Approaches

In this section, state-of-the-art IR: BM25, DRMM and K-NRM are briefly summarized.
BM25 [1] is an effective retrieval weighting scheme that represents the classical probabilistic retrieval model. BM25 has tunable parameters b and k1 [1] (11.32) which governs the effect of term frequency and document length respectively.

$$RSV_d = \sum_{t \in q} \log \left[\frac{N}{\mathrm{df}_t} \right] \cdot \frac{(k_1 + 1)\mathrm{tf}_{td}}{k_1((1-b) + b \times (L_d / L_{\mathrm{ave}})) + \mathrm{tf}_{td}} \tag{11.32}$$

For more effective feature engineering, a variety of neural models have been proposed to cope with IR tasks. Until recently, there had been only a few good results based on deep learning approaches in IR. The proposed deep models mostly took IR as a general matching problem between two pieces of text corpora and applied some semantic matching properties into IR tasks.

Jiafeng Guo et al. [4] point out the problem, list three fundamental contradictory properties between semantic matching (NLP tasks) and relevance matching (IR tasks), and propose DRMM addressing the desired properties explicitly. It takes local interactions with prior embedding, summarizes the interactions into histograms (counting, normalized, log based) and feeds the histogram bins into a feed-forward network to get matching scores. Finally, the matching scores are aggregated by term gate network trained with TV or IDF input which re-ranks them according to the importance of the query terms. Especially, LogCount-based Histogram and IDF input for term gate network training (DRMMlch*idf) performs the best among the DRMM combinations.

Chenyan Xiong et al. [5] on the other hand point out that many deep models using already-trained distributed representations have achieved limited positive results in IR. They proposed the end-to-end model K-NRM which involves embedding learning guided by the differentiable RBF kernels.

Strengths and Weaknesses

In this section, BM25, DRMM and K-NRM are evaluated with some IR axioms [2] and other desirable properties of the IR system.

TFC1

In BM25 weighting scheme, it explicitly formulates that documents with higher document term frequency get higher scores. In the following, the relationship between scores and different tf values with the rest parameter being fixed are demonstrated:

k1 = 2, tf = 1: 3*1 / 3 = 1
k1 = 2, tf = 2: 3*2 / 4 = 3 / 2
k1 = 2, tf = 5: 3*5 / 7 = 15 / 7
On the other hand, both DRMM and K-NRM model query term level similarity with embeddings and extract the features with pooling. In the pooling step, DRMM has one bin of similarity [1, 1] for exact match in the histogram mapping step while K-NRM has one kernel μ = 1, sufficiently small σ = 10^-3 for exact match. With a pairwise ranking loss, both DRMM and K-NRM can learn the aggregation weights for the exact match which leads to both models favoring documents with more occurrence of a query term.

TFC2
BM25 satisfies TFC2 with appropriate k1. In the following, the interaction between different k1 values with a fixed term frequency are explored given that the rest of the terms are fixed:
k1 = 0.5, tf = 2: 1.5*2 / 2.5 = 6 / 5
k1 = 1, tf = 2: 2*2 / 3 = 4 / 3
k1 = 2, tf = 2: 3*2 / 4 = 3 / 2
k1 = 3, tf = 2: 4*2 / 5 = 8 / 5
k1 = 10, tf = 2: 11*2 / 12 = 11 / 6
We can see that k1 limits how much a single query term can affect the score of a given document. If k1 is smaller, the additional term frequency would contribute less to the score and if k1 is bigger, the additional term frequency would contribute more to the score. k1 represents a change of slope of score in terms of term frequency.

In the histogram mapping step of DRMM, logcount-based (LCH) mapping limits the effect of term frequency. That is as term frequency goes higher and higher, it is going to converge to a constant contribution with LCH. In fact, the empirical results show that NH-based models perform significantly worse than CH-based models, and LCH is a better choice among the three histogram mappings. Similarly, in K-NRM the extracted feature is log-scaled which avoids the exaggerated effect of high term frequency. This mechanism also implicitly makes the model favor more distinct query terms.

TDC
BM25, The IDF approximation part penalizes query terms that are common. Thus, given two documents with the same occurrences of query terms (modified sum of term frequency), the document that has more occurrences of discriminative terms would be favored.

DRMM models query term importance using term gate network. If the term gate network is trained with IDF input, then the model would favor documents containing query terms that give more information. Also, the results show that IDF input performs better than TV (term vector).

K-NRM does not contain a devoted component to deal with query term importance. In the whole model, there are no weights for individual query terms. Chenyan Xiong et al. [5] have also experimented with the model using IDF to weight query words when combining their kernel pooling, and the result is not as good as the simple K-NRM. Introducing some weights in the log-sum to aggregate query terms might be an approach to implicitly include the IDF factor. However, obviously the approach comes at the cost of increasing the number of weights.

LNC-1, LNC-2, TF-LNC:
BM25 has a denominator in the term frequency part that determines the document term frequency scaling with respect to the document length. The term L_d / L_{ave} describes the general scaling direction of the term frequency. If a document is longer than average, the denominator gets bigger and the term frequency is scaled down. On the other hand, if a document is shorter than average, the term frequency is scaled up. The parameter b is a tunable parameter which determines the quantity of the term frequency scaled by document length. If b is bigger, the scaling effect of L_d / L_{ave} is more amplified.

In the histogram mapping step of DRMM, three ways of approaches are explored: count-based (CH), normalized-based (NH), logcount-based (LCH) and these three extract features differently in regards to document length. CH-based mapping does not re-weight the term frequency so it most likely is going to favor long documents. NH-based mapping maps the number of counts to proportion and thus the mapping signal shrinks if the document contains a lot more other words. While normalized mapping penalizes long documents, it might be overdoing it. LCH-based mapping penalizes long documents in a desirable fashion. As term frequency goes higher and higher, it is going to converge to a constant contribution with LCH. In fact, the empirical results show that NH-based models perform significantly worse than CH-based models, and LCH is a better choice among the three histogram mappings.

After kernel-pooling, K-NRM aggregates the soft term frequency features with log sum. Similarly as done in DRMM, taking log of the counts can be seen as a penalization to long documents and as a limitation so that term frequency growth are effective till a certain level. Also, C. Xiong et al. [6] have implemented K-NRM with max document length (default: 50) where no long document is considered at the first place. Taking into account that snippets in the searching engine are short and approximately within this default length, K-NRM can be competitive in this IR task.

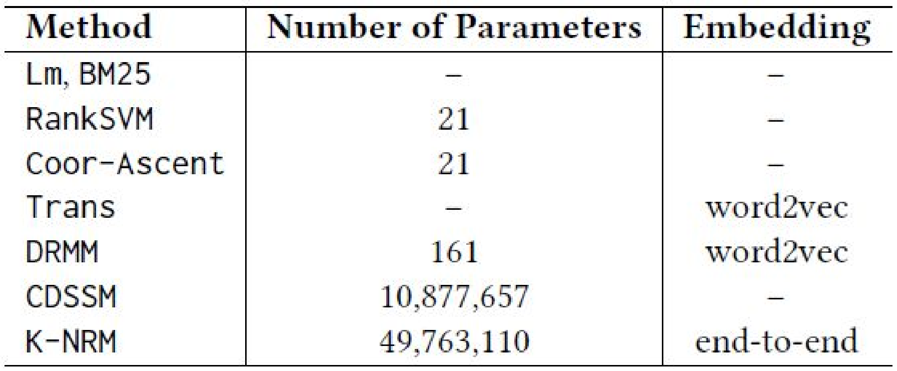
Preprocessing documents to the same length with chopping and padding seems simple and effective to cope with document length issues. However, this might lead to some undesirable situations where the paddings are learned. It might be even messier to resolve this side effect. Instead, processing features though Logarithm is a rather elegant approach.

Rare Query
As Bhaskar et al. [7] points out that many IR models that learn latent representations are not robust to rare query terms. DRMM is fed with pre-trained embedding which has limited vocabulary size, for example in the paper [4], two TREC collections are used, Robust04 and ClueWeb-09-Cat-B with vocabulary sizes 0.6M and 38M. K-NRM also has to assume vocabulary size beforehand. The empirical results [5] show an expected decline of K-NRM's performance on rarer queries. K-NRM performs poorly when the query consists of terms rarely seen or unseen in the training data. It is hard to generalize to the rare term through the embedded relevance signals. However, exact matching models, like BM25 [1], on the other hand can precisely retrieve documents containing rare terms.

Bigram / Trigram Matches
For complete meaning of some phrases, sequentially matching the query terms is desirable. BM25, DRMM and K-NRM are not position-aware and also do not weigh higher for continuous matches. The embedding layer can be substituted with bigram or trigram representation to extract better semantic interactions.

Efficiency
In the setting of the experiment [5], K-NRM only needs to re-rank between 30 documents during testing and is trained with 12. In the setting of experiment [4], DRMM re-ranks the top 2000 documents pre-ranked by the QL model. K-NRM is more expensive to train and with poor retrieval efficiency.

Complexity
While DRMM and KNRM are neural models that learn from latent representations, DRMM uses already-trained embedding while KNRM is trained end-to-end with embedding learning. The essential structural difference is the number of parameters being introduced. DRMM has parameters (W, b) in ranking network for bins and (g) in term gate network for query terms; K-NRM has parameters (W, b) in ranking network and main capacity for embeddings (V*K).
For example, the experiment in paper Table 3 [5], summarizes the number of parameters:



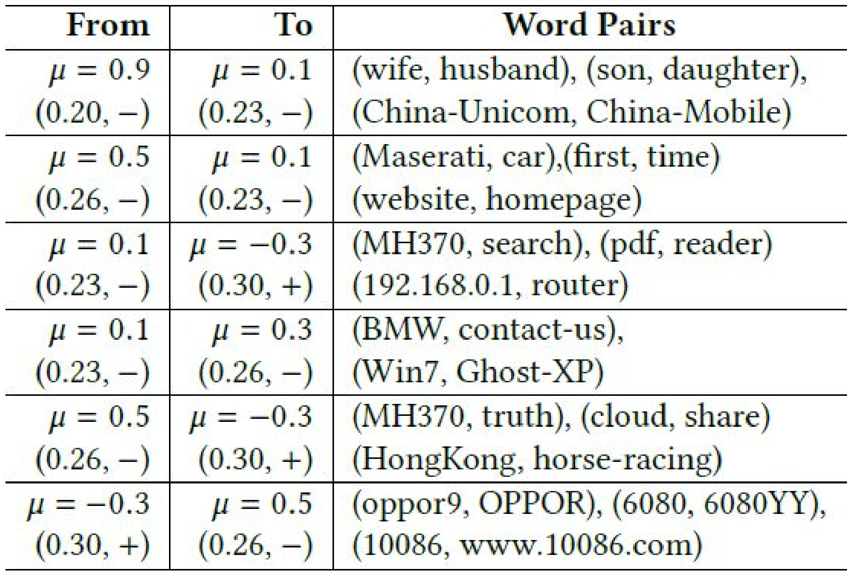
Method	Number of Parameters	Embedding
Lm, BM25	–	–
RankSVM	21	–
Coor-Ascent	21	–
Trans	–	word2vec
DRMM	161	word2vec
CDSSM	10,877,657	–
K-NRM	49,763,110	end-to-end

K-NRM is clearly a lot more complex. However, the performance would not be comparable to DRMM if trained on data of small size since the learning burden is distributed to the ranking and as well as the embeddings, but the model gives space for more improvement as the training data size grows. Compared to DRMM, the embedding of KNRM can capture the training corpus better (This effect would be discussed in soft matching). Moreover, setting the K-NRM hyper-parameters is not trivial. It requires some experiments to find the optimized σ and the coverage setting might also depend on the training data which means the parameters need to be tuned in each query search task.

Soft Matching
Some level of inexact matching is surely desirable. BM25 is an exact match model which is a big weakness. In both of DRMM and K-NRM, different levels of relevance matching are captured through the histogram bins in DRMM and in the differentiable kernels in K-NRM.

Meanwhile, the pooling layer also sizes down the number of parameters to be learned. The structural difference in the embedding layer between the two models leads to very different soft matching performance. According to Table 5 [5], K-NRM can further address the semantic representation nuance in IR scenario to achieve more efficient retrieval. The new retrieval and embedding patterns are:

- Some embedding pairs are pulled away from each other. The scenario could be say If the query already specifies one sub-category, then documents with its category union or another sub-category should be less favored. (first two row of Table 5 [5])
- The embedding learned new soft matches from what two humans tend to connect together which is beyond semantic similarity. (third fourth row of Table 5 [5])
- The embedding learned new soft matches where synonyms are not as important in the IR task setting. (fifth row of Table 5 [5])



From	To	Word Pairs
$\mu = 0.9$ (0.20, −)	$\mu = 0.1$ (0.23, −)	(wife, husband), (son, daughter), (China-Unicom, China-Mobile)
$\mu = 0.5$ (0.26, −)	$\mu = 0.1$ (0.23, −)	(Maserati, car),(first, time) (website, homepage)
$\mu = 0.1$ (0.23, −)	$\mu = -0.3$ (0.30, +)	(MH370, search), (pdf, reader) (192.168.0.1, router)
$\mu = 0.1$ (0.23, −)	$\mu = 0.3$ (0.26, −)	(BMW, contact-us), (Win7, Ghost-XP)
$\mu = 0.5$ (0.26, −)	$\mu = -0.3$ (0.30, +)	(MH370, truth), (cloud, share) (HongKong, horse-racing)
$\mu = -0.3$ (0.30, +)	$\mu = 0.5$ (0.26, −)	(oppor9, OPPOR), (6080, 6080YY), (10086, www.10086.com)

More than Soft Matching

Unlike BM25, both DRMM and K-NRM have a layer for learning to rank so they can more or less reflect user-interpreted matching.

In the term gate network of DRMM, the weights trained with IDF inputs are jointly learned with the ranking network. It suggests that the query-term weights would also be learned from the ranking layer. Especially in top-k ranking, the query term which contains more information should not simply depend on the IDF.

As for K-NRM, it can do another more sophisticated soft-matching. The interactions of K-NRM are learned much more from its ranking layer which reflects more user-interpreted connections instead of being heavily dependent on semantic similarity.

As discussed above, semantic similarity might not always be desirable since it could deviate to another direction for the task. No matter whether the match is exact or inexact, the quality of an IR system should depend on whether it can retrieve what users really care about.

Reference

[1] Christopher D. Manning, Manning Raghavan, & Manning Schütze. 2008. Introduction to Information Retrieval. [chapter pdfs]

[2] Hui Fan, Tao Tao, & ChengZiang Zhai. 2004. A formal study of information retrieval heuristics. Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval (SIGIR '04). [pdf]

[3] Andriy Burkov. The Hundred-Page Machine Learning Book. 2019. [chapter pdfs]

[4] 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 (CIKM '16). [pdf]

[5] 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 on Research and Development in Information Retrieval (SIGIR '17). [pdf]

[6] *C. Xiong, Z. Dai, J. Callan, Z. Liu, and R. Power. End-to-end neural ad-hoc ranking with kernel pooling. In Proceedings of the 40th International ACM SIGIR Conference on Research & Development in Information Retrieval. ACM. 2017.*
https://github.com/AdeDZY/K-NRM

[7] Bhaskar Mitra, Nick Craswell. 2017. Neural Models for Information Retrieval.

https://en.wikipedia.org/wiki/Learning_to_rank

https://en.wikipedia.org/wiki/Okapi_BM25