	0.352 (D↑L↑M↑) 52% 1	0.297 (P↓l↑) 29% 7	0.275 (P↓D↓m↓) 19% 11	0.302 (P↓L↑) 31% 5

Table 2: The percentage of runs that show improvements in terms of a measure when re-ranking all runs from the Trec Web Track 2009–14 based on Err@20 and nDCG@20. The comparisons are conducted in between RE-PACRR and PACRR (P/p), DRMM (D/d), DUETL (L/l), as well as MatchPyramid (M/m). The upper/lower-case characters in brackets indicate a significant difference under two-tailed paired Student's t-tests at 95% or 90% confidence levels relative to the corresponding approach.

Measures	Year	RE-PACRR	PACRR	MatchPyramid	DUETL	DRMM
ERR@20	wt09	91% (D↑L↑)	92% (D↑L↑m↑)	86% (p↓D↑l↑)	77% (P↓m↓)	72% (P↓M↓)
	wt10	98% (P↑D↑L↑M↑)	95% (D↑L↑)	95% (D↑L↑)	69% (P↓D↓M↓)	91% (P↓L↑M↓)
	wt11	98% (P↑D↑L↑M↑)	69% (D↑L↑M↑)	43% (P↓L↑)	26% (P↓D↓M↓)	49% (P↓L↑)
	wt12	98% (P↑d↑L↑M↑)	92% (L↑)	93% (L↑)	68% (P↓D↓M↓)	95% (L↑)
	wt13	94% (P↑D↑L↑M↑)	85% (L↑M↑)	64% (P↓d↓)	61% (P↓D↓)	83% (L↑m↑)
	wt14	96% (P↑D↑L↑M↑)	84% (L↑M↑)	58% (P↓)	52% (P↓)	68%
nDCG@20	wt09	94% (D↑L↑M↑)	96% (D↑L↑M↑)	82% (P↓D↑L↑)	71% (P↓M↓)	68% (P↓M↓)
	wt10	98% (D↑L↑M↑)	98% (D↑L↑m↑)	95% (p↓L↑)	72% (P↓D↓M↓)	94% (P↓L↑)
	wt11	98% (D↑L↑M↑)	94% (D↑L↑M↑)	49% (P↓L↑)	31% (P↓D↓M↓)	56% (P↓L↑)
	wt12	98% (D↑L↑M↑)	97% (D↑L↑M↑)	90% (P↓L↑)	68% (P↓D↓M↓)	92% (P↓L↑)
	wt13	91% (P↑D↑L↑M↑)	88% (D↑L↑M↑)	80% (P↓L↑)	65% (P↓D↓M↓)	80% (P↓L↑)
	wt14	97% (P↑D↑L↑M↑)	90% (D↑L↑M↑)	69% (P↓l↑)	59% (P↓D↓m↓)	74% (P↓L↑)

documents labeled as Junk and Key. Then, all pairs of documents with different labels are generated as test pairs. In total, these three label pairs account for 95% of all document pairs according to the column named "volume" in Table 4.

The results show that significant improvements relative to all baselines are observed in three out of all six years among the three label pairs. Compared with DUETL, MatchPyramid, and DRMM, RE-PACRR performs better in all six years for labeling pairs between HRel and NRel, as well as between Rel and NRel. As for document pairs that are labeled as HRel and Rel, the RE-PACRR performs relatively close to the baselines. In terms of absolute accuracy, on average, the RE-PACRR yields correct predictions on 79.1%, 73.5%, and 59.1% of document pairs for label pairs HRel–NRel, Rel–NRel, and HRel–Rel, respectively, where the decreasing accuracy is due to the increasing difficulty on these document pairs.

5 RELATED WORK

Ad-hoc retrieval systems aim at ranking documents with respect to their relevance relative to a given query. Recently, the promise of deep learning as a potential driver for new advances in retrieval quality has attracted significant attention. Neural IR approaches can be categorized as semantic matching models and relevance matching models.

The former follows the embedding approach adopted in many natural language processing tasks, focusing on comparing the meaning of a query and a document by converting both into a low-dimensional semantic space. For example, ARC-I and ARC-II [15] are two such models that the authors apply to the the tasks of sentence completion, identifying the response to a microblog post, and performing paraphrase detection. Beyond that, Huang et al. [8] propose Deep Structured Semantic Models (DSSM), which learn low-dimensional representations of queries and documents in a

Table 3: The average differences of the measure score for individual runs when re-ranking all runs from the TREC Web Track 2009–14 based on ERR@20 and nDCG@20. The comparisons are conducted between RE-PACRR and PACRR (P/p), DRMM (D/d), DUETL (L/l), as well as MatchPyramid (M/m). The upper/lower-case characters the brackets indicate a significant difference under two-tailed paired Student's t-tests at 95% or 90% confidence levels relative to the corresponding approach.

Measures	Year	RE-PACRR	PACRR	MatchPyramid	DUETL	DRMM
ERR@20	wt09	43% (D↑L↑M↑)	40% (D↑L↑M↑)	31% (P↓D↑L↑)	22% (P↓M↓)	20% (P↓M↓)
	wt10	98% (P↑D↑L↑M↑)	74% (D↑L↑M↑)	54% (P↓d↑L↑)	23% (P↓M↓)	44% (P↓m↓)
	wt11	33% (P↑D↑L↑M↑)	11% (D↑L↑M↑)	-4% (P↓)	-11% (P↓D↓)	-0% (P↓L↑)
	wt12	89% (P↑D↑L↑)	66% (L↑)	68% (L↑)	22% (P↓D↓M↓)	70% (L↑)
	wt13	36% (P↑D↑L↑M↑)	27% (L↑M↑)	9% (P↓D↓)	8% (P↓D↓)	20% (L↑M↑)
	wt14	29% (P↑D↑L↑M↑)	16% (d↑L↑M↑)	5% (P↓)	2% (P↓)	8% (p↓)
nDCG@20	wt09	36% (D↑L↑M↑)	35% (D↑L↑M↑)	20% (P↓D↑)	14% (P↓)	12% (P↓M↓)
	wt10	125% (P↑D↑L↑M↑)	96% (D↑L↑M↑)	66% (P↓L↑)	31% (P↓M↓)	58% (P↓)
	wt11	43% (P↑D↑L↑M↑)	26% (D↑L↑M↑)	0% (P↓d↓l↑)	-11% (P↓D↓m↓)	7% (P↓L↑m↑)
	wt12	122% (P↑D↑L↑M↑)	89% (D↑L↑m↑)	65% (p↓L↑)	21% (P↓D↓M↓)	69% (P↓L↑)
	wt13	43% (P↑D↑L↑M↑)	31% (D↑L↑M↑)	16% (P↓L↑)	9% (P↓d↓M↓)	15% (P↓l↑)
	wt14	31% (P↑D↑L↑M↑)	21% (D↑L↑M↑)	12% (P↓l↑)	6% (P↓D↓m↓)	11% (P↓L↑)

Table 4: Comparison among tested methods in terms of accuracy in ranking document pairs with different label pairs. The columns "volume" and "# queries" record the occurrences of each label combination out of the total pairs, and the number of queries that include a particular label combination among all six years, respectively. The comparisons are conducted between RE-PACRR and PACRR (P/p), DRMM (D/d), DUETL (L/l), as well as MatchPyramid (M/m). The upper/lower-case characters in brackets indicate a significant difference under two-tailed paired Student's t-tests at 95% or 90% confidence levels relative to the corresponding approaches.

Label Pair	volume (%)	# queries	Year	RE-PACRR	PACRR	MatchPyramid	DUETL	DRMM
HRel-NRel	23.1%	262	wt09	0.715 (P↑D↑L↑M↑)	0.694 (D↑L↑M↑)	0.656 (P↓D↑l↑)	0.632 (P↓D↑m↓)	0.602 (P↓L↓M↓)
			wt10	0.846 (D↑L↑M↑)	0.828 (D↑L↑M↑)	0.777 (P↓D↑L↑)	0.707 (P↓M↓)	0.739 (P↓M↓)
			wt11	0.837 (P↑D↑L↑M↑)	0.789 (D↑L↑M↑)	0.757 (P↓D↑L↑)	0.700 (P↓d↓M↓)	0.735 (P↓l↑M↓)
			wt12	0.826 (P↑D↑L↑M↑)	0.795 (D↑L↑M↑)	0.741 (P↓D↑L↑)	0.655 (P↓d↓M↓)	0.694 (P↓l↑M↓)
			wt13	0.758 (D↑L↑M↑)	0.749 (D↑L↑M↑)	0.691 (P↓D↑L↑)	0.647 (P↓M↓)	0.637 (P↓M↓)
			wt14	0.766 (D↑L↑M↑)	0.772 (D↑L↑M↑)	0.672 (P↓d↑)	0.648 (P↓)	0.649 (P↓m↓)
HRel-Rel	8.4%	257	wt09	0.531	0.539	0.538	0.534	0.539
			wt10	0.587 (p↑D↑L↑)	0.571 (D↑L↑)	0.581 (D↑L↑)	0.529 (P↓M↓)	0.544 (P↓M↓)
			wt11	0.582 (P↑d↓L↑)	0.537 (D↓M↓)	0.608 (P↑L↑)	0.536 (D↓M↓)	0.606 (P↑L↑)
			wt12	0.671 (D↑L↑M↑)	0.655 (D↑L↑M↑)	0.607 (P↓D↑L↑)	0.550 (P↓M↓)	0.549 (P↓M↓)
			wt13	0.572 (D↑L↑)	0.573 (d↑L↑)	0.564 (l↑)	0.545 (P↓m↓)	0.552 (p↓)
			wt14	0.602 (P↑D↑L↑M↑)	0.581 (D↑)	0.575 (D↑)	0.551	0.544 (P↓M↓)
Rel-NRel	63.5%	290	wt09	0.682 (P↑D↑L↑M↑)	0.660 (D↑L↑M↑)	0.619 (P↓D↑)	0.598 (P↓D↑)	0.565 (P↓L↓M↓)
			wt10	0.799 (D↑L↑M↑)	0.788 (D↑L↑M↑)	0.718 (P↓L↑)	0.686 (P↓M↓)	0.708 (P↓)
			wt11	0.782 (D↑L↑M↑)	0.771 (D↑L↑M↑)	0.651 (P↓D↑)	0.650 (P↓D↑)	0.614 (P↓L↓M↓)
			wt12	0.741 (P↑D↑L↑M↑)	0.725 (D↑L↑M↑)	0.667 (P↓L↑)	0.619 (P↓D↓M↓)	0.658 (P↓L↑)
			wt13	0.707 (p↑D↑L↑M↑)	0.692 (D↑L↑M↑)	0.635 (P↓D↑L↑)	0.605 (P↓d↑M↓)	0.586 (P↓l↓M↓)
			wt14	0.700 (P↓D↑L↑M↑)	0.721 (D↑L↑M↑)	0.612 (P↓)	0.597 (P↓)	0.603 (P↓)

shared space. DSSM then performs ranking by comparing the cosine similarities between a given query's representation and the respective representations of documents in the collection.

As for relevance matching models, Guo et al. [5] argued that the type of matching generally used for information retrieval differs from the type of matching generally used in NLP tasks. That is, ranking models are concerned with *relevance matching*, where the focus is on determining whether one input text (i.e., a document) is relevant to the other input text (i.e., a query). Methods that perform semantic matching typically learn input representations independent of each other, whereas methods that perform relevance matching consider the interactions between two heterogeneous inputs. In particular, Guo et al. proposed the Deep Relevance Matching Model (DRMM), which takes a sequence of fixed-length query term similarity histograms as input. Each histogram h_j represents the matches between one query term q_j in a given query q and the terms in a given document. The query similarity histograms are each fed through a series of fully connected layers to produce a similarity signal for each query term. The document's relevance score $rel(q, d)$ is a weighted summation of each query term's similarity signal. Guo et al.'s DRMM model [5] has been reported to outperform all the aforementioned semantic matching models by substantial margins on Trec Web Track data. Other relevance matching models include the early MatchPyramid [7, 14] and more recent PACRR [9]. Given that both aim at the ad-hoc retrieval task, MatchPyramid and PACRR share similar architectures: both start from similarity matrix between a query and a document, thereafter combining max-pooling layers on top of convolutional neural network (CNN) kernels, and use either LSTM or several dense layers to generate a relevance score. However, they are motivated quite differently, and employ different CNN structures, leading to a significant difference in their performance, as indicated in Section 4. As a pioneering work, MatchPyramid [14] investigated the usefulness of a model architecture from related work [7], the impact of different kernel sizes, and the impact of different max-pooling sizes. Meanwhile, PACRR [9] exploits different CNN kernels for different kinds of matching patterns, followed by two separate max-pooling layers for the filters and matching patterns. This setting is motivated by domain insights from ad-hoc retrieval, making it different from MatchPyramid and CNN architectures in computer vision, such as in [16]. Finally, Mitra et al. [11] propose DUET, a deep ranking model that considers both exact matches between document and query terms (the *local model*) and the similarity between low-dimensional representations of the query and document (the *distributed model*). The authors make a distinction between retrieval models that use *local representations*, which are representations in which "each term is represented by a unique identifier", and *distributed representations* that are used to "compare the query and the document in the latent semantic space." The RE-PACRR model proposed in this work, being mainly based on PACRR, adopts a relevance matching approach, and thus is mainly compared with models belonging to this category.

6 CONCLUSION

In this work, a novel neural IR model named RE-PACRR is proposed, which revisits a number of task-specific insights from traditional ad-hoc retrieval, and incorporates building blocks to address them in an integrated neural framework. Extensive experiments on Trec Web Track data confirm that the proposed model performs better than all baseline models on three benchmarks. In our future work, we plan to conduct an ablation study to better understand the contribution of the individual novel components as well as their interplay. Another direction is to exploit further insights from the literature on ad-hoc retrieval and to devise suitable components to make them accessible to a neural model.

REFERENCES

[1] Olivier Chapelle, Donald Metlzer, Ya Zhang, and Pierre Grinspan. 2009. Expected reciprocal rank for graded relevance. In *Proceedings of the 18th ACM conference on Information and knowledge management (CIKM '09)*. ACM, New York, NY, USA, 621–630. DOI: http://dx.doi.org/10.1145/1645953.1646033

[2] Nick Craswell, Onno Zoeter, Michael Taylor, and Bill Ramsey. 2008. An experimental comparison of click position-bias models. In *Proceedings of the 2008 International Conference on Web Search and Data Mining*. ACM, 87–94.

[3] Mostafa Dehghani, Hamed Zamani, Aliaksei Severyn, Jaap Kamps, and W Bruce Croft. 2017. Neural Ranking Models with Weak Supervision. *arXiv preprint arXiv:1704.08803* (2017).

[4] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press. http://www.deeplearningbook.org.

[5] Jiafeng Guo, Yixing Fan, Qingyao Ai, and W Bruce Croft. 2016. A deep relevance matching model for ad-hoc retrieval. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*. ACM, 55–64.

[6] Manish Gupta and Michael Bendersky. 2015. Information Retrieval with Verbose Queries. (2015).

[7] Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. 2014. Convolutional Neural Network Architectures for Matching Natural Language Sentences. In *Advances in Neural Information Processing Systems 27*. 2042–2050.

[8] Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. 2013. Learning Deep Structured Semantic Models for Web Search Using Clickthrough Data. In *Proceedings of the 22Nd ACM International Conference on Information & Knowledge Management (CIKM '13)*. ACM, New York, NY, USA, 2333–2338. DOI: http://dx.doi.org/10.1145/2505515.2505665

[9] Kai Hui, Andrew Yates, Klaus Berberich, and Gerard de Melo. 2017. A Position-Aware Deep Model for Relevance Matching in Information Retrieval. *arXiv preprint arXiv:1704.03940* (2017).

[10] Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems (TOIS)* 20, 4 (2002), 422–446.

[11] Bhaskar Mitra, Fernando Diaz, and Nick Craswell. 2017. Learning to Match Using Local and Distributed Representations of Text for Web Search. In *Proceedings of WWW 2017*. ACM.

[12] Bhaskar Mitra, Eric Nalisnick, Nick Craswell, and Rich Caruana. 2016. A dual embedding space model for document ranking. *arXiv preprint arXiv:1602.01137* (2016).

[13] I. Ounis, G. Amati, V. Plachouras, B. He, C. Macdonald, and C. Lioma. 2006. Terrier: A High Performance and Scalable Information Retrieval Platform. In *Proceedings of ACM SIGIR'06 Workshop on Open Source Information Retrieval (OSIR 2006)*.

[14] Liang Pang, Yanyan Lan, Jiafeng Guo, Jun Xu, and Xueqi Cheng. 2016. A Study of MatchPyramid Models on Ad-hoc Retrieval. *CoRR* abs/1606.04648 (2016). http://arxiv.org/abs/1606.04648

[15] Liang Pang, Yanyan Lan, Jiafeng Guo, Jun Xu, Shengxian Wan, and Xueqi Cheng. 2016. Text Matching As Image Recognition. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI'16)*. 2793–2799.

[16] Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).

[17] Tao Tao and ChengXiang Zhai. 2007. An exploration of proximity measures in information retrieval. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 295–302.

[18] Hamed Zamani and W Bruce Croft. 2017. Relevance-based Word Embedding. *arXiv preprint arXiv:1705.03556* (2017).