Co-PACRR: A Context-Aware Neural IR Model for Ad-hoc Retrieval

Kai Hui
Max Planck Institute for Informatics / SAP SE
kai.hui@sap.com

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

Neural IR models, such as DRMM and PACRR, have achieved strong results by successfully capturing relevance matching signals. We argue that the context of these matching signals is also important. Intuitively, when extracting, modeling, and combining matching signals, one would like to consider the surrounding text (local context) as well as other signals from the same document that can contribute to the overall relevance score. In this work, we highlight three potential shortcomings caused by not considering context information and propose three neural ingredients to address them: a disambiguation component, cascade k-max pooling, and a shuffling combination layer. Incorporating these components into the PACRR model yields Co-PACRR, a novel context-aware neural IR model. Extensive comparisons with established models on Trec Web Track data confirm that the proposed model can achieve superior search results. In addition, an ablation analysis is conducted to gain insights into the impact of and interactions between different components. We release our code to enable future comparisons[1].

ACM Reference Format:
Kai Hui, Andrew Yates, Klaus Berberich, and Gerard de Melo. 2018. Co-PACRR: A Context-Aware Neural IR Model for Ad-hoc Retrieval. In *WSDM 2018: The Eleventh ACM International Conference on Web Search and Data Mining, February 5–9, 2018, Marina Del Rey, CA, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3159652.3159689

1 INTRODUCTION

State-of-the-art neural models for ad-hoc information retrieval aim to model the interactions between a query and a document to produce a relevance score, which are analogous to traditional interaction signals such as BM25 scores. Guo et al. [7] pointed out that a neural IR model should capture query-document interactions in terms of *relevance matching* signals rather than capturing *semantic matching* signals as commonly used in natural language processing (NLP). Relevance matching focuses on the pertinence of local parts of the document with respect to the query (e.g., via n-gram matches), whereas semantic matching captures the overall semantic similarity between the query and the entire document. Accordingly, relevance matching over unigrams has been successfully modeled using histograms in the *DRMM* model [7], using a convolutional layer in *DUET*'s local model [15], and using a pool of kernels in the more recent *K-NRM* model [23]. In addition, position-aware relevance matching signals are further captured in *PACRR* [10] with the goal of encoding matching signals beyond unigrams, such as n-gram matches and "soft" n-gram matches, in which the order of some terms is modified.

Existing models have achieved strong results by focusing on modeling relevance matching signals. However, we argue that the context of such signals are also important but has yet to be fully accounted for in these models. Intuitively, a matching signal contributes to the final relevance score within the context of its local text window and the context of all matching signals from the whole document. Given a matching signal, a text window that embeds the signal is referred to as its *local context*, whereas all matching signals from the same document are referred to as the signal's *global context.* Inspired by past research within the IR community, we first highlight three particular shortcomings that can be addressed by incorporating context. Thereafter, we introduce novel neural components to address the shortcomings within PACRR [10], a state-of-the-art neural IR model. This ultimately leads to Co-PACRR (context-aware PACRR), a novel model as summarized in Figure 1.

To start with, when disregarding the local context, the matching signals extracted between terms from a query and a document may suffer from *ambiguity*. For example, in the query "*Jaguar SUV price*", the term "Jaguar" refers to a car brand, but "Jaguar" also happens to be the name of a species of animal. Such ambiguity can mislead a model to extract false positive matching signals. In the above example, an occurrence of the term "jaguar" referring to the animal should not contribute much to the document's relevance score.

Beyond this, accounting for the global document context may be important as well. Some such signals are desirable, while others need to be disregarded. In particular, we conjecture that the *the location of the matches* is important to better account for the level of reading effort needed to reach the relevant information. For example, consider two pseudo-documents that are both concatenations

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

WSDM 2018, February 5–9, 2018, Marina Del Rey, CA, USA

ACM ISBN 978-1-4503-5581-0/18/02…$15.00
https://doi.org/10.1145/3159652.3159689

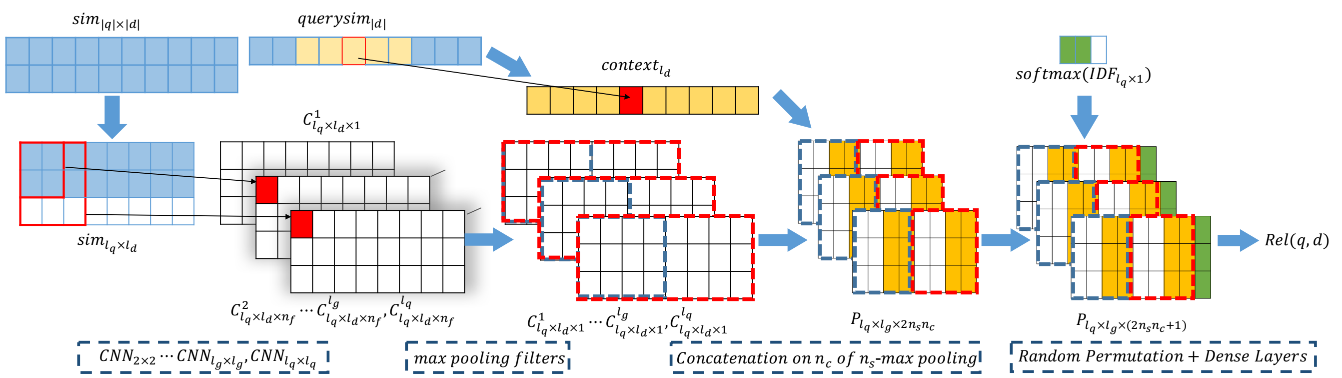

Figure 1: The pipeline of Co-PACRR. The inputs include two matrices, namely, $sim_{|q|\times|d|}$ and $querysim_{|d|}$. All these similarity matrices are truncated/zero-padded to the dimensionalities governed by l_q and l_d. Several 2D convolutional kernels are first applied to the similarity matrices, one for each $l_g \in [2, l_g]$. Next, max pooling is applied to the filters, leading to $l_g + 1$ 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*. At the same time, the context similarity corresponding to each term in *top-n_s* from $context_{l_d}$ is also appended, leading to $P_{l_q\times l_g\times(2n_s n_c)}$. Finally, the query terms' normalized IDFs are appended, and a feed forward network is applied, after permuting the rows in $P_{l_q\times l_g\times(2n_s n_c+1)}$, yielding a query-document relevance score $rel(q,d)$. In this plot, a setting with $l_d = 8$, $l_q = 3$, $l_g = 3$, $n_c = 2$, $n_s = 2$, and $cpos = [50\%, 100\%]$ is shown.

of one relevant and one non-relevant document, but in a different order. Although the same relevant information is present, extra effort is required when reading the pseudo-document where the non-relevant document appears first.

Not all aspects of the document context, however, are beneficial. In particular, we argue that the order in which the document matches different query terms may vary, as there can be many ways to express the same information request. When combining matching signals from different query terms, PACRR employs a recurrent layer, whereas DRMM, K-NRM, the local model in DUET, and MatchPyramid employ feedforward layers. Both kinds of models may be sensitive to the order in which query terms are matched, as the signals from individual query term matches and their associated positions in a query are jointly considered. Learning a query term order-dependent combination is particularly concerning when the query dimension is zero-padded (as in most models), because the aggregation layer may incorrectly learn to down-weight the positions that are zero-padded more often (e.g., at the end of a short query) in the training data. More generally, the aggregation layer may learn to treat matching signals differently depending on the position of a term within the query. This may hurt the model's ability to generalize to different reformulations of a given query, and it is also unnecessary because positional information is already accounted for in an earlier layer.

To close these gaps, we introduce neural components to cater to both the local and the global context. Intuitively, to avoid extracting false positive matching signals due to ambiguity, matching signals are double-checked based on their local context and penalized if there is a mismatch between the senses of words between the document and the query. To consider the global context of matching signals, the signals' strengths at different document positions are considered. To disregard the absolute positions of terms in the query, the sequential dependency over query terms is decoupled before the aggregating combination layer. While these ideas apply more generally, we incorporate them into the PACRR architecture to develop specific neural components, which leads to the Co-PACRR model that contains the following new components:

- A *disambiguation* building block to address the challenge of ambiguity by co-considering salient matching signals together with the local text window in which they occurred.
- A *cascade k-max pooling* approach in place of regular k-max pooling layers, enabling the model to benefit from information about the location of matches. These locations are jointly modeled together with the matching signals. This is inspired by the cascade model [4], which is based on the idea that relevance gains are influenced by the amount of relevant information that has already been observed.
- A *shuffling combination* layer to regularize the model so as to disregard the absolute positions of terms within the query. Removing query-dependent context before combination improves the generalization ability of the model.

Contributions. We incorporate the aforementioned building blocks into the established PACRR model, leading to the novel Co-PACRR model, jointly modeling matching signals with their local and global context. Through a comparison with multiple state-of-the-art models including DRMM, K-NRM, the local model in DUET, MatchPyramid, and the PACRR model on six years of TREC Web Track benchmarks, we demonstrate the superior performance of Co-PACRR. Remarkably, when re-ranking the search results from a naïve initial ranker, namely a query-likelihood ranking model, the re-ranked runs are ranked within the top-3 on at least five years based on

ERR@20. In addition, we also investigate the individual and joint effects of the proposed components to better understand the proposed model in an **ablation analysis**.

Organization. The rest of this paper unfolds as follows. We discuss related work in Section 2 and put our work in context. Section 3 recaps the basic neural-IR model PACRR, and thereafter Section 4 describes the proposed building components in detail. The setup, results, and analyses of our extensive experimental evaluation can be found in Section 5 and Section 6, before concluding in Section 7.

2 RELATED WORK

In ad-hoc retrieval, a system aims at creating a ranking of documents according to their relevance relative to a given query. The recent promises of deep learning methods as potential drivers for further advances in retrieval quality have attracted significant attention. Unlike learning-to-rank methods, where models are learned on top of a list of handcrafted features [13], a neural IR model aims at modeling the interactions between a query and a document directly based on their free text. Actually, the interactions being learned in a neural IR model correspond to one of the feature groups employed in learning-to-rank methods. They involve both a query and a document, as do BM25 scores. The proposed Co-PACRR belongs to this class of neural IR models and is hence compared with other neural IR models in Section 5.

As described in Section 1, neural IR approaches can be categorized as *semantic matching* and *relevance matching* models. The former follows the embedding approach adopted in many natural language processing tasks, aiming at comparing the semantic meaning of two pieces of text by mapping both into a low-dimensional representation space. Therefore, models developed for natural language processing tasks can also be used as retrieval models by assigning a similarity score to individual query-document pairs. For example, ARC-I and ARC-II [19] are two such models developed for the tasks of sentence completion, identifying the response to a microblog post, and performing paraphrase detection. In addition, Huang et al. [9] proposed Deep Structured Semantic Models (DSSM), which learn low-dimensional representations of a query and a document in a semantic space before evaluating the document according to its cosine similarity relative to the query. Similar approaches such as C-DSSM [20] further employed alternative means to learn dense representations of the documents.

In comparison, Guo et al. [7] argued that the matching required in information retrieval is different from the matching used in NLP tasks, and that *relevance matching* is better suited for retrieval tasks. Relevance matching compares two text sequences jointly, namely, a document and a query, by directly modeling their interactions. In relevance matching, local signals such as unigram matches are important. Meanwhile, semantic matching seeks to model the semantic meaning of the two text sequences independently, and the matching is considered in a semantic space. Accordingly, the Deep Relevance Matching Model (DRMM) [7] was proposed to model unigram relevance matching by encoding a query-document pair in terms of a histogram of similarities between terms from the query and the document. More recently, K-NRM [23] relied on a pool of kernels in place of the histogram, capturing the unigram relevance matching in a more smooth manner, addressing the issues of bin boundaries in generating histograms. In addition to the unigram signals, position-aware neural IR models have been proposed, such as MatchPyramid [8, 18], which is motivated by works from computer vision [21], and PACRR [10], which follows the ideas of term dependency [12, 14] and query term proximity [22] modeling from ad-hoc retrieval. Both encode matching signals beyond a single term with convolutional neural networks (CNNs). Beyond that, Mitra et al. [15] proposed DUET, a hybrid deep ranking model combining both kinds of matching, with two independent building blocks, namely, a local model for relevance matching and a distributed model for semantic matching. The proposed Co-PACRR model belongs to the class of relevance matching models, and attempts to further incorporate the context of matching signals.

3 BACKGROUND

In this section, we summarize the PACRR model [10], which we build upon by proposing novel components. When describing PACRR, we follow the notation from [10]. 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.

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

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 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 or zero-padded accordingly;

(2) **CNN kernels and max-pooling layers**: multiple CNN kernels with l_f filters capture the query-document interactions, like n-gram matching, corresponding to 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 on the filter dimension to retain the strongest matching signal for each 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 k-max pooling layers, keeping the top-n_s strongest signals for each query term and CNN kernel pair, leading to

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

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

(4) **combination of the 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 a LSTM layer to generate the ultimate relevance score $rel(d, q)$.

Tweaks. Before moving on, we make two changes in order to ease the development of the proposed model. For simplicity, this revised model is denoted as PACRR in the following sections. First, according to our pilot experiments, the performance of the model does not change when replacing the LSTM layer with a stack of dense layers, which have been demonstrated to be able to simulate an arbitrary function [6]. Such dense layers can easily be trained in parallel, leading to faster training [6], whereas back-propagation through an LSTM layer is much more expensive due to its sequential nature. From Section 5, it can be seen that efficiency is important for this study due to the number 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 to a cross-entropy loss as in Eq. 2, following [5], where it has been demonstrated that a cross-entropy loss may lead to better results.

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

4 METHOD

In this section, we describe the novel components in the Co-PACRR model as summarized in Figure 1.

Disambiguation: checking local context when extracting matching signals. Beyond the query-document similarity matrix $sim_{l_q \times l_d}$ used by PACRR, we introduce an input vector denoted as $querysim_{|d|}$ that encodes the similarity between document context vectors and a query vector. Document context vectors represent the meaning of text windows over the document, while the query vector represents the query's meaning. In particular, the vector of a query *queryvec* is computed by averaging the word vectors of all query terms. Similarly, given a position i in a document, its context vector of length, governed by w_c, is computed by averaging the embeddings of all the terms appeared in its surrounding context,

$$context2vec(i) = \frac{\sum_{j \in [i - w_c, i + w_c]} word2vec(\mathbf{d}[i])}{2 * w_c + 1} .$$

Thereafter, the match between the query and a document context at position i is computed by taking the cosine similarity between the query vector and context vector, that is,

$$querysim(i) = cosine(context2vec(i), queryvec) .$$

We employ pre-trained word2vec[2] embeddings due to their widespread availability. In the future, one may desire to replace this with specialized embeddings such as dual embeddings [16] or relevance-based embeddings [24].

Intuitively, to address the challenge of false positive matches stemming from ambiguity, the extracted matching signals on position i are adjusted in the model according to the corresponding similarity between its context and the query. In particular, when combining the top-n_s signals from individual query terms, the corresponding similarities for these top-n_s signals 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 layers, namely, a feed-forward network, to take any ambiguity into account when determining the ultimate score. For example, in the "jaguar" example from Section 1, if the context of "jaguar" consists of terms like "big cat" and "habitat", the context will have a low similarity with a query context containing terms such as "SUV" and "price", informing the model that such occurrences of "jaguar" actually refer to a different concept than the one in the query.

Cascade k-max pooling: encode the location of the relevance information. As discussed in Section 1, to put individual relevance signals into the context of the whole document, both the strength and the positions of match signals matter. We propose to encode such global context by conducting k-max pooling at multiple positions in a document, instead of pooling only on the entire document. For example, one could conduct multiple k-max pooling operations 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 the gained useful information after reading the first, second, third, and fourth quarters 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. We set the length of this array using a hyper-parameter n_c and perform pooling at equal intervals. For example, $n_c = 4$ in the previous example, and $n_c = 2$ results in $cpos = [50\%, 100\%]$.

Shuffling combination: regularizing the query-dependent information. As mentioned in Section 1, the combination of relevance signals among different query terms is supposed to be query-independent to avoid learning a dependency on query term positions. In light of this, we propose to randomly shuffle rows in $P_{l_q \times (l_g n_s)}$ before aggregating them. Note that each row contains signals for multiple n-gram lengths; shuffling the rows does not prevent the model from recognizing n-grams. We argue that, taking advantage of this independence, the shuffling regularizes the query-dependent information and effectively improves the generalization ability of the model by 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 normally in the tail of $sim_{l_q \times l_d}$ [11]. Without shuffling, a model might remember that the relevance signals at the tail of a query (i.e., 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.

5 EVALUATION

In this section, we empirically compare the proposed Co-PACRR with multiple state-of-the-art neural IR models using manual relevance judgments from six years of the TREC Web Track. Following [10], the comparison is based on three benchmarks, namely, re-ranking search results from a simple initial ranker, denoted as RERANKSIMPLE, re-ranking all runs from the TREC Web Track, denoted as RERANKALL, and further examining the classification accuracy in determining the order of document pairs, denoted as

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

PairAccuracy. We compare our model with multiple state-of-the-art neural IR models including the PACRR model [10], MatchPyramid [18], DRMM [7], the local model of DUET (DUETL) [15], and the most recent K-NRM [23] model. As discussed in Section 2, our focus is on evaluating deep relevance matching models, and hence the comparisons are limited to 1) modeling the interactions between a query and a document, excluding the learning-to-rank features for a single document or a query, e.g., PageRank scores, and 2) modeling relevance matching rather than semantic matching [7].

5.1 Experimental Setup

We rely on the 2009–2014 Trec Web Track ad-hoc task benchmarks[3]. In total, there are 300 queries and around 100k judgments (qrels). Six years (2009–14) of query-likelihood baselines (*QL*) provided by the Lemur project's online Indri services[4] [5] serve as the initial ranker in RerankSimple. In addition, the search results from runs submitted by participants from each year are employed in the RerankALL, where there are 71 (2009), 55 (2010), 62 (2011), 48 (2012), 50 (2013), and 27 (2014) runs. ERR@20 [2] is employed as evaluation measure, following the configuration in the Trec Web Track [3], which is computed with the script from Trec[6]. Note that ERR 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 [2]. 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.

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 two remaining years for training and validation. Model parameters and the number of training iterations are chosen by maximizing the ERR@20 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 results 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. This was motivated by an observation that the closeness of the subsets for training and for validation can adversely influence the model selection. We argue that this approach minimizes the effects of the choice of training and validation data. Upper/lowercase characters are employed to indicate the significant difference under two-tailed Student's t-tests at 95% or 90% confidence levels relative to the corresponding approach, denoted as P/p for PACRR, M/m for MatchPyramid, D/d for DRMM, L/l for DUETL and K/k for K-NRM.

Variants of Co-PACRR. With the proposed components, namely, the cascade k-max pooling (C), the disambiguation component (D), and the shuffling combination (S), there are seven model variants in total by including or excluding one of the three building blocks. They are denoted as X(XX)-PACRR, where the X represents the building blocks that are turned on. For example, with cascade k-max pooling and shuffling combination turned on, the model is denoted as CS-PACRR. Meanwhile, with all three components, namely CDS-PACRR, the model is simply referred to as Co-PACRR. In evaluations based on the RerankSimple and RerankALL benchmarks, only the results for Co-PACRR are reported. Meanwhile, the results for the other six variants are reported in Section 6.1 on the PairAccuracy benchmark for an ablation test.

Choice of hyper-parameters. In this work, we focus on evaluating the effects of the proposed building blocks and their interactions, without exhaustively fine-tuning hyper-parameters due to limited computing resources. For the disambiguation building block, we fix the size of the context window as $w_c = 4$ on both sides, leading to a context vector computed over 9 terms, namely, 4+4+1. For the cascade component, we conduct k-max pooling with $cpos = [25\%, 50\%, 75\%, 100\%]$, namely, $n_c = 4$. For the combination phase, we use two fully connected layers of size 16. Apart from the two modifications mentioned in Section 3, we further fix the model choices for PACRR following the original configurations [10]. In particular, the PACRR-firstk variant is employed, fixing the unified similarity matrix 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 is $n_f = 32$. Beyond that, we fix the batch size to 16 and we train individual models to at most 150 iterations. Note that most of the aforementioned hyper-parameters can be tuned given sufficient time and hardware, and the chosen parameters follow those in Hui et al. [10] or are based on preliminary experiments for a better focus on the proposed models. In Section 6.2 we consider the impact of the disambiguation parameter w_c and the cascade parameter n_c.

Due to the availability of training data, K-NRM is trained with a frozen word embedding layer, and with an extra fully connected middle layer including 30 neurons to partially compensate for lost strength due to the frozen word embeddings. This is slightly different from the model architecture described in Xiong et al. [23]. This setting also serves for the purpose of allowing fair model comparisons, given that all the compared models could be co-trained with the word embeddings, resulting in a better model capacity at the costs of prolonged training times and a need for much more training data [10]. Note that with the frozen embedding layer, the evaluation can focus on the model strength that comes from different model architectures, demonstrating the capacity of relatively small models in performing ad-hoc retrieval. All the models are trained with a cross-entropy loss as summarized in Eq. 2, given that different loss functions can also influence the results.

[3]http://trec.nist.gov/tracks.html
[4]http://boston.lti.cs.cmu.edu/Services/clueweb09_batch/
[5]http://boston.lti.cs.cmu.edu/Services/clueweb12_batch/
[6]http://trec.nist.gov/data/web/12/gdeval.pl

Table 1: ERR@20 on Trec Web Track 2009–14 when re-ranking search results from *QL*. The relative improvements (%) relative to QL and ranks among all runs within the respective years according to ERR@20 are also reported.

Year	Co-PACRR			PACRR			MatchPyramid			DRMM			DUETL			K-NRM		
wt09	0.096 (D↑)	6%	47	0.102 (D↑)	13%	41	0.103	14%	38	0.086 (P↓)	-5%	50	0.092	1%	45	0.091	1%	48
wt10	0.160 (P↑K↑D↑L↑M↑)	136%	3	0.146 (K↑D↑L↑m↑)	116%	4	0.131 (p↓L↑)	93%	9	0.131 (P↓L↑)	92%	9	0.103 (P↓K↓D↓M↓)	52%	25	0.128 (P↓L↑)	88%	10
wt11	0.167 (P↑K↑D↑L↑M↑)	52%	2	0.139 (k↑L↑M↑)	26%	15	0.114 (P↓K↓D↓)	3%	31	0.133 (L↑M↑)	21%	19	0.112 (P↓K↓D↓)	1%	35	0.129 (p↓L↑M↑)	17%	23
wt12	0.359 (K↑D↑L↑M↑)	99%	1	0.363 (K↑D↑L↑M↑)	101%	1	0.244 (P↓D↓)	35%	15	0.320 (P↓K↑L↑M↑)	77%	3	0.206 (P↓K↓D↓)	13%	22	0.269 (P↓D↓L↑)	49%	11
wt13	0.189 (K↑D↑L↑M↑)	82%	1	0.184 (K↑D↑L↑M↑)	77%	1	0.131 (P↓D↓)	26%	18	0.166 (P↓K↑L↑M↑)	60%	3	0.130 (P↓D↓)	25%	20	0.141 (P↓D↓)	36%	12
wt14	0.232 (P↑K↑D↑L↑M↑)	84%	1	0.210 (K↑d↑L↑M↑)	67%	4	0.163 (P↓D↓)	29%	19	0.191 (p↓K↑L↑M↑)	52%	10	0.159 (P↓D↓)	26%	20	0.167 (P↓D↓)	32%	17

5.2 Results for Co-PACRR

RerankSimple. We first examine how well the proposed model performs when re-ranking search results from a simple initial ranker, namely, the query-likelihood (QL) model, to put our results in context as in Guo et al. [7]. 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 like Terrier [17]. The results are summarized in Table 1. The ERR@20 of the re-ranked runs is reported, together with the improvements relative to the original QL. 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 ERR@20.

It can be seen that, by simply re-ranking the search results from the query-likelihood method, Co-PACRR can already achieve the top-3 best results in 2010–14. Whereas for 2009, very limited improvements are observed. Combined with Table 3, though variants of Co-PACRR can improve different runs in Trec around 90%, the relative improvements w.r.t. QL are less than 10%, which is worse than the improvements from PACRR and MatchPyramid on 2009. This illustrates that the re-ranking model cannot work independently, as its performance highly depends on the initial ranker. Actually, in Table 1 all compared models experience difficulties in improving QL on 2009, where DRMM even receives a worse ranking. This might be partially explained by the difference of the initial ranker in terms of the recall rate. Intuitively, there should be enough relevant documents to be re-ranked in the initial ranking, otherwise the re-ranker is unable to achieve anything, no matter its quality. The recall rates of QL in different years are as follows (in parentheses): 2009 (0.35), 2010 (0.37), 2011 (0.67), 2012 (0.46), 2013 (0.61), and 2014 (0.68), where 2009 witnesses the lowest recall. However, there may also be other causes for these results.

RerankALL. Given that the search results from QL only account for a small subset of all judged documents, and, more importantly, that the performance of a re-ranker also depends on the initial runs, we evaluate our re-ranker's performance by 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*. Table 2 summarizes the percentages of systems that see improvements based on ERR@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 at least 90% of runs, and on average more than 96% of runs, can be improved by Co-PACRR, which implies a good adaptability, namely, the proposed Co-PACRR can work together with a wide range of initial rankers using different methods. Compared with other neural IR models, in terms of the absolute numbers, Co-PACRR improves the highest number of systems in all the years; when conducting significance tests, in three out of six years, the proposed Co-PACRR significantly outperforms all the baselines. Noticeably, Co-PACRR uniformly achieves good results on all six years, whereas all other methods fail to improve more than 75% of systems in at least one year. There are similar observations for the average improvements shown in Table 3, where Co-PACRR performs best in terms of the average improvements for all six years; on four out of six years Co-PACRR leads other methods with a significant difference. This table shows that Co-PACRR can improve different runs in each year by at least 34% on average.

PairAccuracy. Ideally, a re-ranking model should make correct decisions when ranking all kinds of documents. Therefore, we further rely on a pairwise ranking task to compare different models in this regard. Compared with the other two benchmarks, we argue that PairAccuracy can lead to more comprehensive and more robust comparisons, as a result of its inclusion of all the labeled ground-truth data and its removal of the effects of initial rankers.

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 to be ranked 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). Note that the label Nav actually indicates that a document can satisfy a navigational intent rather than assigning a degree of relevance as Rel and HRel, which makes it difficult to compare navigational documents with other kinds of

Table 2: The percentage of runs that show improvements in terms of ERR@20 when re-ranking all runs from the TREC Web Track 2009–14.

Year	Co-PACRR	PACRR	MatchPyramid	DRMM	DUETL	K-NRM
wt09	90% (D↑L↑)	93% (D↑L↑)	88% (D↑l↑)	70% (P↓K↓M↓)	74% (P↓K↓m↓)	89% (D↑L↑)
wt10	98% (K↑D↑L↑M↑)	96% (D↑L↑M↑)	89% (P↓K↓L↑)	91% (P↓K↓L↑)	74% (P↓K↓D↓M↓)	95% (D↑L↑M↑)
wt11	98% (P↑K↑D↑L↑M↑)	71% (D↑L↑M↑)	15% (P↓K↓D↓L↓)	42% (P↓K↓L↑M↑)	21% (P↓K↓D↓M↑)	69% (D↑L↑M↑)
wt12	98% (P↑K↑d↑L↑M↑)	95% (K↑L↑M↑)	73% (P↓k↓D↓)	94% (K↑L↑M↑)	72% (P↓k↓D↓)	83% (P↓D↓l↑m↑)
wt13	93% (p↑K↑d↑L↑M↑)	86% (K↑L↑M↑)	56% (P↓D↓)	87% (K↑L↑M↑)	43% (P↓K↓D↓)	63% (P↓D↓L↑)
wt14	96% (K↑D↑L↑M↑)	84% (K↑L↑M↑)	61% (P↓K↑L↑)	69% (K↑L↑)	39% (P↓D↓M↓)	43% (P↓D↓M↓)

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.

Year	Co-PACRR	PACRR	MatchPyramid	DRMM	DUETL	K-NRM
wt09	47% (p↑K↑D↑L↑M↑)	42% (K↑D↑L↑M↑)	29% (P↓D↑L↑)	17% (P↓K↓M↓)	16% (P↓K↓M↓)	35% (P↓D↑L↑)
wt10	93% (P↑K↑D↑L↑M↑)	76% (D↑L↑M↑)	51% (P↓K↓L↑)	48% (P↓K↓L↑)	27% (P↓K↓D↓M↓)	68% (D↑L↑M↑)
wt11	39% (P↑K↑D↑L↑M↑)	10% (D↑L↑M↑)	-22% (P↓K↓D↓l↓)	-3% (P↓K↓L↑M↑)	-17% (P↓K↓D↓m↑)	8% (D↑L↑M↑)
wt12	84% (K↑D↑L↑M↑)	74% (K↑L↑M↑)	28% (P↓D↓)	69% (K↑L↑M↑)	29% (P↓D↓)	44% (P↓D↓)
wt13	38% (K↑D↑L↑M↑)	30% (K↑L↑M↑)	4% (P↓D↓)	22% (K↑L↑M↑)	-4% (P↓K↓D↓)	11% (P↓D↓L↑)
wt14	34% (P↑K↑D↑L↑M↑)	20% (K↑d↑L↑M↑)	6% (P↓K↑L↑)	10% (p↓K↑L↑)	-4% (P↓D↓M↓)	-4% (P↓D↓M↓)

relevant documents, e.g., a navigational document versus a document labeled as HRel. 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 their limited number. After aggregating the labels as described, all pairs of documents with different labels are generated as test pairs. From the "volume" and "# queries" columns in Table 4, we can see that different label pairs actually account for quite different volumes in the ground truth, making their respective degrees of influence different. On the other hand, different label pairs actually also represent different difficulties in making a correct prediction, as the closeness of two documents in terms of their relevance determines the difficulty of the predictions. Intuitively, it is easier to distinguish between HRel and NRel documents than to compare a HRel document with a Rel document. Actually, human assessors tend to also disagree more when dealing with document pairs that are very close with each other in terms of their relevance [1]. It can also be seen that these three label pairs being considered account for 95% of all document pairs from Table 4.

From the upper part of Table 4, for the label pair HRel-NRel, Co-PACRR achieves the highest accuracy in terms of the absolute number, and significantly outperforms all baselines on three years. We have similar observations for Rel-NRel, where, however, Co-PACRR performs worse than PACRR in 2014. As for the label pair HRel-Rel, however, Co-PACRR performs very close to the other models, and on 2011, it performs worse than DUETL. Therefore, we can conclude that Co-PACRR outperforms the other baseline results when comparing documents that are far away in terms of relevance, while performing similarly in dealing with harder pairs. In terms of the absolute accuracy, on average, Co-PACRR yields correct predictions on 78.7%, 73.6%, and 58.7% of document pairs for the label pairs HRel–NRel, Rel–NRel, and HRel–Rel, respectively, where the decreasing accuracy confirms the different difficulties in making predictions for different kinds of pairs.

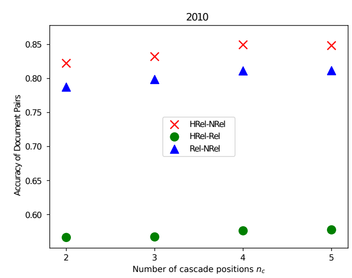

Figure 2: The accuracy on document pairs when using different number of cascade positions n_c for the cascade k-max pooling layer.

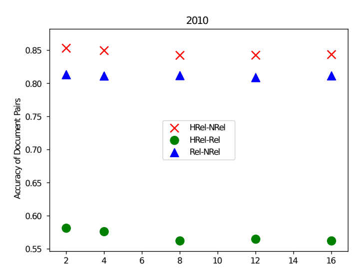

Figure 3: The accuracy on document pairs when varying the size of the context window w_c for the disambiguation component.

Table 4: Comparisons 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.

Label Pair	volume (%)	# queries	Year	Co-PACRR	PACRR	MatchPyramid	DRMM	DUETL	K-NRM
HRel-NRel	23.1%	262	wt09	0.720 (P↑K↑D↑L↑M↑)	0.695 (D↑L↑M↑)	0.654 (P↓K↓D↑L↑)	0.597 (P↓K↓M↓)	0.593 (P↓K↓M↓)	0.689 (D↑L↑M↑)
			wt10	0.850 (P↑K↑D↑L↑M↑)	0.831 (k↑D↑L↑M↑)	0.768 (P↓D↑L↑)	0.740 (P↓K↓L↑M↓)	0.677 (P↓K↓D↓M↓)	0.797 (p↓D↑L↑)
			wt11	0.829 (P↑K↑D↑L↑M↑)	0.778 (K↑D↑L↑M↑)	0.693 (P↓K↓D↓L↑)	0.728 (P↓k↓L↑M↑)	0.638 (P↓K↓D↓M↓)	0.749 (P↓d↑L↑M↑)
			wt12	0.801 (K↑D↑L↑M↑)	0.790 (K↑D↑L↑M↑)	0.703 (P↓)	0.672 (P↓K↓)	0.683 (P↓)	0.728 (P↓D↑)
			wt13	0.752 (K↑D↑L↑M↑)	0.744 (K↑D↑L↑M↑)	0.654 (P↓L↑)	0.648 (P↓l↑)	0.636 (P↓K↓d↓M↓)	0.663 (P↓L↑)
			wt14	0.772 (K↑D↑L↑M↑)	0.770 (K↑D↑L↑M↑)	0.670 (P↓K↑D↑L↑)	0.653 (P↓M↓)	0.639 (P↓M↓)	0.640 (P↓M↓)
HRel-Rel	8.4%	257	wt09	0.545 (L↑)	0.534	0.537	0.543 (L↑)	0.529 (K↓D↓)	0.542 (L↑)
			wt10	0.576 (D↑L↑)	0.577 (D↑L↑)	0.591 (D↑L↑)	0.542 (P↓M↓)	0.545 (P↓M↓)	0.572
			wt11	0.576 (P↑K↑D↓L↑m↓)	0.522 (D↓M↓)	0.589 (P↑K↑D↓L↑)	0.615 (P↑K↑L↑M↑)	0.507 (D↓M↓)	0.518 (D↓M↓)
			wt12	0.645 (K↑D↑L↑M↑)	0.644 (K↑D↑L↑M↑)	0.575 (P↓D↑)	0.528 (P↓K↓L↓M↓)	0.583 (P↓D↑)	0.583 (P↓D↑)
			wt13	0.575 (K↑D↑L↑M↑)	0.579 (m↑)	0.551 (p↓)	0.560	0.558	0.551
			wt14	0.602 (P↑K↑D↑L↑M↑)	0.575 (K↑d↑)	0.569 (K↑D↑)	0.558 (p↓K↑M↓)	0.560 (K↑)	0.507 (P↓D↓L↓M↓)
Rel-NRel	63.5%	290	wt09	0.676 (P↑K↑D↑L↑M↑)	0.663 (K↑D↑L↑M↑)	0.619 (P↓K↓D↑L↑)	0.555 (P↓K↓M↓)	0.563 (P↓K↓M↓)	0.650 (P↓D↑L↑M↑)
			wt10	0.811 (P↑K↑D↑L↑M↑)	0.791 (K↑D↑L↑M↑)	0.708 (P↓K↓L↑)	0.710 (P↓K↓L↑)	0.639 (P↓K↓D↓M↓)	0.751 (P↓D↑L↑M↑)
			wt11	0.787 (P↑K↑D↑L↑M↑)	0.770 (K↑D↑L↑M↑)	0.616 (P↓K↓)	0.607 (P↓K↓)	0.621 (P↓K↓)	0.711 (P↓D↑L↑M↑)
			wt12	0.735 (p↑K↑D↑L↑M↑)	0.721 (K↑D↑L↑M↑)	0.640 (P↓K↓L↑)	0.651 (P↓k↓L↑)	0.616 (P↓K↓D↓M↓)	0.673 (P↓d↑L↑M↑)
			wt13	0.700 (K↑D↑L↑M↑)	0.689 (K↑D↑L↑M↑)	0.612 (P↓D↑L↑)	0.589 (P↓K↓M↓)	0.579 (P↓K↓M↓)	0.623 (P↓D↑L↑)
			wt14	0.708 (p↓K↑D↑L↑M↑)	0.717 (K↑D↑L↑M↑)	0.620 (P↓K↓D↑L↑)	0.597 (P↓K↓M↓)	0.586 (P↓K↓M↓)	0.647 (P↓D↑L↑M↑)

Label Pair	volume (%)	# queries	Year	C-PACRR	D-PACRR	S-PACRR	CD-PACRR	CS-PACRR	DS-PACRR
HRel-NRel	23.1%	262	wt09	0.702 (K↑D↑L↑M↑)	0.701 (K↑D↑L↑M↑)	0.704 (p↑K↑D↑L↑M↑)	0.704 (K↑D↑L↑M↑)	0.716 (P↑K↑D↑L↑M↑)	0.713 (P↑K↑D↑L↑M↑)
			wt10	0.839 (p↑K↑D↑L↑M↑)	0.842 (P↑K↑D↑L↑M↑)	0.843 (P↑K↑D↑L↑M↑)	0.842 (p↑K↑D↑L↑M↑)	0.848 (P↑K↑D↑L↑M↑)	0.843 (p↑K↑D↑L↑M↑)
			wt11	0.808 (K↑D↑L↑M↑)	0.820 (P↑K↑D↑L↑M↑)	0.824 (P↑K↑D↑L↑M↑)	0.810 (p↑K↑D↑L↑M↑)	0.821 (P↑K↑D↑L↑M↑)	0.836 (P↑K↑D↑L↑M↑)
			wt12	0.812 (P↑K↑D↑L↑M↑)	0.785 (K↑D↑L↑M↑)	0.794 (K↑D↑L↑M↑)	0.786 (K↑D↑L↑M↑)	0.819 (P↑K↑D↑L↑M↑)	0.786 (K↑D↑L↑M↑)
			wt13	0.749 (K↑D↑L↑M↑)	0.745 (K↑D↑L↑M↑)	0.755 (K↑D↑L↑M↑)	0.736 (K↑D↑L↑M↑)	0.766 (P↑K↑D↑L↑M↑)	0.751 (K↑D↑L↑M↑)
			wt14	0.773 (K↑D↑L↑M↑)	0.767 (K↑D↑L↑M↑)	0.766 (K↑D↑L↑M↑)	0.768 (K↑D↑L↑M↑)	0.785 (P↑K↑D↑L↑M↑)	0.777 (K↑D↑L↑M↑)
HRel-Rel	8.4%	257	wt09	0.535 (D↓l↑)	0.539 (L↑)	0.543 (L↑)	0.541 (L↑)	0.539 (L↑)	0.546 (p↑L↑)
			wt10	0.585 (D↑L↑)	0.575 (D↑L↑)	0.580 (D↑L↑)	0.568 (p↓d↑L↑M↓)	0.581 (D↑L↑)	0.571 (d↑L↑)
			wt11	0.533 (D↓M↓)	0.565 (P↑K↑D↓L↑M↓)	0.570 (P↑K↑D↓L↑M↓)	0.548 (k↑D↓L↑M↓)	0.552 (P↑K↑D↓L↑M↓)	0.584 (P↑K↑D↓L↑)
			wt12	0.680 (P↑K↑D↑L↑M↑)	0.637 (K↑D↑L↑M↑)	0.658 (P↑K↑D↑L↑M↑)	0.651 (K↑D↑L↑M↑)	0.675 (P↑K↑D↑L↑M↑)	0.641 (K↑D↑L↑M↑)
			wt13	0.553	0.581 (K↑d↑L↑M↑)	0.580 (K↑D↑L↑M↑)	0.557	0.572 (k↑M↑)	0.587 (K↑D↑L↑M↑)
			wt14	0.598 (p↑K↑D↑L↑M↑)	0.589 (K↑D↑L↑M↑)	0.570 (K↑)	0.599 (K↑D↑L↑M↑)	0.596 (P↑K↑D↑L↑M↑)	0.584 (K↑D↑L↑m↑)
Rel-NRel	63.5%	290	wt09	0.672 (p↑K↑D↑L↑M↑)	0.665 (K↑D↑L↑M↑)	0.667 (K↑D↑L↑M↑)	0.670 (p↑K↑D↑L↑M↑)	0.679 (P↑K↑D↑L↑M↑)	0.671 (P↑K↑D↑L↑M↑)
			wt10	0.795 (K↑D↑L↑M↑)	0.802 (K↑D↑L↑M↑)	0.806 (P↑K↑D↑L↑M↑)	0.805 (p↑K↑D↑L↑M↑)	0.807 (P↑K↑D↑L↑M↑)	0.807 (P↑K↑D↑L↑M↑)
			wt11	0.778 (K↑D↑L↑M↑)	0.779 (K↑D↑L↑M↑)	0.788 (P↑K↑D↑L↑M↑)	0.775 (K↑D↑L↑M↑)	0.792 (P↑K↑D↑L↑M↑)	0.787 (P↑K↑D↑L↑M↑)
			wt12	0.728 (K↑D↑L↑M↑)	0.730 (K↑D↑L↑M↑)	0.724 (K↑D↑L↑M↑)	0.726 (K↑D↑L↑M↑)	0.737 (P↑K↑D↑L↑M↑)	0.729 (K↑D↑L↑M↑)
			wt13	0.705 (P↑K↑D↑L↑M↑)	0.685 (K↑D↑L↑M↑)	0.696 (P↑K↑D↑L↑M↑)	0.693 (K↑D↑L↑M↑)	0.714 (P↑K↑D↑L↑M↑)	0.697 (P↑K↑D↑L↑M↑)
			wt14	0.707 (P↓K↑D↑L↑M↑)	0.705 (P↓K↑D↑L↑M↑)	0.717 (K↑D↑L↑M↑)	0.698 (P↓K↑D↑L↑M↑)	0.715 (K↑D↑L↑M↑)	0.717 (K↑D↑L↑M↑)

6 DISCUSSION

6.1 Ablation Analysis

In this section, we attempt to gain further insights about the usefulness of the proposed model components, namely, the cascade k-max pooling (C), the disambiguation (D) and the shuffling combination (S) layer, by drawing comparisons among different model variants. As mentioned, the PairAccuracy benchmark is the most comprehensive due to its inclusion of all document pairs and its removal of the effects of an initial ranker, making the analysis based solely on the proposed neural models. Therefore, our analysis in this section mainly considers PairAccuracy.

Effects of the individual building blocks. We first incorporate the proposed components into PACRR one at time, leading to the C-PACRR, D-PACRR, and S-PACRR model variants, which we use to examine the effects of these building blocks separately. Table 4 demonstrates that the shuffling combination (S-PACRR) alone can boost the performance on three different label pairs, significantly outperforming PACRR on two to three years out of six years for all three label combinations, and performing at least as well as PACRR on the remaining years. As mentioned in Section 1, the shuffling combination performs regularization by preventing the model from learning query-dependent patterns. On the other hand, adding the C-PACRR or D-PACRR component to PACRR actually hurts the performance on 2014 over the Rel-NRel label pair, and only occasionally improves PACRR on other years. Intuitively, both building blocks introduce extra weights into PACRR, increasing the number of nodes for combination by adding the context vectors or by using multiple pooling layers, making the model more prone to overfitting. Such changes might be an issue when only limited training data is available.

Joint effects of different components. To resolve the extra complexity introduced by the cascade pooling layers and the disambiguation building blocks, we further combine these two with the shuffling component, leading to CS-PACRR and DS-PACRR. Meanwhile, we also investigate the joint effects between them by examining CD-PACRR. From Table 4, compared with the PACRR model, both CS-PACRR and DS-PACRR achieve better results not only relative to C-PACRR and D-PACRR, but also to S-PACRR. This is especially true for CS-PACRR, which significantly outperforms

PACRR on all years for HRel-NRel pairs, and on five years for Rel-NRel pairs. This demonstrates that both the cascade pooling and the disambiguation components can help only after introducing extra regularization to offset the extra complexity being introduced. As for CD-PACRR, not surprisingly, it performs on a par with C-PACRR and D-PACRR, and worse than the CS-PACRR and DS-PACRR. Finally, we put all components together and end up with the Co-PACRR model discussed in Section 5, which performs better than C-PACRR and D-PACRR, and similar to S-PACRR, but occasionally worse than CS-PACRR on 2012–14. We argue that this is due to the joint usage of the cascade k-max pooling and the disambiguation, making the model much more complex and thereby expensive to train like CD-PACRR, therefore requiring more training data to work well. We note that DS-PACRR performs better than the S-PACRR variant, supporting our argument that the full model's decreased performance is caused by the added complexity, and not by adding the disambiguation component itself, and this also applies to the cascade k-max pooling layer. In short, we conclude that all three components can lead to improved results. Moreover, we suggest that, when limited training data is available, either CS-PACRR or DS-PACRR could be employed in place of Co-PACRR, since they are less data-hungry compared with Co-PACRR.

6.2 Tuning of Hyper-parameters

Finally, we further investigate the effects of the two hyper-parameters introduced by our proposed components, namely, the number of cascade positions n_c and the size of the context window w_c, which govern the cascade k-max pooling component and the disambiguation component, respectively. Figures 2 and 3 show the effects of applying different n_c and w_c on 2010, where the x-axis represents the configurations of the hyper-parameter, and the y-axis represents the corresponding accuracy on document pairs. In the case of cascade k-max pooling, we uniformly divide [0%, 100%] into n_c parts, e.g., with $n_c = 5$ we have $cpos = [20\%, 40\%, 60\%, 80\%, and 100\%]$. Owing to space constraints, we omit the plots for other years. From Figures 2 and 3, we observe that the model is robust against different choices of n_c and w_c within the investigated ranges, and the trend of the accuracy relative to different choices of hyper-parameters is consistent among the three kinds of label pairs. Furthermore, increasing the number of cascade positions slightly increases the accuracy, whereas increasing the context window size past $w_c = 4$ reduces the accuracy.

7 CONCLUSION

In this work we proposed the novel Co-PACRR neural IR model that incorporates the local and global context of matching signals into the PACRR model through the use of a disambiguation building block, a cascade k-max pooling layer, and a shuffling combination layer. Extensive experiments on TREC Web Track data demonstrated the superior performance of the proposed Co-PACRR model. Notably, the model is trained using TREC data consisting of about 100k training instances, illustrating that models performing ad-hoc retrieval can greatly benefit from architectural improvements as well as an increase in training data. As for future work, one potential direction is the combination of handcrafted learning-to-rank features with the interactions learned by Co-PACRR, where an effective way to learn such features (e.g., PageRank scores) inside the neural model appears non-trivial.

REFERENCES

[1] Omar Alonso and Stefano Mizzaro. 2012. Using crowdsourcing for TREC relevance assessment. *Information Processing & Management* 48, 6 (2012), 1053–1066.
[2] 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. https://doi.org/10.1145/1645953.1646033
[3] Kevyn Collins-Thompson, Craig Macdonald, Paul Bennett, Fernando Diaz, and Ellen M Voorhees. 2015. *TREC 2014 web track overview.* Technical Report. DTIC Document.
[4] 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.
[5] 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).
[6] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press. http://www.deeplearningbook.org.
[7] 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.
[8] 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.
[9] 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)*. 2333–2338.
[10] Kai Hui, Andrew Yates, Klaus Berberich, and Gerard de Melo. 2017. A Position-Aware Deep Model for Relevance Matching in Information Retrieval. In *EMNLP '17*.
[11] Kai Hui, Andrew Yates, Klaus Berberich, and Gerard de Melo. 2017. Position-Aware Representations for Relevance Matching in Neural Information Retrieval. In *Proceedings of the 26th International Conference on World Wide Web Companion*. International World Wide Web Conferences Steering Committee, 799–800.
[12] Samuel Huston and W. Bruce Croft. 2014. A Comparison of Retrieval Models using Term Dependencies. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management (CIKM'14)*. 111–120.
[13] Tie-Yan Liu et al. 2009. Learning to rank for information retrieval. *Foundations and Trends® in Information Retrieval* 3, 3 (2009), 225–331.
[14] Donald Metzler and W Bruce Croft. 2005. A Markov random field model for term dependencies. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 472–479.
[15] 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.
[16] Bhaskar Mitra, Eric Nalisnick, Nick Craswell, and Rich Caruana. 2016. A dual embedding space model for document ranking. *arXiv preprint arXiv:1602.01137* (2016).
[17] 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)*.
[18] 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
[19] 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.
[20] Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Grégoire Mesnil. 2014. Learning Semantic Representations Using Convolutional Neural Networks for Web Search. In *Proceedings of the 23rd International Conference on World Wide Web (WWW '14 Companion)*.
[21] Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).
[22] 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.
[23] Chenyan Xiong, Zhuyun Dai, Jamie Callan, Zhiyuan Liu, and Russell Power. 2017. End-to-End Neural Ad-hoc Ranking with Kernel Pooling. In *Proceedings of the 40th International ACM SIGIR Conference (SIGIR '17)*. ACM.
[24] Hamed Zamani and W Bruce Croft. 2017. Relevance-based Word Embedding. *arXiv preprint arXiv:1705.03556* (2017).