End-to-End Neural Ad-hoc Ranking with Kernel Pooling

Chenyan Xiong[1]， **Zhuyun Dai[1],**
Jamie Callan[1], Zhiyuan Liu[2], and Russell Power[3]

[1] :Language Technologies Institute, Carnegie Mellon University
[2]:Tsinghua University
[3]:Semantic Scholar, Allen Institute for AI

Motivation

Soft Match

- Soft match between queries and documents has drawn much attention in IR
- Bridges the vocabulary gap: `you-know-who' `Voldemort'

A Recent Success: The Deep Relevance Matching Model (DRMM)

- Word2vec for matching query terms to document terms
- Histogram Pooling to `count' matches of different qualities

Are the current word embeddings the right choice for IR?

- sim(hotel, motel)=0.9, sim(Tokyo, London)=0.9
- Query: `Tokyo hotels'
- Documents: `Tokyo motels' `Hotel in London'

☺ ☹

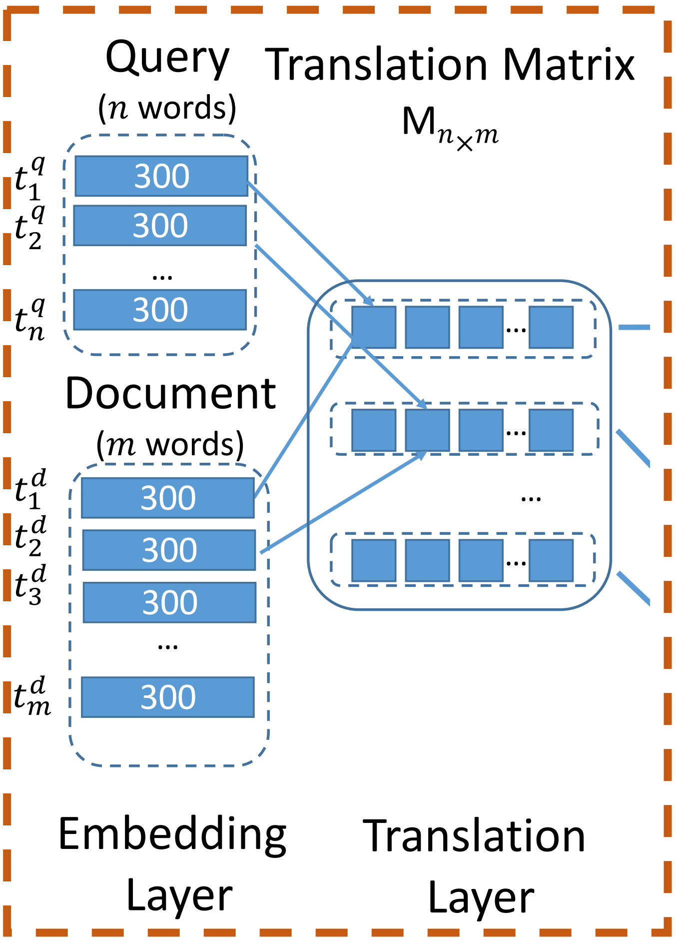
Key Ideas in Our Work

- Build on the ideas in the Deep Relevance Matching Model (DRMM)
- Kernel based Neural Ranking Model (K-NRM)
- Learn a word-similarity metric tailored for matching query and document in ad-hoc ranking.

End-to-end
learning a word
embedding
supervised by
relevance signals.

with a new
kernel pooling layer
that enforces
multi-level
soft match patterns

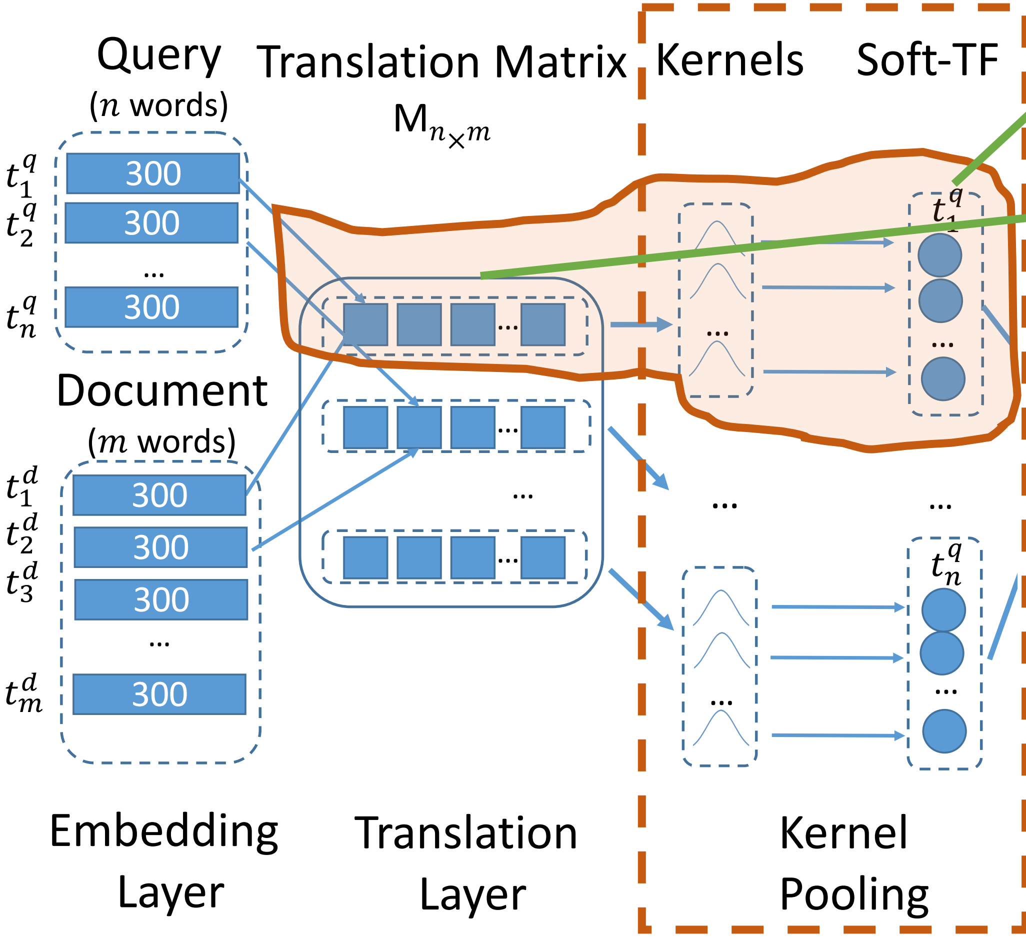
K-NRM Model: Embedding-Based Translation Model

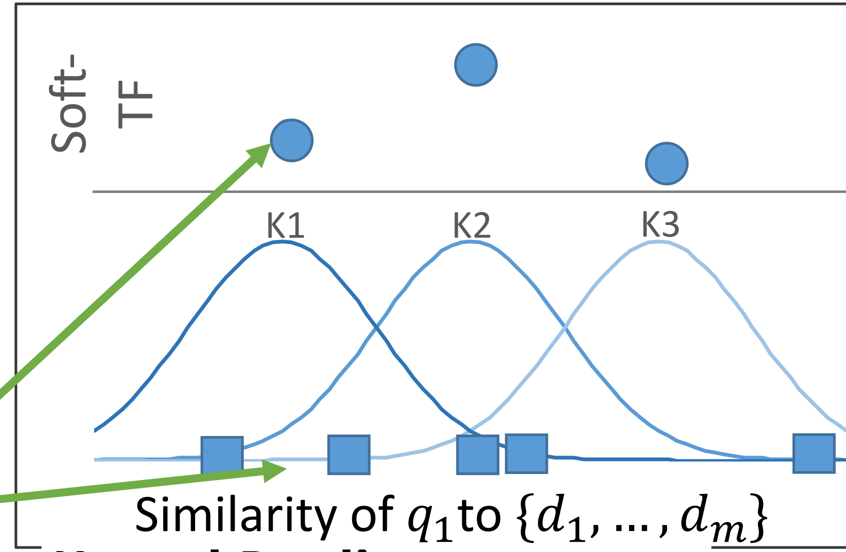


Query-word to document-word translation model.

- Embeds a word into a continuous vector
- Cosine similarities as translation scores

K-NRM Model: Kernel Pooling(1)

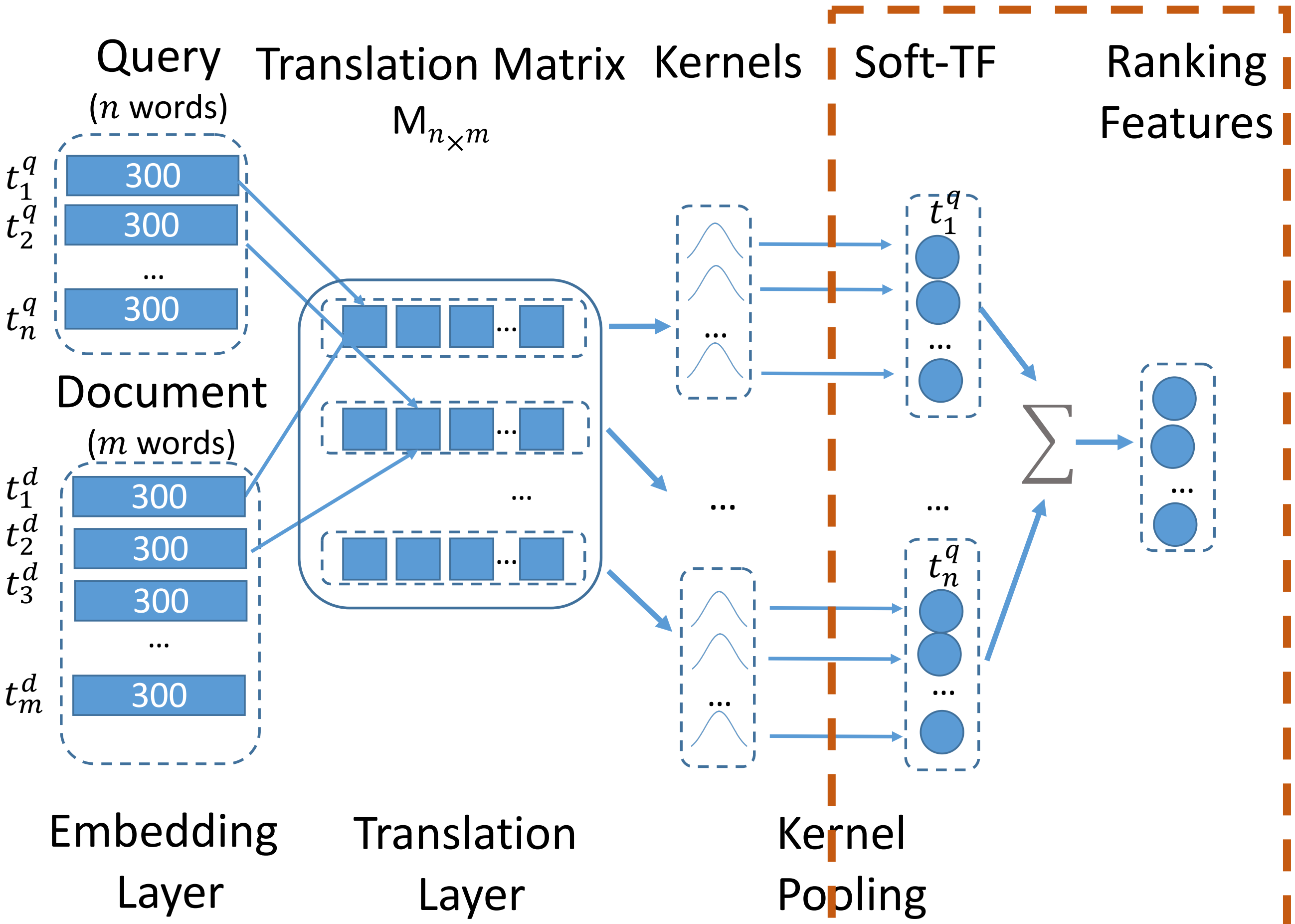




Kernel Pooling

- How many scores are softly in [1, 0.8] ?in [0.8, 0.6]..?
- [1, 0.8]
- RBF Kernel $\mu = 0.9, \sigma = 0.1$
- Soft-TF: softly counting soft-match term frequencies
- $K_k(M_i) = \sum_{j=1}^{m} \exp(-\frac{M_{ij}-\mu_k}{2{\sigma_k}^2})$
- $\vec{K}(M_i) = \{K_1(M_i), \dots, K_K(M_i)\}$
- K soft-TF features for each query-term.

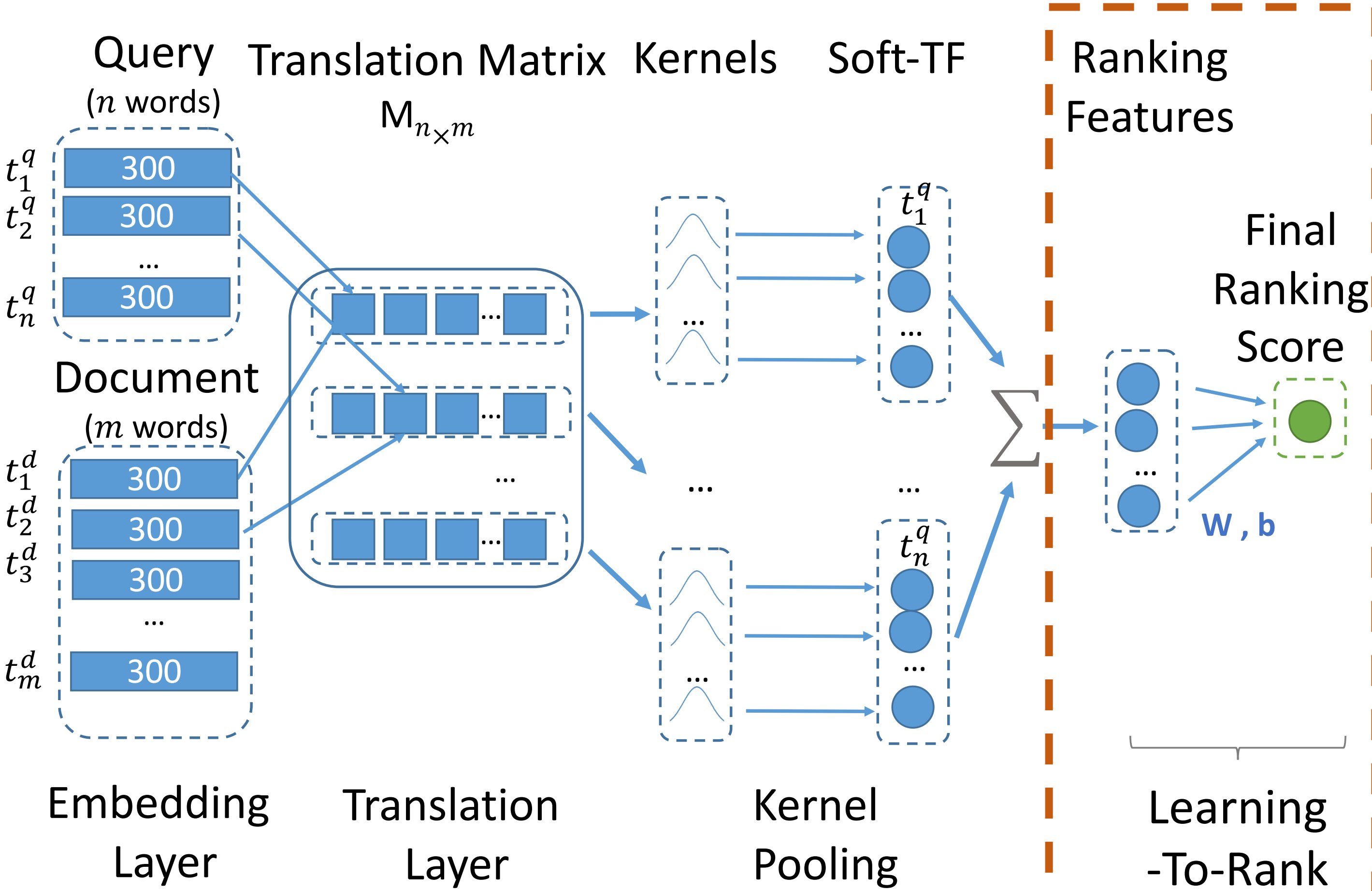
K-NRM Model: Kernel Pooling(2)



Aggregate per-query Soft-TF features with log-sum.

Output: $\phi(M)$, K ranking features for a query document pair

K-NRM Model: Learning-To-Rank



Combine ranking features into the ranking score

- $f(q,d) = \tanh(w \cdot \phi(M) + b)$

Trained in a pairwise manner.

Experimental Methodology: Dataset

- A sample of online search log from **Sogou**, a major Chinese search engine

	Training	Testing
Queries	95,229	1,000
Documents Per Query	12.17	30.50
Search Sessions	31,201,876	4,103,230
Vocabulary Size	165,877	19,079

No Overlap

Experimental Methodology: Training and Testing Labels

We train and test our model using user clicks

(Manual labels not available)

Testing Scenarios:

Testing-SAME	Testing-DIFF	Testing-Raw
Train: DCTR	Train: DCTR	Train: DCTR
Test: DCTR	Test: TACM	Test: Raw Click

DCTR: Document Click Through Rate model [*A.Chuklin et al. 2015*]

TACM: Time-Aware Click Model [*Y.Liu et al. 2016*]

RAW Click: Only the clicked doc is relevant (single click sessions)

Experimental Methodology: K-NRM Model & Baselines

K-NRM - Embedding

- 300-d, initialized with word2vec trained on titles

K-RNM - 11 Kernels

- 1 exact-match kernel: $\mu = 1, \sigma = 0.0001$
- 10 soft-match kernels
  - equally distributed in [-1, 1] (cosine similarity value range)
  - $\mu = -0.9, -0.7, \ldots, 0.7, 0.9, \qquad \sigma = 0.1$

Baselines

1. Unsupervised word-based retrieval: Lm, BM25
2. Word-based LeToR: RankSVM, Coor-Ascent. 20 IR-fusion features.
3. Recent Neural IR methods: Trans, DRMM, CDSSM

Overall Performance: Testing-SAME

Testing-SAME

Train: DCTR

Test: DCTR

- Test and train using the same click model
- Easier task

Method	NDCG@1		NDCG@3		NDCG@10	
Lm	0.1261	−20.89%	0.1648	−26.46%	0.2821	−20.45%
BM25	0.1422	−10.79%	0.1757	−21.60%	0.2868	−10.14%
RankSVM	0.1457	−8.59%	0.1905	−14.99%	0.3087	−12.97%
Coor-Ascent	$0.1594^{\ddagger\S\P}$	–	$0.2241^{\ddagger\S\P}$	–	$0.3547^{\ddagger\S\P}$	–
Trans	0.1347	−15.50%	0.1852	−17.36%	0.3147	−11.28%
DRMM	0.1366	−14.30%	0.1902	−15.13%	0.3150	−11.20%
CDSSM	0.1441	−9.59%	0.2014	−10.13%	$0.3329^{\ddagger\S}$	−6.14%
K-NRM	$\mathbf{0.2642}^{\dagger\ddagger\S\P}$	+65.75%	$\mathbf{0.3210}^{\dagger\ddagger\S\P}$	+43.25%	$\mathbf{0.4277}^{\dagger\ddagger\S\P}$	+20.58%

Overall Performance: Testing-DIFF

Testing-DIFF

Train: DCTR

Test: TACM

- Test with a more accurate click model
- Harder task

Method	NDCG@1		NDCG@3		NDCG@10	
Lm	0.1852	−11.34%	0.1989	−17.23%	0.3270	−13.38%
BM25	0.1631	−21.92%	0.1894	−21.18%	0.3254	−13.81%
RankSVM	0.1700	−18.62%	0.2036	−15.27%	0.3519	−6.78%
Coor-Ascent	$0.2089^{‡¶}$	–	$0.2403^{‡}$	–	$0.3775^{‡¶}$	–
Trans	0.1874	−10.29%	0.2127	−11.50%	0.3454	−8.51%
DRMM	0.2068	−1.00%	$0.2491^{‡}$	+3.67%	$0.3809^{‡¶}$	+0.91%
CDSSM	0.1846	−10.77%	$0.2358^{‡}$	−1.86%	0.3557	−5.79%
K-NRM	$\mathbf{0.2984}^{†‡§¶}$	+42.84%	$\mathbf{0.3092}^{†‡§¶}$	+28.26%	$\mathbf{0.4201}^{†‡§¶}$	+11.28%

Overall Performance: Testing-RAW

Testing-RAW

Train: DCTR

Test: Raw Click

- Only the (only) clicked document is relevant
- The most difficult task
- The clicked document: 1 position higher
- **K-NRM fit the underlying relevance signal rather than the training click model**

Rank = 1/MRR = 4.1

Rank = 1/MRR = 3.0

Method	MRR		W/T/L
Lm	0.2193	−9.19%	416/09/511
BM25	0.2280	−5.57%	456/07/473
RankSVM	0.2241	−7.20%	450/78/473
Coor-Ascent	0.2415‡	–	–/–/–/
Trans	0.2181	−9.67%	406/08/522
DRMM	0.2335‡	−3.29%	419/12/505
CDSSM	0.2321‡	−3.90%	405/11/520
K-NRM	**0.3379**†‡§¶	+39.92%	507/05/424

Sources of Effectiveness #1: IR-Specialized Soft-Match

Embeddings with different levels of IR-specialization

0. exact-match: K-NRM using only exact-match kernel.

1. **Word2vec**: pre-trained on surrounding text
2. **Click2vec**: pre-trained with (query-term, clicked-doc-term)
3. **Full model**: trained end-to-end, pairwise user preference

(3) > (2) > (1) > (0): **more IR-specialized embeddings are more effective.**

	Testing-RAW	
K-NRM Variant	**MRR**	
exact-match	0.2147	–37%
word2vec	$0.2427^{\dagger\P}$	–28%
click2vec	$0.2667^{\dagger\ddagger\P}$	–21%
max-pool	0.2260	–33%
mean-pool	$0.2714^{\dagger\ddagger\P}$	–20%
full model	$\mathbf{0.3379}^{\dagger\ddagger\S\P*}$	–

* Testing-SAME and Testing-DIFF had similar performance.

Sources of Effectiveness #2: Multi-Level Soft-Match

Three different pooling methods. All end-to-end

1. **Max-pool**: use the best-match
2. **Mean-pool**: all soft-match scores are mixed together
3. **Full model**: kernel pooling enforces multi-level soft match

(3) > (2) > (1): multi-level soft-match provides more information

	Testing-RAW	
K-NRM Variant	**MRR**	
exact-match	0.2147	−37%
word2vec	0.2427†¶	−28%
click2vec	0.2667†‡¶	−21%
max-pool	0.2260	−33%
mean-pool	0.2714†‡¶	−20%
full model	**0.3379†‡§¶***	–

* Testing-SAME and Testing-DIFF had similar performance.

Effects on Word Embeddings

Recall our motivation:

“ Content-based word embedding may not fit ad-hoc search ”

- This proves true: **58%** of word2vec word pairs were moved across kernels by K-NRM.

- How does K-NRM move word pairs?

Effects on Word Embeddings:

Word Pair Movements

1. Word pairs decoupled.
   - considered related by word2vec, but not by K-NRM
   - **> 90%** are decoupled!
2. New soft match discovered.
   - less frequently appear in the same surrounding context, but convey similar search intent
3. Matching strength level
   - Strong <-> Weak

Decouple

(wife, husband), (son, daughter),
(China-Unicom, China-Mobile),
(Maserati, car),
(website, homepage)

New soft match

(MH370, search), (pdf, reader),
(BMW, contact-us),
(192.168.0.1, router),

Change Levels

Weak->Strong

(MH370, truth), (cloud, share)

Strong->Weak

(oppor9, OPPOR),
(10086, www.10086.com)

Conclusion

- Embeddings trained for search are more effective than embeddings trained from surrounding text
- Embeddings and soft-match bins must be tuned together.
- We propose a new kernel pooling technique:
  - Allows end-to-end training of the word embedding
  - Guides the embedding to form effective multi-level soft-match patterns tailored for ad-hoc ranking
- **Delivers robust soft match between query and documents**
  - Moved 58% of word2vec word pairs across kernels
  - Decouples > 90% of word pairs that were considered related by word2vec
  - Discovers new soft match patterns of different types

Thank you!

Questions?

Testing labels

- Cut-off threshold chosen to make sure the label distribution the same as TREC's

Condition	Label	Label Distribution
Testing-SAME	DCTR Scores	70%, 19.6%, 9.8%, 1.3%,1.1%
Testing-DIFF	TACM Scores	79%, 14.7%, 4.6%, 0.9%, 0.9%
Testing-RAW	Raw Clicks	2,349,280 clicks

Performance of tail queries

Table 9: Ranking accuracy on Tail (frequency < 50), Torso (frequency $50 - 1K$) and Head (frequency $> 1K$) queries. † indicates statistically significant improvements of K-NRM over Coor-Ascent on Testing-RAW. Frac is the fraction of the corresponding queries in the search traffic. Cov is the fraction of testing query words covered by the training data.

	Frac	Cov	Testing-RAW, MRR		
			Coor-Ascent	K-NRM	
Tail	52%	85%	0.2977	0.3230†	+8.49%
Torso	20%	91%	0.3202	0.3768†	+17.68%
Head	28%	99%	0.2415	0.3379†	+39.92%

Requirement of training data

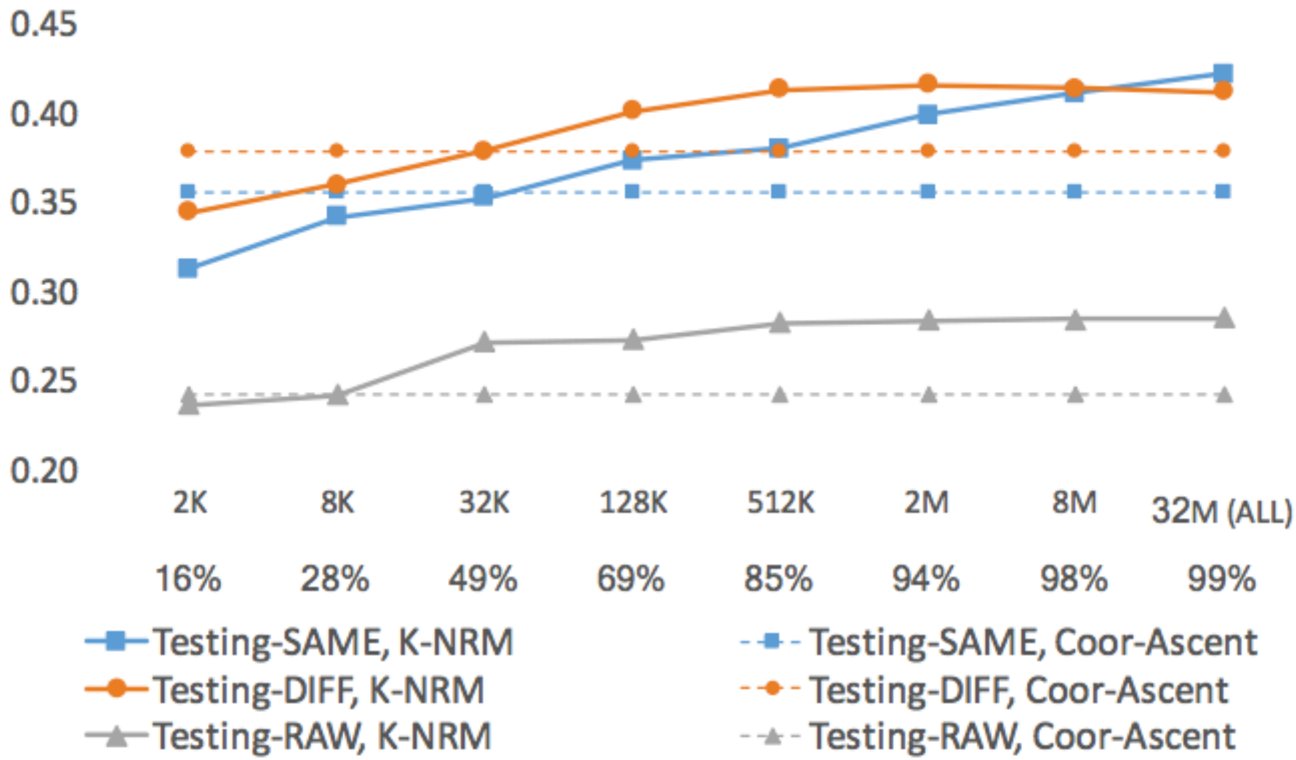


Figure 4: K-NRM's performances with different amounts of training data. X-axis: Number of sessions used for training, and the percentages of testing vocabulary covered (second row). Y-axis: NDCG@10 for Testing-SAME and Testing-DIFF, and MRR for Testing-RAW.

Sensitivity to kernel parameters

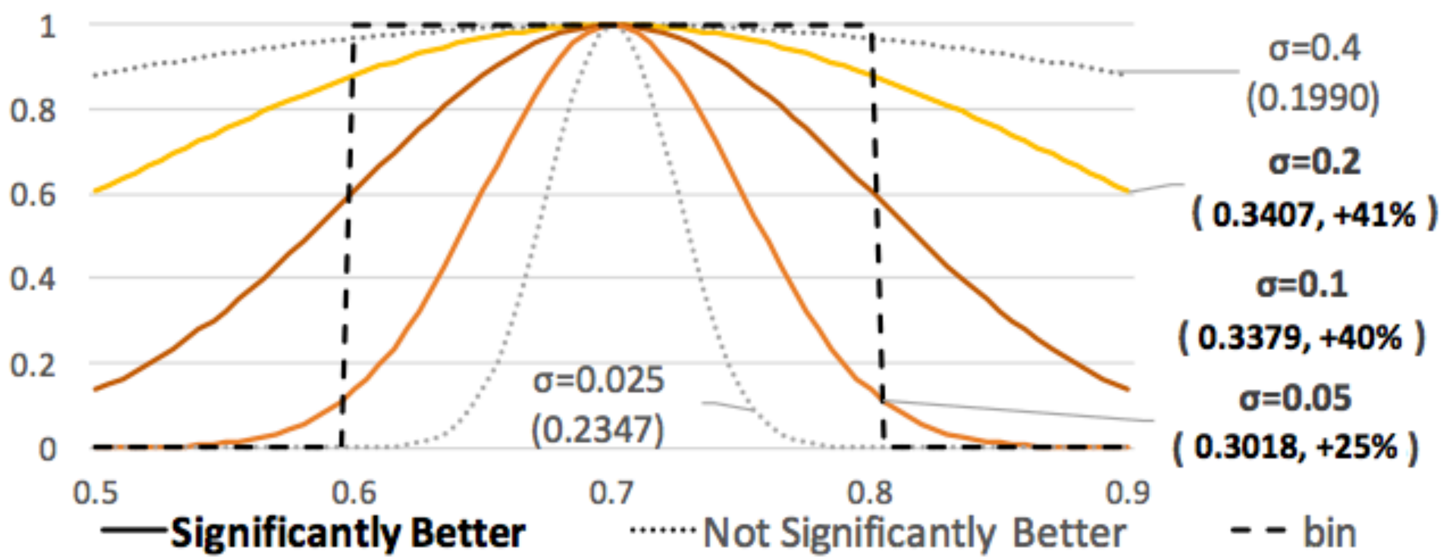
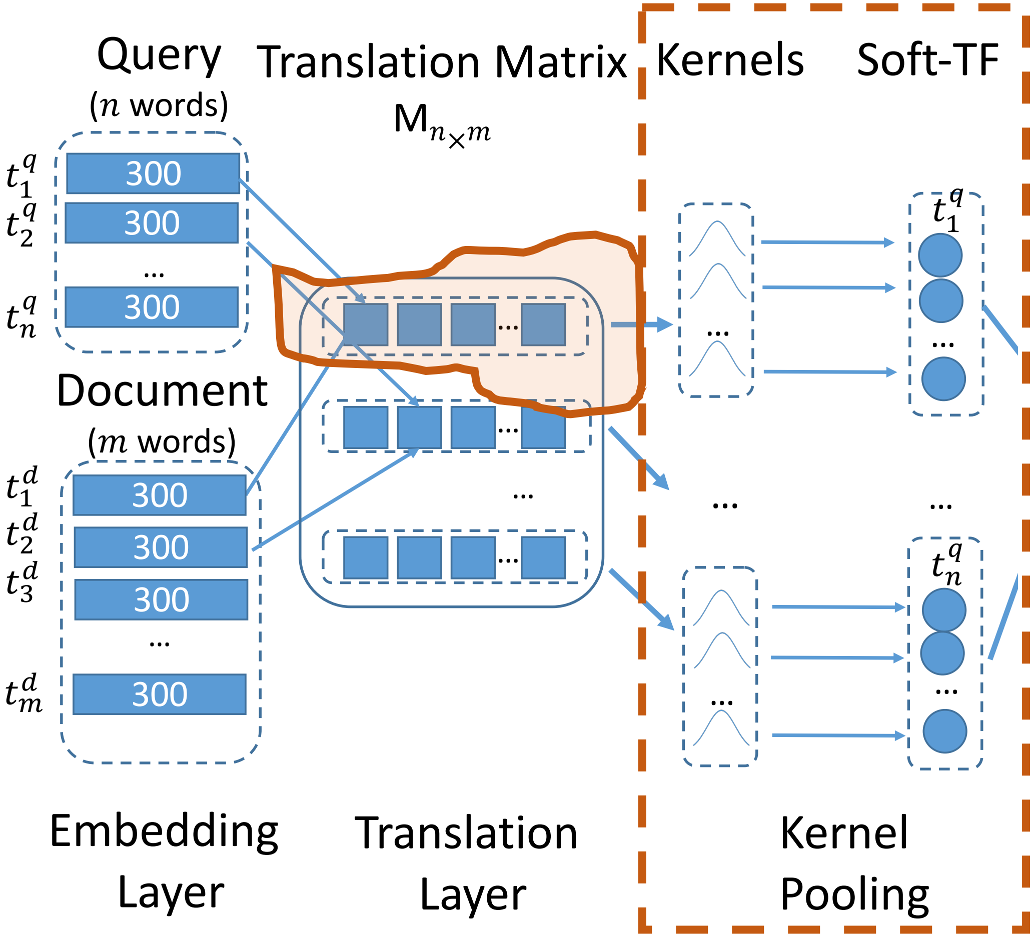


Figure 5: K-NRM's performance with different σ. MRR and relative gains over Coor-Ascent are shown in parenthesis. Kernels drawn in solid lines indicate statistically significant improvements over Coor-Ascent.

K-NRM Model:
Kernel Pooling: revisit Histogram Pooling



DRMM: Histogram Pooling

- How many word pairs' similarity score are in [1, 0.8], [0.8, 0.6]...?
- $B_k(M_i) = \sum_{j=1}^{m} \mathrm{I}\{M_{ij} \text{ in bin k}\}$
- $\vec{B}(M_i) = \{B_1(M_i), \ldots, B_K(M_i)\}$