Convolutional Neural Networks for Soft-Matching N-Grams in Ad-hoc Search

Zhuyun Dai
Language Technologies Institute
Carnegie Mellon University
zhuyund@cs.cmu.edu

Chenyan Xiong
Language Technologies Institute
Carnegie Mellon University
cx@cs.cmu.edu

Jamie Callan
Language Technologies Institute
Carnegie Mellon University
callan@cs.cmu.edu

Zhiyuan Liu
Department of Computer Science and Technology
Tsinghua University
liuzy@tsinghua.edu.cn

ABSTRACT

This paper presents Conv-KNRM, a Convolutional Kernel-based Neural Ranking Model that models n-gram soft matches for ad-hoc search. Instead of exact matching query and document n-grams, Conv-KNRM uses Convolutional Neural Networks to represent n-grams of various lengths and soft matches them in a unified embedding space. The n-gram soft matches are then utilized by the kernel pooling and learning-to-rank layers to generate the final ranking score. Conv-KNRM can be learned end-to-end and fully optimized from user feedback. The learned model's generalizability is investigated by testing how well it performs in a related domain with small amounts of training data. Experiments on English search logs, Chinese search logs, and TREC Web track tasks demonstrated consistent advantages of Conv-KNRM over prior neural IR methods and feature-based methods.

ACM Reference format:
Zhuyun Dai, Chenyan Xiong, Jamie Callan, and Zhiyuan Liu. 2018. Convolutional Neural Networks for Soft-Matching N-Grams in Ad-hoc Search. In *Proceedings of WSDM'18, February 5–9, 2018, Marina Del Rey, CA, USA,* 9 pages.
DOI: http://dx.doi.org/10.1145/3159652.3159659

1 INTRODUCTION

A recent success of neural methods in information retrieval (neural IR) is the development of interaction based models [13, 21, 29]. Interaction based models thrive with encoding word-word translations using word embeddings, and utilizing new pooling methods to better summarize the word translations into ranking signals [11, 13, 29]. Learned end-to-end from user feedbacks [23, 29], the word embeddings can encode soft matches tailored for relevance ranking, which has significant advantages over traditional feature-based methods [29, 30]. These initial successes of neural IR were mainly from soft matching individual words. On the other hand, the query and document often match at n-grams, such as phrases [18], concepts [2], and entities [28]; how to effectively model n-gram soft-matches remains an open question in neural IR.

This paper presents a new Convolutional Kernel-based Neural Ranking Model(Conv-KNRM). We first embed words in continuous vectors (embeddings), and then employ Convolutional Neural Networks (CNN) to compose adjacent words' embeddings to n-gram embeddings. In the n-gram embedding space, soft-matching n-grams is as simple as calculating the similarity of two n-grams' embeddings. The current state-of-the-art kernel pooling and learning-to-rank techniques are then used to combine the n-gram soft-matches to the final ranking score [29].

The CNN is the key to modeling n-grams. Typical IR approaches treat n-grams as discrete terms and use them the same as unigrams. For example, a document bigram 'white_house' is one term, has its own term frequency, and can only be matched to 'white_house' in queries. However, treating n-grams atomically in neural IR will explode the parameter space, and suffer from data sparsity. This work avoids the problem by learning a convolutional layer that forms n-grams from individual words' embeddings. The convolutional layer projects all n-grams into a unified embedding space, allowing matching n-grams of different lengths. For instance, 'white house' in the document can provide partial evidence for the query 'George Walker Bush'.

The whole Conv-KNRM model can be trained end-to-end with relevance signals such as clicks, so that the n-gram soft matches are fully optimized towards search accuracy. We also present a simple yet effective domain adaptation method for applying Conv-KNRM to search domains where large scale training data is not available. We first train the word embedding and convolutional layers in the source domain that has sufficient training labels. The trained Conv-KNRM is then adapted to a target domain with limited annotations by only re-training the learning-to-rank layer. The assumption is that the soft matching patterns learned on one domain are likely to generalize to similar domains, while the importance of each type of soft match can vary across domains.

Our experiments on an English search log from Bing and a Chinese search log from Sogou show the advantages of soft-matching n-grams. As a precision oriented method, Conv-KNRM almost doubled the NDCG@1 of standard feature-based learning-to-rank methods. On the English log, Conv-KNRM outperformed state-of-the-art neural methods by over 30%; in the Chinese log where many unigrams

WSDM'18, February 5–9, 2018, Marina Del Rey, CA, USA
 ISBN 978-1-4503-5581-0/18/02…$15.00
DOI: http://dx.doi.org/10.1145/3159652.3159659

are already compound word, soft-matching n-grams still provided significant improvements. Our further study reveals that the key to Conv-KNRM's advantages is its ability to cross-match n-grams with different lengths in a unified space.

Our domain adaptation experiment shows that the n-gram soft matches are generalizable. When adapted to the TREC Web Track task, the pre-trained Conv-KNRM from Bing's log outperformed two strong learning-to-rank baselines. A case study found that some connections between queries and their relevant documents can only be made by soft-matching n-grams, for example, 'atypical squamous cells' and 'cervical cancer'. To the best of our knowledge, this is the first time such cross-domain generalization ability has been achieved by neural methods in ad hoc search[1].

In the rest of this paper, Section 2 discusses related work; Section 3 describes our model architecture; Section 4 describes the domain adaptation method; Experimental setups and evaluation results are presented in Section 5 and Section 6. We conclude in Section 7.

2 RELATED WORK

The current neural IR methods can be categorized into two classes: *representation* based and *interaction* based [13]. The earlier attempts of neural IR research were mainly about how to learn good *representation* of the query and document, and the ranking was simply done by their representations' similarities, for example, DSSM [15] and its convolution version CDSSM [26]. A more recent example is the weakly supervised ranking model in which all word embeddings of a query or document are combined into one vector, and the match of two vectors is done by deep neural networks [9].

The *interaction* based methods, on the other hand, directly model query-document matches at the word level. They are rooted in statistical translation models, which construct a translation matrix of word pairs between query and document, and summarize it to a ranking score [4]. The main challenge of translation models is that the word-pair translations are too sparse to learn. To overcome this problem, word embeddings [20] are introduced to calculate the translation scores [12]. How to combine the word-level translation scores to generate query-document ranking scores has also been improved by neural methods such as Convolutional Neural Networks [14, 23].

A later study found that the CNN filters tend to mix the match signals in the translation matrix at various levels and are suboptimal for ad hoc search [22]. The DRMM model introduces the histogram pooling (pyramid pooling [11]) technique to summarize the translation matrix; it demonstrated that it is more effective to 'count' the word-level translation scores at different soft match levels, instead of to weight-sum them [13]. The *interaction* based model and the *representation* based model can also be combined in a duet architecture [21].

Another trend of neural IR research is to learn customized word embeddings by and for ad-hoc ranking. The surrounding text based word embeddings, e.g. word2vec [20] and GloVe [24], have been questioned about their suitability for ad hoc search [1, 25]. Diaz et al. train word embeddings using pseudo relevance feedback (PRF) documents, which are more effective than globally trained word2vec in query expansion [10]. The relevance feedback based word embeddings are then also found to be more effective in ad hoc ranking [30].

K-NRM unified the progress of IR customized embeddings and *interaction* based model [29]. It first embeds words and builds the translation matrix using the similarities between query and document words' embeddings. Then it uses kernel-pooling to summarize the word embeddings and provide soft match signals for learning to rank. The kernel-pooling shares the advantage of pyramid pooling [13] that it 'counts' the soft matches at multiple levels, while also being differentiable so that word embeddings and ranking parameters can be learned together. When trained with user feedback in a search log, K-NRM outperforms both neural IR methods and feature-based learning-to-rank by a large margin [29].

Though the soft matching of n-grams in information retrieval remains an open topic, there has been a large amount of research that utilizes n-gram exact matches. The sequential dependency model (SDM) that includes n-gram phrase matches has been a standard in many IR systems [18]. There is also much work about how to better weight n-grams in SDM, for example, by emphasizing frequent and meaningful concepts [3, 32]. A more recent trend is to use entities to introduce explicit semantics from knowledge graphs to search systems [28]. The majority of these work focuses on exact matching n-grams, because learning a good statistical translation model score for every possible n-gram pair inevitably faces data sparsity and parameter explosion.

Modeling n-grams is much easier in the embedding space. Neural methods have shown the benefits of modeling n-grams in some related text processing tasks, especially with Convolutional Neural Networks. For example, in sentence classification, CNN has been used to compose word embeddings into n-gram representations, which are then max-pooled and combined by a feed-forward neural network to classify the sentence [17]. That research demonstrated CNN's ability of composing n-gram embeddings, while its ability in relevance ranking is still being explored.

3 CONVOLUTIONAL N-GRAM RANKING

This section presents our convolutional kernel-based neural ranking Model (**Conv-KNRM**), shown in Figure 1. It first composes n-gram embeddings using CNN, and constructs translation matrices between n-grams of different lengths in the n-gram embedding space (Section 3.1). Then it ranks with the n-gram soft matches using kernel-pooling and learning to rank (Section 3.2).

3.1 N-gram Composing and Cross-matching

Given a query q and document d, Conv-KNRM embeds their words by a word embedding layer, composes n-grams with a CNN layer, and cross-matches query n-grams and document n-grams of variant lengths to the translation matrices.

The **word embedding** layer maps each word t of a text to an L-dimensional continuous vector (embedding): $t \rightarrow \vec{t}$. A query q or document d is treated as a text sequence of m words $\{t_1, ..., t_m\}$, and is modeled as an $m \times L$ matrix:

$$T = \begin{bmatrix} \vec{t}_1 \\ ... \\ \vec{t}_m \end{bmatrix}. \quad (1)$$

[1]Trained models available at: http://boston.lti.cs.cmu.edu/appendices/WSDM2018-ConvKNRM/

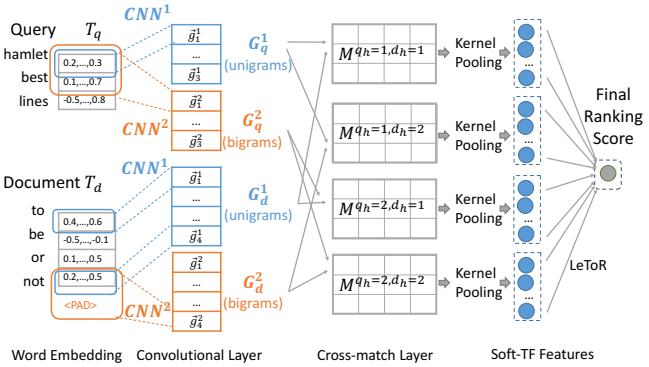

Figure 1: The Conv-KNRM Architecture. Given input query and document, the embedding layer maps their words into distributed representations, the convolutional layer generates n-gram embeddings ; the cross-match layer matches the query n-grams and document n-grams of different lengths, and forms the translation matrices; the kernel pooling layer generates soft-TF features and the learning-to-rank (LeToR) layer combines them to the ranking score. The case with Unigrams and Bigrams ($h_{max} = 2$) is shown.

We denote the embedding matrix of the query and the document by T_q and T_d respectively.

The **convolutional layer** applies convolution filters to compose n-grams from the text (T_q or T_d). A convolution filter slide over the text like a sliding window. For each window of h words, the filter sums up all elements in the h words' embeddings $T_{i:i+h}$, weighted by the filter weights $w \in \mathbb{R}^{hL}$,and produces a continuous score:

$$v = w \cdot T_{i:i+h}, v \in \mathbb{R}. \tag{2}$$

Using F different filters $w_1, ..., w_F$ gives F scores, each describing $T_{i:i+h}$ in a different perspective. Then we add a bias and apply a non-linear activation function, and obtain an F-dimensional embedding for the h-gram:

$$\vec{g}_i^h = \text{relu}\left(W^h \cdot T_{i:i+h} + \vec{b}^h\right), i = 1...m. \tag{3}$$

$\vec{g}_i^h \in \mathbb{R}^F$ is the embedding of the i-th h-gram. The f-th element in $\vec{g}_i^h$ is the score of the f-th filter. W^h and $\vec{b}^h$ are the weights of the F convolution filters. $|W^h| = (hL) \times F$ and $|\vec{b}^h| = F$. When a convolution filter slides across the boundary of the text, we append up to $h-1$ '<PAD>' symbols for padding.

Thus, for each n-gram length $h \in \{1, .., h_{max}\}$, the CNN layer converts the text embedding T into h-gram embedding G^h.

$$G^h = \text{CNN}^h(T) = \begin{bmatrix} \vec{g}_1^h \\ ... \\ \vec{g}_m^h \end{bmatrix} \tag{4}$$

$|G^h| = m \times F$. Each of its rows correspond to a h-gram vector of length F. h-gram embeddings for the query and the document are denoted as G_q^h and G_d^h respectively.

The 'convolution' assumption is applied in the n-gram compositions: the same set of convolution filters is used to compose all n-grams. Thus, instead of learning an individual embedding for each n-gram in the corpus, the model only needs to learn the CNN weights for combining word-level embeddings, which have much fewer parameters.

The **cross-match layer** matches query n-grams and document n-grams of different lengths. For query n-grams of length h_q and document n-grams of length h_d, a translation matrix M^{h_q,h_d} is constructed. Its elements are the similarity scores between the corresponding query-document n-gram pairs.

$$M_{i,j}^{h_q,h_d} = \cos\left(\vec{g}_i^{h_q}, \vec{g}_j^{h_d}\right) \tag{5}$$

The unified embedding representations allow cross-matching n-grams of different lengths, e.g., the query trigram"convolutional neural networks" and the document bigram "deep learning". It generates h_{max}^2 translation matrices.

$$\mathcal{M} = \left\{M^{h_q,h_d} | 1 \le h_q \le h_{max}, 1 \le h_d \le h_{max}\right\} \tag{6}$$

3.2 Ranking with N-gram Translations

Conv-KNRM uses the kernel-pooling technique and a learning-to-rank layer to calculate the ranking score using the n-gram translations $\mathcal{M}$. This part extends K-NRM [29] to n-grams.

Kernel-pooling is a pooling technique that uses K Gaussian kernels to count the soft matches of word or n-gram pairs at K different strength levels. Each kernel K_k summarizes the translation scores as soft-TF counts in the region defined by its mean μ_k and width δ_k. As a result, a translation matrix M is pooled to a K-dimensional soft-TF feature vector $\phi(M) = \{K_1(M), ..., K_K(M)\}$. Such counting-based pooling methods have shown better performance than score-based ones like mean-pooling or max-pooling [13, 29].

Kernel-pooling is applied to each M^{h_q,h_d} matrix in $\mathcal{M}$ to generate the soft-TF feature vector $\phi\left(M^{h_q,h_d}\right)$, which describes the distribution of match scores between query h_q-grams and document h_d-grams. This leads to the ranking features as follows.

$$\Phi(\mathcal{M}) = \phi\left(M^{1,1}\right) \oplus ... \oplus \phi\left(M^{h_q,h_d}\right) \oplus ... \oplus \phi\left(M^{h_{max},h_{max}}\right)$$

$\Phi(\mathcal{M})$ has $K \times h_{max}^2$ dimensions, K soft-TF features for each of the h_{max}^2 translation matrices in $\mathcal{M}$.

The **learning-to-rank (LeToR)** layer combines the soft-TF ranking features $\Phi(\mathcal{M})$ into a ranking score:

$$f(q,d) = \tanh\left(w_r^T \Phi(\mathcal{M}) + b_r\right) \tag{7}$$

w_r and b_r are the linear ranking parameters to learn. $|w_r| = |\Phi(\mathcal{M})|$ and $|b_r| = 1$. tanh() is the activation function.

Standard pairwise learning-to-rank is used to train the model.

$$l = \sum_q \sum_{d^+, d^- \in D_q^{+,-}} \max(0, 1 - f(q,d^+) + f(q,d^-)) \tag{8}$$

$D_q^{+,-}$ are q's pairwise preferences: d^+ ranks higher than d^-.

All of Conv-KNRM layers are differentiable; the whole model, including word embeddings ($\mathcal{V}$), CNN filters (W_h, b_h), and learning-to-rank layers (w_r, b_r) can be learned end-to-end from training data. For a model with vocabulary size $|V|$, L-dimensional word

embeddings, F filters, h_{max} maximum n-gram length and K kernels, the embedding layer has $|V| \times L$ parameters, the CNN layer has $O(|h_{max}|LF)$ parameters, and the learning-to-rank layer has $K \times h_{max} + 1$ parameters.

The main capacity of the model is in the word embedding and CNN filters. They are expected to learn the word embeddings and n-gram compositions from training data and provide desired multi-level n-gram soft matches. The learning-to-rank layer serves as a linear feature combiner as in standard feature-based ranking.

3.3 Summary

Conv-KNRM adds the ability of soft matching n-grams to the recent state-of-the-art K-NRM model [29] with convolutional neural networks (CNNs). Without CNNs, Conv-KNRM withdraws to K-NRM.

Matching n-grams is a well-established idea in information retrieval. However, n-grams are usually treated identically to words: they are atomic index terms, have distinct term frequencies, use the same weighting function as unigrams, and must match exactly in query and document [7]. If that approach is used by a neural ranker, the number of parameters to be learned can grow very large. The neural model has to deal with data sparsity and low efficiency problems, which are even harder for longer n-grams. Conv-KNRM avoids these problems by using CNNs to compose n-grams without dramatically enlarging the parameter space. It makes soft-matching n-grams convenient and efficient.

4 DOMAIN ADAPTATION

End-to-end training Conv-KNRM requires large-scale training data, for example, user clicks in a commercial search log [29] or industry-scale annotations [21]. However, for many search domains such as TREC benchmarks, such data are not available. We propose a domain adaption strategy that learns Conv-KNRM from a source domain that has sufficient training data, and then re-trains its learning-to-rank layer in the target domain with limited labels.

The parameters of the embedding and convolution layers are learned in the source domain to absorb the rich relevance signals in the training data. They are then used in the target domain to generate soft-TF features $\Phi(M)$. Xiong, et al. [29] showed that kernel-pooled soft-TF features reveal different types of soft match. For example, one kernel may count synonyms (e.g., 'oppor9' and 'OPPOR'); another kernel may count word pairs from the same concept class (e.g., 'son' and 'daughter', 'Java' and 'C++'). These soft match patterns are likely to be stable across related domains.

The learning-to-rank parameters indicate the importance of each kernel. They are re-trained on the target domain, because the importances of each type of soft matches can change over domains. For instance, the synonym kernel is of low importance in search logs as all candidate documents already contain the query words [29]; however, synonyms can be a strong signal in a recall-oriented domain.

Re-training the ranking layer in the target domain is a standard feature based learning-to-rank tasks. This allows one to add domain-specific features from the target domain. There is also no limitation on which learning to rank model to use in the target domain. One can leverage the power of any learning-to-rank model such as RankSVM [16] or LambdaMART [27].

Table 1: Query log datasets. Sogou-Log is a sample of Chinese query logs from Sogou.com in 2016. Bing-Log is a sample of English query logs from Bing.com in 2006

	Sogou-Log		Bing-Log	
	Training	**Testing**	**Training**	**Testing**
Language	Chinese		English	
Fields	Title		Title, Snippet	
Queries	95,229	1,000	99,043	1,000
Docs Per Query	12.17	30.50	50	50
Search Sessions	31M	4.1M	2.10M	0.14M
Vocabulary Size	165,877	19,079	131,225	41,940

5 EXPERIMENTAL METHODOLOGY

This section describes our datasets, how training and testing were performed, our baseline algorithms, and implementation details.

5.1 Datasets

Conv-KNRM was evaluated using two search logs in different languages (Sogou, Bing), and a TREC dataset (ClueWeb09-B).

Sogou-Log: Sogou.com is a major Chinese commercial search engine. The same settings as K-NRM were used [29]. The same sample of Sogou log and training-testing splits are used (Table 1). The testing queries were sampled from queries with more than 1000 sessions (the head); none of them were in the training set.

Documents were represented by titles. The search log did not contain document body text. Testing document's body texts were crawled, and were used by the traditional IR baselines for stronger baseline performance. Body texts of training documents were not available [29]. The Chinese text was segmented by ICTCLASS [31]; then Chinese words were treated like English words.

Bing-Log: We used a one-month sample of a 2006 Bing log from the WSDM 2009 Web Search Click Data Workshop. It contained the top 50 URLs for each query, and clicked URLs in each session. Following Sogou-Log, we split the Bing sessions into training and testing sets with no overlapping queries (Table 1). Test queries were sampled uniformly because the log contained few head queries.

Bing-Log includes documents' titles and snippets. Most snippets had 30-50 words. We can not crawl enough body texts because URLs were from 2006. All texts were tokenized and lower-cased.

ClueWeb09-B is used for domain adaptation experiments. The ClueWeb09-B corpus contains about 50 million English web documents from 2009. The TREC 2009-2012 Web Tracks created 200 queries and corresponding relevance judgments. We followed a standard re-ranking methodology in prior research [8, 28]: re-rank the top 100 candidate documents retrieved by Galago using sequential dependency model queries; the INQUERY stopword list augmented with web-specific stop words; KStemming; and spam filtering using Waterloo spam score with threshold 60. Documents were parsed by Boilerpipe using the 'KeepEverytingExtractor'. The title and the first 50 words in the body field were used to be more consistent with the source domain (Bing-Log)'s title and snippet.

5.2 In-Domain Training and Testing

Training and testing labels on Sogou-Log and Bing-Log were generated following prior research [29].

Table 2: Training and testing labels for each dataset. DCTR used the DCTR click model to infer scores that were mapped to 5 Likert scales [5]. Clicks used the sole click in a session as the binary label. TREC labels were the 5 official grades.

Dataset	Train	Test
Sogou-Log	DCTR	Testing-SAME: DCTR Testing-RAW: Clicks
Bing-Log	DCTR	Testing-SAME: DCTR Testing-RAW: Clicks
ClueWeb09-B	Embedding & CNN: Bing-Log, DCTR	TREC labels
	LeToR: TREC labels	

Training Labels: The training labels for the Sogou and Bing logs were generated by the DCTR click model from user clicks in the training sessions [5], and training preference pairs were constructed accordingly. DCTR uses the click through rate for each query-document pair as the relevance score. DCTR is a strong baseline in click model competitions [5].

Testing-SAME: This set of testing labels was generated by DCTR, as described for training data. This setting evaluates the model's ability to fit explicit user preferences.

Testing-RAW: This set of testing labels was motivated by the cascade assumption [5]. Only the clicked document in a single-click session was considered relevant. 57% of Sogou testing sessions had only one click. 92% of Bing testing sessions had only one click. The Testing-DIFF condition was omitted due to space limits; it produced results similar to Testing-RAW.

5.3 Domain Adaptation Training and Testing

NIST provides 200 queries and corresponding relevance judgments for ClueWeb09-B. The domain adaptation experiment tests how well the n-gram soft matches trained from one domain (Bing) generalize to a similar domain (ClueWeb09-B). Both datasets contain English web documents, the timespans are somewhat similar (2006 vs. 2009), and TREC queries are similar to Bing queries[2]. However, the two datasets have different documents, different indexing methods, and different the initial rankers (Bing vs. Galago); the relevance labels are also rather different (clicks vs. manual assessments).

On ClueWeb09-B, `Conv-KNRM` was first trained with the Bing log (as described above). Then the embeddings and convolution filters were 'frozen' and soft TF-features $\Phi(\mathcal{M})$ were extracted using the same kernels for ClueWeb09-B. We also included the initial retrieval score from Galago with sequential dependency model (`Galago-SDM`) to provide whole-document information as `Conv-KNRM` only uses the title and the first 50 words of the body. The learning-to-rank parameters were retrained and tested using TREC relevance judgments (Table 2), 10-fold cross-validation, and `RankSVM` to add regularization (as discussed in Section 4).

5.4 Baselines

Traditional IR baselines included standard unsupervised retrieval models and feature-based learning-to-rank models. Unsupervised methods included `BM25` and language model with Dirichlet smoothing (`Lm`), applied on the full text of Sogou documents, or the title plus snippet of Bing documents. Learning-to-rank baselines were `RankSVM` [16] and coordinate ascent (`Coor-Ascent`) [19] They used 20 features: Boolean AND; Boolean OR; Coordinate match; Cosine similarity of bag-of-words vectors; TF-IDF; BM25; language models with no smoothing, Dirichlet smoothing, JM smoothing, and two-way smoothing; all applied on the title and body (or snippet). Default parameters were used in feature extraction.

For ClueWeb09-B, we used state-of-the-art baselines from prior research [28] [3]. The baselines include Indri's language model (`Indri`), Galago with sequential dependency model queries (`Galago-SDM`), and learning-to-rank models: `RankSVM` and `Coor-Ascent`.

Neural IR baselines included `CDSSM` [26], MatchPyramid (`MP`) [23], `DRMM` [13], and `K-NRM` [29].

`CDSSM` [26] is uses CNNs to build query and document *representations* on their words' letter-tri-grams (or Chinese characters in Sogou-Log [29]). The ranking scores are calculated by the similarity between the representations.

`MP` [23] and `DRMM` [13] are both *interaction* based models built upon the embedding translation matrix. `MP` uses CNNs to directly combine the translation scores to the ranking score, while `DRMM` uses histogram pooling to count multiple levels of soft-TF, and use learning-to-rank afterwards.

`K-NRM` is a state-of-the-art neural model previously tested on the Sogou-Log dataset [29]. It uses kernel-pooling instead of `DRMM`'s histogram pooling, and learns the word embeddings and the ranking layers end-to-end. It is the main baseline in our experiments.

Among these neural IR baselines, `DRMM` and `K-NRM` were compared on the ClueWeb09-B dataset. `DRMM` uses fixed embeddings and only learns the learning-to-rank layers, and can be trained with limited training data. `K-NRM` was tested the same as `Conv-KNRM` in the domain adaption fashion. `MP` and `CDSSM` performed worse than `DRMM` on TREC data in previous studies [13, 22].

It is unfair for unsupervised [30] or pseudo-supervised [9] neural IR methods to compete with `Conv-KNRM`, which is trained end-to-end with large amount of supervisions.

5.5 Implementation Details

Model Training: All supervised traditional IR models were trained and tested using cross-validation on the testing data. On search logs, 5-fold cross validation were used to be consistent with the previous study on Sogou-Log [29]. On ClueWeb09-B, the 10-fold cross validation splits from the provided baselines were used. All `RankSVM`'s used the linear kernel with the hyper-parameter C selected from the range [0.0001, 10] on the development set. Recommended settings of `Coor-Ascent` were kept. All neural IR methods are trained on the training splits. On ClueWeb09-B, `DRMM` was cross-validated; `K-NRM` and `Conv-KNRM` was pretrained on Bing-Log, then used RankSVM with cross-validation to retrain the learning-to-rank layer.

Document Fields: On Sogou-Log, traditional IR methods used both title and body, and neural IR methods only used title [29], as discussed in section 5.1. On Bing-Log, all methods used the title and snippets. On ClueWeb09-B, all methods used title and body, except `K-NRM` and `Conv-KNRM` which used title and first 50 words

[2]NIST sampled the TREC queries from a Bing search log. We removed TREC queries from our Bing training data. Our training and testing data have no queries in common.

[3]https://boston.lti.cs.cmu.edu/appendices/SIGIR2017_word_entity_duet/

in the body as snippets, to be consistent with the source domain. When multiple fields were used, a separate set of features from each field was generated; the combination weights were learned as well.

Word Embeddings: DRMM used pre-trained word2vec embeddings from the candidate documents in the search log, or the ClueWeb corpus. MP, K-NRM, and Conv-KNRM embeddings were all learned end-to-end using the query logs. For Sogou-log, we set embedding dimension $L = 300$ [29] . For Bing-Log, we set $L = 100$ because our pilot study showed that $L = 100$ has similar performance with $L = 300$ but the training is 3 times faster.

Hyper Parameters: n-gram lengths were $h = 1, 2, 3$. Longer n-grams with $h > 3$ usually exceed the length of web search queries. The number of CNN filters F was 128; we found that F in the range of $(50, 300)$ give similar results. The kernel pooling layers in K-NRM and Conv-KNRM and the histogram pooling layer in DRMM all used 11 kernels/bins. The first one is the exact match kernel $\mu = 1, \sigma = 10^{-3}$, or bin $[1, 1]$. The other 10 kernels/bins equally split the cosine range $[-1, 1]$: the μ or bin centers were: $\mu_1 = 0.9, \mu_2 = 0.7, ..., \mu_{10} = -0.9$. The σ of the soft match bins were set to be 0.1 [29].

Model Implementation and Efficiency: The model was implemented with Tensorflow. The optimization used the Adam optimizer, with batch size 16, learning rate 0.001, and early stopping with the patience of 5 epochs. The training of Conv-KNRM took about 12 hours on an AWS GPU machine. The training time is similar with prior work using only unigrams [29]. Most computation time was spent on the embedding layer; the convolutional layer was very efficient.

6 EVALUATION RESULTS

Three experiments were conducted to analyze Conv-KNRM's performance: its ranking accuracy when trained end-to-end, contributions of n-gram soft match, and the effectiveness when adapted to new domain.

6.1 End-to-End Accuracy

The ranking accuracies of each ranking method on the Sogou-Log and Bing-Log datasets are shown in Table 3.

On **Testing-SAME**, Conv-KNRM outperformed all feature-based and neural IR baselines by large margin with statistical significance. The closest baseline is K-NRM, the non-convolutional version of Conv-KNRM, but the differences were still large. Conv-KNRM performed better in higher ranking positions: its NDCG@1 almost doubled Coor-Ascent . These results show Conv-KNRM's effectiveness when trained and tested on the same labeling scenario.

Testing-Raw evaluates the model's performance by raw user clicks. The same stable improvements of Conv-KNRM over all baselines were observed. Since this evaluation uses sessions with only one click, the MRR scores directly reflect the reciprocal rank of user-clicked documents. On Sogou-Log, the average rank of clicked documents of all methods except K-NRM and Conv-KNRM was below rank 5. K-NRM pulled the clicked document to rank 3, and Conv-KNRM further promoted it to rank 2.7. On Bing-Log, Conv-KNRM pulled the clicked document of all methods more than 1 position higher.

The only neural IR baselines that outperformed feature-based learning-to-rank are the two *interaction* based and *end-to-end* trained ones: MP and K-NRM. Although other neural IR methods can improve over unsupervised baselines, feature-based learning-to-rank methods are harder to beat; end-to-end learned embeddings and match-based techniques are necessary for current neural IR methods to provide additional improvements [22, 29, 30]

Comparing the two strong neural IR baselines, K-NRM outperforms MP by a large margin. Both methods use end-to-end learned word embeddings to build the translation matrix. The difference is that K-NRM uses kernel-pooling to summarize 'soft-TF' counts from the translation matrix, while MP directly applies the CNN to combine the translation scores. CNN in MP only has access to the translation scores in the translation matrix, for example, a 2×2 CNN filter sees the similarity scores between two adjacent query words and two adjacent document words, but not their embeddings. Our experiments and prior studies show that counting the frequencies of multi-level soft matches are more effective than weight-summing the similarities [13, 29]—"similarity does not necessarily mean relevance" [6].

Recall that Conv-KNRM is a richer model than K-NRM only because it leverages convolutional neural networks to learn the n-gram compositions and thus enable n-gram soft matches. The improvements of Conv-KNRM over K-NRM reveal the advantage of n-gram soft matches. The relative improvements on Sogou and Bing also correlate with our intuitions of n-gram's importance in Chinese and English. In Chinese, words are segmented by word segmentation tools. An important goal of Chinese word segmentation research is to cut meaningful phrases into one word. For example, 'information retrieval', 'deep learning', and 'The People's Republic of China' are all unigrams in Chinese. As a result, the gains are much larger on English than on Chinese.

6.2 Contribution of N-Gram Soft-match

This experiment studied the contribution of n-gram soft matches by comparing several Conv-KNRM's variations. Conv-KNRM composes n-grams with lengths up to h_{max} and cross-matches them in a unified embedding space. We started with K-NRM, which is Conv-KNRM without CNNs, and incrementally added bigram matches (+Bigram), trigram matches (+Trigram), cross unigram-bigram matches (+Uni-x-Bi), and cross all three n-grams' matches which is the Full Model. Results are shown in Table 4.

Longer n-gram were more effective in English. On the Bing-Log, trigrams were better than bigrams, and bigrams were better than unigrams. The effect was weaker in Chinese, with mixed performances on different settings. Presumably it is because many Chinese phrases were glued to one word by word segmentation.

Cross matching n-grams of different lengths boosted accuracy in both languages. +Uni-x-Bi performed significantly better than +Bigram on most metrics, and Full Model outperformed all other variants significantly. Cross matching is effective because related concepts do not necessarily have the same length, e.g. 'FIFA' and 'world cup'. Composing n-grams using CNNs makes cross matching simple: all n-grams, despite with different lengths, are represented and matched in the same embedding space.

6.3 Domain Adaption

Our third experiment examined the effectiveness of Conv-KNRM when adapted to a domain where large scale training data is not

Table 3: Ranking accuracy of Conv-KNRM and baseline methods. Relative performances compared with K-NRM are in percentages. †, ‡, §, ¶, ∗ indicate statistically significant improvements over Coor-Ast†, DRMM‡, CDSSM§, MP¶ and K-NRM∗, respectively.

Method	Sogou-Log						Bing-Log					
	Testing-SAME				Testing-Raw		Testing-SAME				Testing-RAW	
	NDCG@1		NDCG@10		MRR		NDCG@1		NDCG@10		MRR	
BM25	0.142	−45%	0.287	−34%	0.228	−33%	0.043	−79%	0.123	−63%	0.102	−61%
RankSVM	0.146	−44%	0.309	−29%	0.224	−34%	0.128	−39%	$0.266^{\ddagger}$	−20%	0.207	−22%
Coor-Ascent	$0.169^{\ddagger\S}$	−34%	$0.355^{\ddagger\S}$	−16%	0.242	−29%	0.142	−32%	$0.268^{\ddagger}$	−20%	0.208	−22%
DRMM	0.137	−51%	0.315	−27%	0.234	−31%	0.137	−34%	0.247	−26%	0.200	−25%
CDSSM	0.144	−44%	$0.333^{\ddagger}$	−23%	0.232	−32%	0.156	−25%	0.273	−18%	0.212	−20%
MP	$0.218^{\dagger\ddagger\S}$	−15%	$0.379^{\dagger\ddagger\S}$	−12%	0.240	−29%	$0.182^{\dagger\ddagger\S}$	−12%	$0.301^{\dagger\ddagger\S}$	−10%	$0.244^{\dagger\ddagger\S}$	−8%
K-NRM	$0.264^{\dagger\ddagger\S\P}$	−−	$0.428^{\dagger\ddagger\S\P}$	−−	$0.338^{\dagger\ddagger\S\P}$	−−	$0.208^{\dagger\ddagger\S\P}$	−−	$0.334^{\dagger\ddagger\S\P}$	−−	$0.265^{\dagger\ddagger\S\P}$	−−
Conv-KNRM	$\mathbf{0.336}^{\dagger\ddagger\S\P*}$	+30%	$\mathbf{0.481}^{\dagger\ddagger\S\P*}$	+11%	$\mathbf{0.358}^{\dagger\ddagger\S\P*}$	+5%	$\mathbf{0.300}^{\dagger\ddagger\S\P*}$	+44%	$\mathbf{0.437}^{\dagger\ddagger\S\P*}$	+31%	$\mathbf{0.354}^{\dagger\ddagger\S\P*}$	+34%

Table 4: Ranking accuracy of Conv-KNRM variants. Relative performances compared with Unigram-only model (K-NRM) are in percentages. †, ‡, §, ¶ indicate statistically significant improvements over Unigram†, +Bigram‡, +Trigram§ and +Uni-x-Bi¶, respectively.

Conv-KNRM Variant	Sogou-Log						Bing-Log					
	Testing-SAME				Testing-Raw		Testing-SAME				Testing-RAW	
	NDCG@1		NDCG@10		MRR		NDCG@1		NDCG@10		MRR	
Unigram	0.264	−−	0.428	−−	0.338	−−	0.208	−−	0.334	−−	0.265	−−
+Bigram	$0.287^{\dagger}$	+11%	0.442	+2%	0.314	−8%	$0.235^{\dagger}$	+13%	$0.385^{\dagger}$	+15%	$0.301^{\dagger}$	+14%
+Trigram	$0.286^{\dagger}$	+11%	$0.454^{\dagger}$	+5%	$0.330^{\ddagger}$	−3%	$0.252^{\dagger}$	+21%	$0.399^{\dagger}$	+20%	$0.318^{\dagger\ddagger}$	+20%
+Uni-x-Bi	$0.308^{\dagger}$	+19%	$0.458^{\dagger\ddagger}$	+6%	$0.346^{\ddagger\S}$	+2%	$0.275^{\dagger\ddagger}$	+32%	$0.417^{\dagger\ddagger\S}$	+25%	$0.335^{\dagger\ddagger\S}$	+26%
Full Model	$0.336^{\dagger\ddagger\S\P}$	+30%	$0.481^{\dagger\ddagger\S\P}$	+11%	$0.358^{\dagger\ddagger\S}$	+5%	$\mathbf{0.300}^{\dagger\ddagger\S\P}$	+44%	$\mathbf{0.437}^{\dagger\ddagger\S\P}$	+31%	$\mathbf{0.354}^{\dagger\ddagger\S\P}$	+34%

Table 5: Performance on ClueWeb09-B using domain adaptation. Relative performance in percentages are compared to Coor-Ascent. W(in)/T(ie)/L(oss) to Coor-Ascent are compared at NDCG@20. †, ‡, §, ¶∗ indicate statistically significant improvements over Indri†, Galago SDM‡, RankSVM§, Coor-Ascent ¶ and DRMM+SDM ∗.

Method	ClueWeb09-B												
	NDCG@1		ERR@1		NDCG@10		ERR@10		NDCG@20		ERR@20		W/T/L
Indri	0.239	−6%	$0.062^{\ddagger*}$	−12%	0.229	−15%	0.130	−15%	0.236	−12%	0.139	−14%	68/31/101
Galago-SDM	0.219	−14%	0.053	−25%	0.238	−11%	0.130	−16%	$0.250^{\dagger}$	−7%	0.139	−14%	63/39/98
RankSVM	0.236	−7%	0.064^{*}	−10%	$0.256^{\dagger\ddagger}$	−5%	$0.146^{\dagger\ddagger}$	−5%	$0.263^{\dagger\ddagger*}$	−2%	$0.154^{\ddagger*}$	−5%	82/47/71
Coor-Ascent	$0.255^{\dagger\ddagger*}$	−−	0.071	−−	$0.268^{\dagger\ddagger}$	−−	0.154	−−	$0.268^{\dagger\ddagger}$	−−	0.162	−−	−/−/−
DRMM+SDM	0.215	−16%	0.049	−31%	$0.261^{\dagger\ddagger}$	−3%	0.136	−12%	0.243	−9%	0.138	−15%	66/34/100
K-NRM	0.235	−8%	0.057	−20%	$0.264^{\dagger\ddagger}$	−2%	0.140	−9%	$0.269^{\dagger\ddagger*}$	+0%	0.149	−8%	69/42/89
Conv-KNRM-exact	0.231	−10%	0.064^{*}	−10%	$0.263^{\dagger\ddagger}$	−2%	$0.146^{\dagger\ddagger*}$	−5%	$0.270^{\dagger\ddagger*}$	+1%	$0.155^{\dagger\ddagger*}$	−4%	78/42/80
Conv-KNRM	$\mathbf{0.294}^{\dagger\ddagger\S\P*}$	+15%	$0.093^{\dagger\ddagger\S\P*}$	+31%	$\mathbf{0.289}^{\dagger\ddagger\S\P*}$	+8%	$0.172^{\dagger\ddagger\S\P*}$	+12%	$0.287^{\dagger\ddagger\S\P*}$	+7%	$0.181^{\dagger\ddagger\S\P*}$	+12%	88/38/74

available. We trained Conv-KNRM's word embeddings and convolution filters in Bing-Log with a large amount of user preference labels from clicks, and re-trained the learning-to-rank part on ClueWeb09-B's TREC ranking labels. Results are shown in Table 5.

Indri and Galago-SDM are two unsupervised baselines. RankSVM, Coor-Ascent, and DRMM+SDM are supervised baselines. They used in-domain training with cross-validation, because their parameters can be learned with TREC-scale training labels. DRMM+SDM is a variant of DRMM that uses Galago-SDM score as an additional feature; it showed higher performance than the standard DRMMM. Conv-KNRM and K-NRM were both trained using domain adaption. We also examined Conv-KNRM-exact which only uses exact-matches of n-grams, e.g. 'world cup' can only be matched to 'world cup'.

As shown in Table 5, K-NRM was not able to beat DRMM+SDM, meaning that the effectiveness of unigram level soft matches was weakened by domain differences. Conv-KNRM-exact performed about the same, and was weaker than learning-to-rank approaches. It does no more than exact phrase matching as in SDM. Conv-KNRM differs from K-NRM and Conv-KNRM-exact by soft-matching n-grams; it outperformed the two strong traditional learning-to-rank models. The results demonstrate that the learned n-gram soft matches of Conv-KNRM can generate to a different domain.

Feature Weight Analysis: to further study the domain adaption, we investigated the importance of Conv-KNRM soft-TF features ($\Phi(\mathcal{M})$) in the adapted model. If the soft n-gram matching patterns learned from the Bing-Log is generalizable, their soft-TF feature

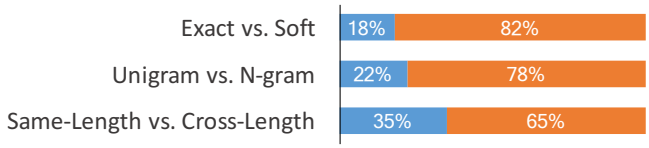

Figure 2: Learned weights of different parts of ranking features on ClueWeb09-B. The percentage is the fraction of absolute weights on each side learned by linear RankSVM.

weights would share a reasonable chunk of learning to rank weights in the adapted model.

We verified this by analyzing the weights RankSVM assigned to different groups of ranking features, as shown in Figure 2. Each analysis divides features into two groups, e.g. exact-match features and soft-match features. It then calculates the percentage of weight given to each type of features by summing up the absolute weight values of the feature set. In Figure 2, most of the weight goes to soft matches (Exact v.s. Soft); N-gram matches have more weight than unigram matches (Unigram vs. N-gram); and matching n-grams of different lengths is important compared to matching n-grams with same length (Same-length v.s. Cross-length). The high feature weights on soft n-gram match features reveals that these features do provide useful information to the learning-to-rank model—more useful than n-gram exact matches, despite that the n-gram soft matches were trained and tested on two rather different domains: different labels, different documents, and non-overlapping queries.

Case Studies: We performed case studies to better understand the soft n-gram matching. Table 6 shows examples of relevant documents that are correctly placed at rank 1 by Conv-KNRM, but not by RankSVM and KNRM. By sorting n-gram pairs according to each feature's individual performance, we can find the most important soft match that makes the document ranked highly, as highlighted in Table 6. These cases demonstrated the effectiveness in Conv-KNRM. First, Conv-KNRM overcomes the lexical mismatch, and finds query-document connections that are difficult for exact-match-based approaches, e.g. 'sewing instructions' and 'quilting 101'. Second, Conv-KNRM captures n-gram matches that are different with word matches like K-NRM. For example, ('atypical squamous', 'cervical cancer') is a strong match, but the connection between their unigram pairs, e.g. ('atypical', 'cervical'), are much weaker. These examples also illustrates Conv-KNRM's generalizability: the matchings make sense in various contexts than just in one dataset.

In summary, the domain adaptation experiment provides a thorough view of the generalization ability of Conv-KNRM. The evaluations on ClueWeb09-B shows that the cross-domain soft n-gram matching provides significant gains over in-domain feature-based learning-to-rank. Feature weight analysis demonstrates that the adapted model puts the majority of feature weights on n-gram soft match signals. Case studies prove that the learned soft-matches are intuitive and cover universally meaningful information needs. To the best of our knowledge, this is the first time we have seen such generalization ability in neural IR models.

7 CONCLUSION

This paper presents Conv-KNRM, a convolutional kernel-based neural ranking model that models n-gram soft matches for ad hoc search. Treating n-grams as discrete index terms faces the problem of dimension explosion and data sparsity. In contrast, Conv-KNRM uses Convolutional Neural Networks to compose n-gram embeddings from word embeddings, and cross-matches n-grams of various lengths in the unified embedding space. It then applies kernel pooling to extract ranking features, and uses learning-to-rank to obtain the final ranking score.

Our experiments on Chinese and English search logs demonstrate the advantages of soft-matching n-grams in relevance ranking. Gonv-GRAM almost doubled the NDCG@1 scores compared to feature-based ranking approaches, and outperformed the previous state-of-the-art model by over 30% at the top. Trained end-to-end with user feedback, Conv-KNRM learns n-gram soft match patterns tailored for matching queries and relevant documents, for example, the query 'farm' is matched to 'eat & drink'. Such IR-customized n-gram soft-match has not been seen much in previous work.

Based on our analysis, the key to Conv-KNRM's advantages is cross-matching n-grams of different lengths. Cross-matching consistently outperformed its non-cross-matching variants. On the Chinese search log, Conv-KNRM without cross-matching is about the same as its unigram competitor K-NRM, due to the phrase-like characteristics of Chinese unigrams. Cross-matching is important because related concepts do not necessarily have the same number of words, for instance, 'deep learning' and 'convolutional neural network'. But there has been little study on it due to the limitation of discrete n-gram representation. The CNN approach of modeling n-grams makes cross-matching feasible, efficient, and effective.

Beyond the good performance when trained end-to-end in domain, we show that the model trained on one domain is also generalizable to a related search domain. The model learned from Bing-log significantly outperformed strong learning-to-rank baselines when adapted to TREC Web Track task, despite important domain differences including corpus, queries and evaluation conditions. Experiments show that the embedding and CNN layers can be directly used in another related domain to generate n-gram soft-matching features. Further analysis explains the generalizability: the learned n-gram soft-matching patterns encode universal properties of language usage in ad hoc search tasks, and provide important evidences for relevance ranking even when used across domains.

8 ACKNOWLEDGMENTS

This research was supported by National Science Foundation (NSF) grant IIS-1422676. We thank Shane Culpepper and RMIT for providing the computational environment that enabled this work. Any opinions, findings, and conclusions in this paper are the authors' and do not necessarily reflect those of the sponsors.

Table 6: Examples of matched n-grams between query and snippets. Black phrases contribute more to the relevance score than gray ones.

Query	Snippet
sewing instructions	...home free resources! newsletter sewing ideas...quilting 101 what is a quilt...
atypical squamous cells	...treatment decision tools cervical cancer : prevention and early detection...
moths	... grouping of moth families commonly known as the 'smaller moths ' (micro , lepidoptera)...
fickle creek farm	.. bed & breakfast inns extended stay lodging rv parks where to eat & drink nightlife ...
university phoenix	campus locations programs : bachelor degree masters degrees account degrees business degree...
wedding budget calculator	...planning tips photographs bridal board my perfect planner tools my check lists...

REFERENCES

[1] Qingyao Ai, Liu Yang, Jiafeng Guo, and W Bruce Croft. 2016. Analysis of the paragraph vector model for information retrieval. In *Proceedings of the 2016 ACM on International Conference on the Theory of Information Retrieval (ICTIR 2016)*. ACM, 133–142.

[2] Michael Bendersky, Donald Metzler, and W. Bruce Croft. 2011. Parameterized concept weighting in verbose queries. In *Proceedings of the 34th annual international ACM SIGIR conference on Research and Development in Information Retrieval (SIGIR 2011)*. ACM.

[3] Michael Bendersky, Donald Metzler, and W Bruce Croft. 2012. Effective query formulation with multiple information sources. In *Proceedings of the Fifth ACM International Conference on Web Search and Data Mining (WSDM 2012)*. ACM, 443–452.

[4] Adam Berger and John Lafferty. 1999. Information Retrieval as statistical translation. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*. ACM, 222–229.

[5] Aleksandr Chuklin, Ilya Markov, and Maarten de Rijke. 2015. Click Models for Web Search. *Synthesis Lectures on Information Concepts, Retrieval, and Services* 7, 3 (2015), 1–115.

[6] Nick Craswell, W Bruce Croft, Jiafeng Guo, Bhaskar Mitra, and Maarten de Rijke. 2017. Report on the SIGIR 2016 Workshop on Neural Information Retrieval (Neu-IR). In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*. ACM.

[7] W Bruce Croft, Donald Metzler, and Trevor Strohman. 2010. *Search Engines: Information Retrieval in Practice*. Addison-Wesley Reading.

[8] Jeffrey Dalton, Laura Dietz, and James Allan. 2014. Entity Query Feature Expansion using Knowledge Base Links. In *Proceedings of the 37th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2014)*. ACM, 365–374.

[9] Mostafa Dehghani, Hamed Zamani, Aliaksei Severyn, Jaap Kamps, , and W. Crof, Bruce. 2017. Neural Ranking Models with Weak Supervision. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*. ACM.

[10] Fernando Diaz, Bhaskar Mitra, and Nick Craswell. 2016. Query Expansion with Locally-Trained Word Embeddings. In *Proceedings of the 54th Annual Meeting of the Association for Computational (ACL)*. ACL.

[11] Kristen Grauman and Trevor Darrell. 2005. The pyramid match kernel: Discriminative classification with sets of image features. In *Tenth IEEE International Conference on Computer Vision (ICCV) Volume 1*, Vol. 2. IEEE, 1458–1465.

[12] Jiafeng Guo, Yixing Fan, Qingyao Ai, and W Bruce Croft. 2016. Semantic Matching by Non-Linear Word Transportation for Information Retrieval. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management (CIKM)*. ACM.

[13] Jiafeng Guo, Yixing Fan, Ai Qingyao, 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 (CIKM)*. ACM.

[14] 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 (NIPS)*.

[15] Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. 2013. Learning deep structured semantic models for web search using click through data. In *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management (CIKM)*. ACM, 2333–2338.

[16] Thorsten Joachims. 2002. Optimizing search engines using clickthrough Data. In *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2002)*. ACM, 133–142.

[17] Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (2014-10). Association for Computational Linguistics.

[18] 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 (SIGIR 2005)*. ACM, 472–479.

[19] Donald Metzler and W Bruce Croft. 2007. Linear feature-based models for information retrieval. *Information Retrieval* (2007).

[20] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 2tth Advances in Neural Information Processing Systems 2013 (NIPS)*.

[21] Bhaskar Mitra, Fernando Diaz, and Nick Craswell. 2017. Learning to Match Using Local and Distributed Representations of Text for Web Search. In *Proceedings of the 25th International Conference on World Wide Web (WWW)*. ACM.

[22] Liang Pang, Yanyan Lan, Jiafeng Guo, Jun Xu, and Xueqi Cheng. 2016. A study of matchpyramid models on ad-hoc retrieval. *arXiv preprint arXiv:1606.04648* (2016).

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

[24] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation. In *Empirical Methods in Natural Language Processing (EMNLP)*. 1532–1543.

[25] Navid Rekabsaz, Mihai Lupu, Allan Hanbury, and Hamed Zamani. 2017. Word Embedding Causes Topic Shifting; Exploit Global Context!. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*. ACM.

[26] 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)*. ACM.

[27] Qiang Wu, Christopher JC Burges, Krysta M Svore, and Jianfeng Gao. 2010. Adapting boosting for information retrieval measures. *Information Retrieval* (2010).

[28] Chenyan Xiong, Jamie Callan, and Tie-Yan Liu. 2017. Word-Entity Duet Representations for Document Ranking. In *Proceedings of the 40th annual international ACM SIGIR conference on Research and Development in Information Retrieval (SIGIR 2017)*. ACM.

[29] 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 annual international ACM SIGIR conference on Research and Development in Information Retrieval (SIGIR 2017)*. ACM.

[30] Hamed Zamani and W. Croft, Bruce. 2017. Relevance-based Word Embedding. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*. ACM.

[31] Hua-Ping Zhang, Hong-Kui Yu, De-Yi Xiong, and Qun Liu. 2003. HHMM-based Chinese lexical analyzer ICTCLAS. In *Proceedings of the second SIGHAN workshop on Chinese language processing*. ACL.

[32] Guoqing Zheng and Jamie Callan. 2015. Learning to Reweight Terms with Distributed Representations. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2015)*. ACM.