## What does Conv-KNRM stand for?

According to (Dai, et al., 2018) Conv-KNRM stands for Convolutional Kernel-based Neural Ranking Model.

## What are Conv-KNRM used for?

According to (Dai, et al., 2018) “(…) Conv-KNRM (…) [is a] 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 ngrams of various lengths and soft matches hem in a unified embedding space. Thee n-gram soft matches are then utilized by the kernel pooling and learning-to-rank layers to generate the final ranking score.”

## How do interaction based models work?

According to (Dai, et al., 2018) interaction based models work by “(…) encoding word-word translations using word embeddings, and utilizing new pooling methods to better summarize the word translations into ranking signals.”.

## How does pooling work?

## What is soft-matching?

## What are n-grams?

## What are n-gram embeddings?

## How does the Conv-KNRM model work in general?

According to (Dai, et al., 2018) “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 ngrams is as simple as calculating the similarity of two n-grams’ embeddings. The current state-of-the-art kernel pooling and learningto-rank techniques are then used to combine the n-gram soft- matches to the final ranking score [29].”.

## What is the purpose of the CNN in the Conv-KNRM?

According to (Dai, et al., 2018) “[T]he 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’.”.

## How can the Conv-KNRM be trained?

According to (Dai, et al., 2018) “[t]he 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.”.

## How can IR methods be categorized?

According to (Dai, et al., 2018) “[t]he current neural IR methods can be categorized into two classes: representation based and interaction based [13].”.

## How do representation based methods work?

According to (Dai, et al., 2018) “[t]he 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].”.

## How do interaction based methods work?

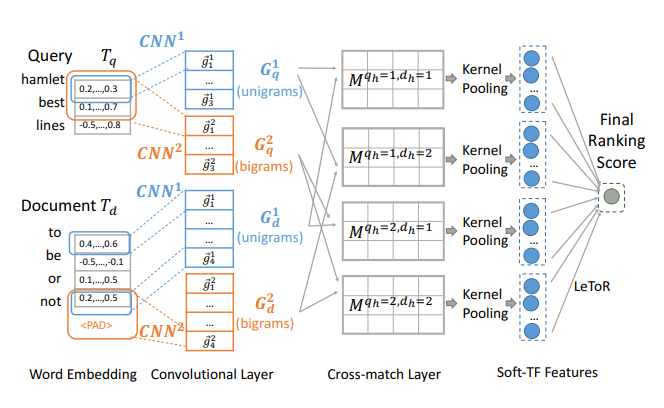
According to (Dai, et al., 2018) “[t]he interaction based methods, on the other hand, directly model query-document matches at the word level. They are rooted in statistical translation models, which construct a translation matrix of word pairs between query and document, and summarize it to a ranking 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].”.

## How does a KNRM work?

According to (Dai, et al., 2018) “[i]t first embeds words and builds the translation matrix using the similarities between query and document words’ embeddings. Often 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].”.

## How does the Conv-KNRM work in detail?

According to (Dai, et al., 2018) “It first composes ngram 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).”.

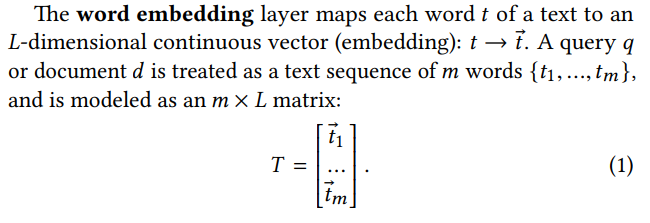


Note. This image is from (Dai, et al., 2018).

According to (Dai, et al., 2018) the KNRM consists of 5 layers.

1. Word embedding layer
2. Convolutional layer
3. Cross-match layer
4. Kernel-pooling layer
5. Learning-to-rank (LeToR) layer

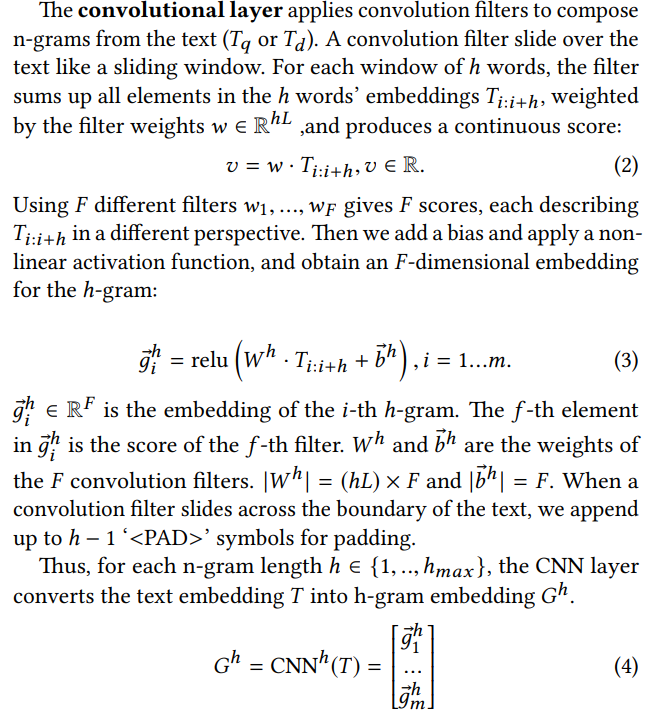
### Word Embedding Layer

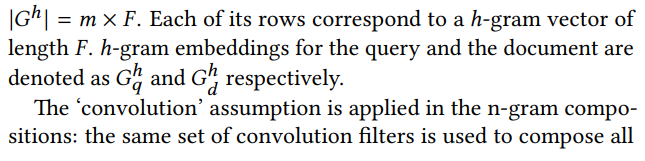


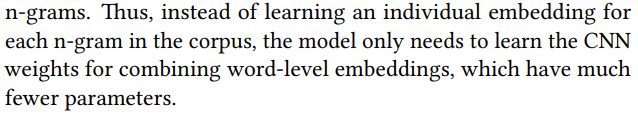


Note. The images are from (Dai, et al., 2018).

### Convolutional Layer

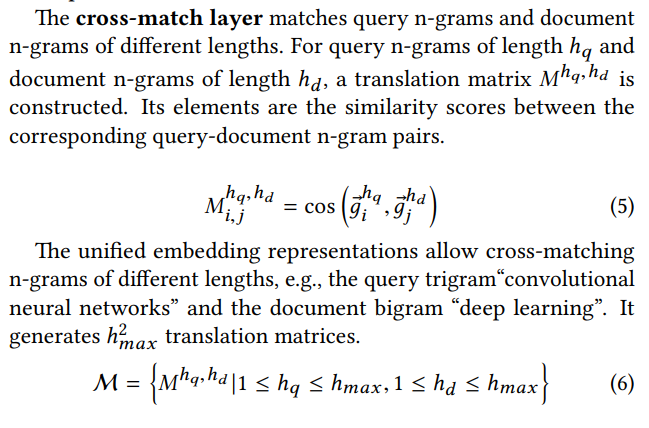






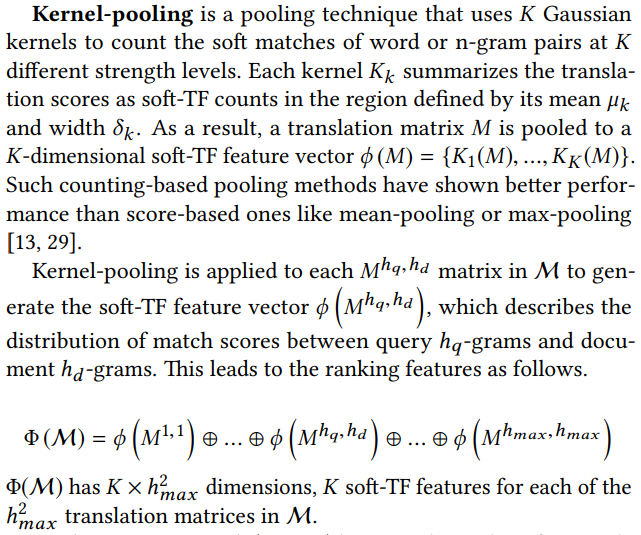
Note. The images are from (Dai, et al., 2018).

### Cross-Match Layer



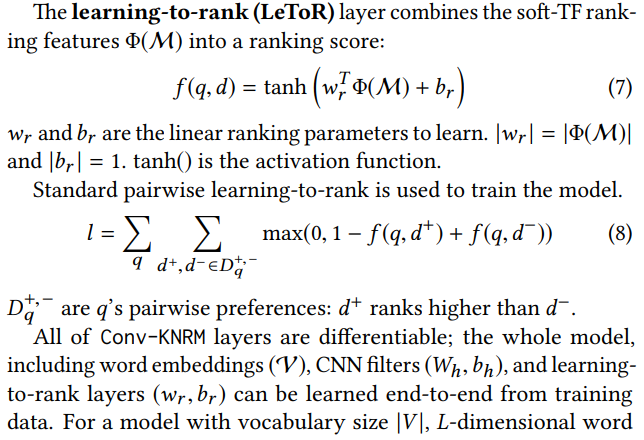
Note. The images are from (Dai, et al., 2018).

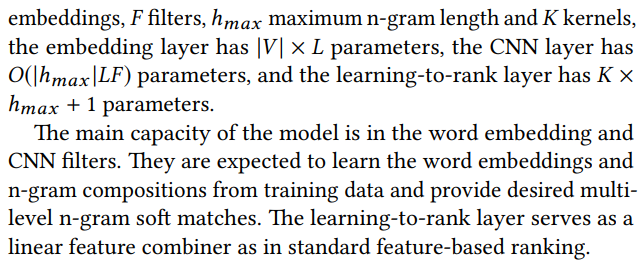
### Kernel-Pooling Layer



Note. The images are from (Dai, et al., 2018).

### Learning-to-Rank (LeToR) Layer





Note. The images are from (Dai, et al., 2018).

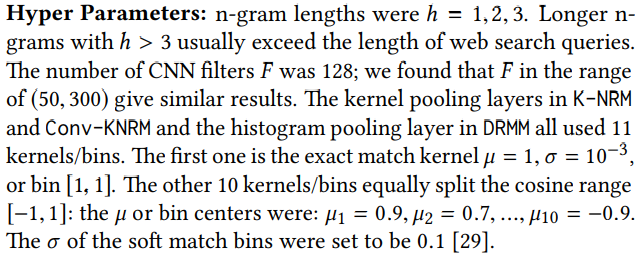
## How should the Conv-KNRM be trained?

According to (Dai, et al., 2018) “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.”.

## How should the word embeddings be generated?

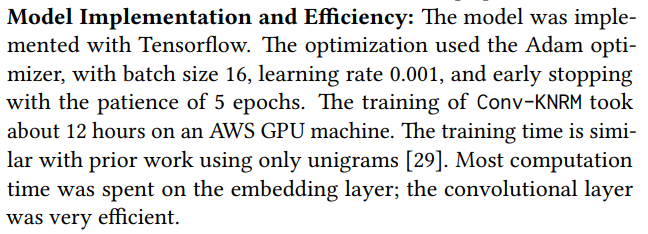
According to (Dai, et al., 2018) “DRMM used pre-trained word2vec embeddings from the candidate documents in the search log, or the ClueWeb corpus. MP, K-NRM, and Conv-KNRM embeddings were all learned end-to-end using the query logs. For Sogou-log, we set embedding dimension L = 300 [29] . For Bing-Log, 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.”.

## What hyperparameter settings should be used?



Note. This image is from (Dai, et al., 2018).

## How should the Conv-KNRM be implemented?



Note. This image is from (Dai, et al., 2018).

## Which evaluation metrics should be used?

(Dai, et al., 2018) used NDCG as evaluation metric.