11-442 / 11-642 / 11-742:
Search Engines

Learning to Rank: Neural Models

Jamie Callan
Carnegie Mellon University
callan@cs.cmu.edu

1

Introduction

Neural / deep learning ranking has become popular again

- Also studied in the 1990's (Kwok, 1995; Caid et al., 1995; …)
- Many recent successes in other language technologies

How are neural models different from what we have seen so far?

- Fewer hand-crafted features and functions
- More complex methods for combining evidence / weights
- Many more parameters

Do they work for ad-hoc search?

- Systems from the 1990's were never best (but, also not terrible)
- Currently, they are better than feature-based learning-to-rank

2

2

Introduction: Overview of Neural Ranking Models

We cover several recent neural methods of ad-hoc retrieval

- DSSM — Representation-based model
- DRMM, KMRM, Conv-KNRM — Interaction-based models
- BERT reranking — Interaction-based models

Our goals

- Learn about newer work on ad-hoc retrieval
- Identify general themes in neural ranking research
- Identify similarities and differences with older models

3

3

Outline

Introduction

Deep Structured Semantic Models (DSSM)

Deep Relevance Matching Model (DRMM)

Kernel-based Neural Ranking Model (K-NRM)

Convolutional Kernel-based Neural Ranking Model (Conv-KNRM)

BERT reranking

DeepCT

doc2query

4

4

Deep Structured Semantic Models (DSSM): Motivating Ideas

Often there is a vocabulary mismatch between the query and matching documents

- E.g., query is ‘cat’, but document contains ‘kittens’
- Traditional retrieval models don’t handle vocabulary gap well
  - One solution: Fix the query (e.g., pseudo relevance feedback)

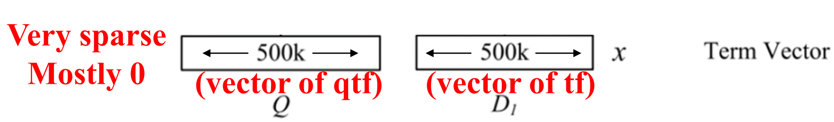
DSSM addresses the vocabulary mismatch

- Map text (e.g., q or d) to a low-dimensional latent space
  - Word-based → concept-based (hopefully)
- Match query and document in the latent space
  - The matching process won’t be sensitive to vocabulary choices

Deep Structured Semantic Models (DSSM): Architecture

DSSM uses a vocabulary of 500K terms (ignore all other terms)

- Reasonable for Web search



(Huang, et al., 2013)

Deep Structured Semantic Models (DSSM): Architecture

Map each word to a vector ('hashing')

- Add delimiters, then break the word into trigrams
  - 'deep' → '#deep#' → '#de', 'dee', 'eep', 'ep#'
  - 'deeper' → '#deeper#' → '#de', 'dee', 'eep', 'epe', 'per', 'er#'
- Each word is represented by a vector of trigrams

	..	#de	..	dee	..	eep	..	epe	..	ep#	..	er#	..	per	..
deep	..	1	..	1	..	1	..	0	..	1	..	0	..	0	0
deeper	..	1	..	1	..	1	..	1	..	0	..	1	..	1	0

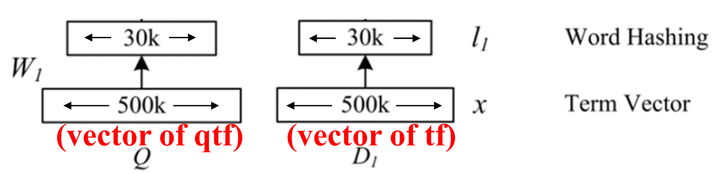
  - Vectors represent lexical or orthographic similarity
- 500K term vocabulary → 30K trigram vocabulary
- Low collision rate (e.g., 22 out of 500k terms)
- Robust to out-of-vocabulary problems

(Huang, et al., 2013)

Deep Structured Semantic Models (DSSM): Architecture

Map each word to a vector ('hashing')

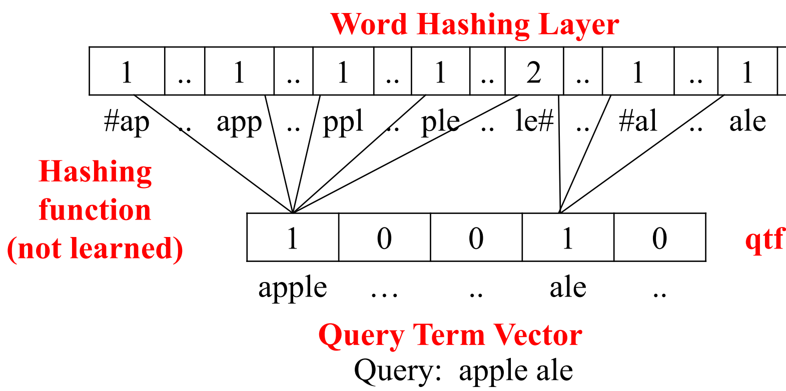
- Add delimiters, then break the word into trigrams
  - 'deep' → '#deep#' → '#de', 'dee', 'eep', 'ep#'
  - 'deeper' → '#deeper#' → '#de', 'dee', 'eep', 'epe', 'per', 'er#'
- Each word is represented by a vector of trigrams



(Huang, et al., 2013)

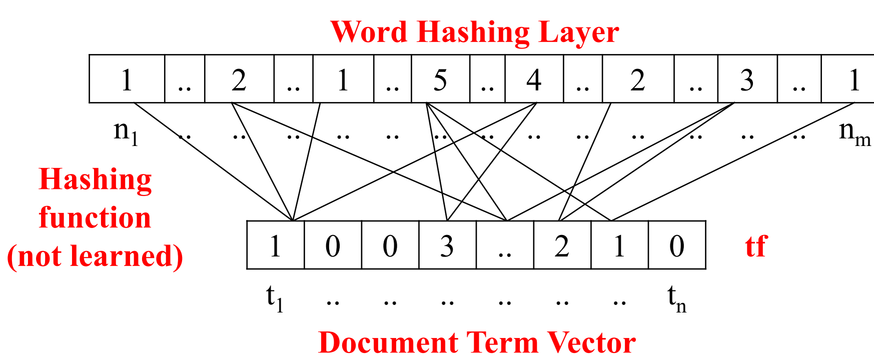
Deep Structured Semantic Models (DSSM): Architecture

Word hashing layer width: 30K ngrams (ngram vocabulary)
Text vector length: 500K terms (term vocabulary)



Deep Structured Semantic Models (DSSM): Architecture

Word hashing layer width: 30K ngrams (ngram vocabulary)
Text vector length: 500K terms (term vocabulary)



Deep Structured Semantic Models (DSSM): Architecture

The hashed representation captures <u>orthographic</u> similarity

- Spelling
- Case
- Hyphenation
- …
- Similar to case conversion, stemming, etc

It <u>does not</u> capture conceptual similarity

- E.g., cats and kittens

trigrams	cat	cats	bat	bats
#ba	0	0	1	1
bal	0	0	0	0
all	0	0	0	0
al#	0	0	0	0
bat	0	0	1	1
#ca	1	1	0	0
cat	1	1	0	0
at#	1	0	1	0
ats	0	1	0	1
ts#	0	1	0	1

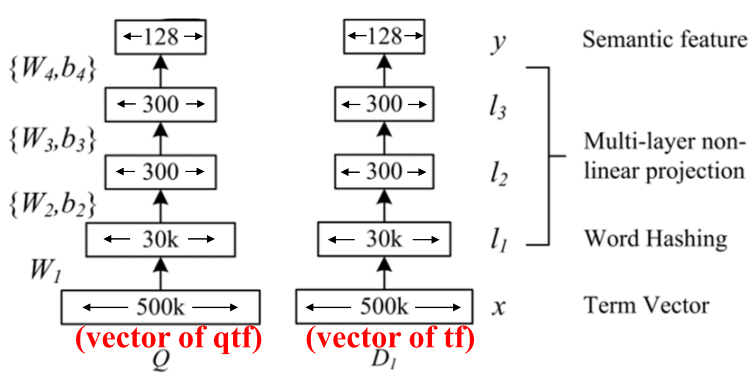
Deep Structured Semantic Models (DSSM): Architecture

Use a three-layer feed-forward neural network to create "semantic features" for each text

- Reasons for 300 and 128 dimensions are not explained

$l_i = f(W_i l_{i-1} + b_i)$

$l_1 = W_1 x$



(Huang, et al., 2013)

Deep Structured Semantic Models (DSSM): Hidden Layers

Given input vector x and output vector y, layer i's weights are:

$$l_1 = W_1 x$$

$$l_i = f(W_i l_{i-1} + b_i), i = 2, \ldots, N-1$$

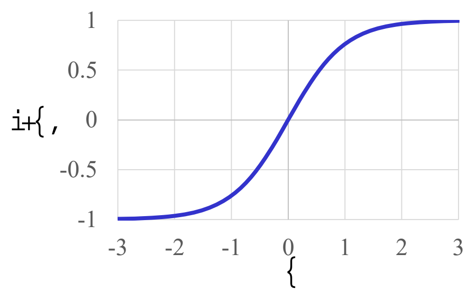
$$y = f(W_N l_{N-1} + b_N)$$

The word hashing layer

Non-linear projection layers

The activation function is tanh

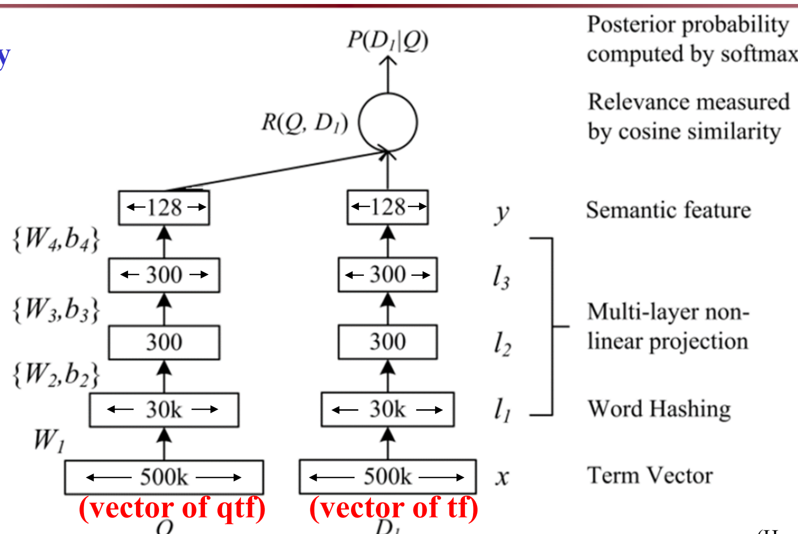
$$f(x) = \frac{1 - e^{-2x}}{1 + e^{-2x}}$$



(Huang, et al., 2013)

Deep Structured Semantic Models (DSSM): Architecture

Cosine similarity



(Huang, et al., 2013)

Deep Structured Semantic Models (DSSM): Type of Neural IR Model

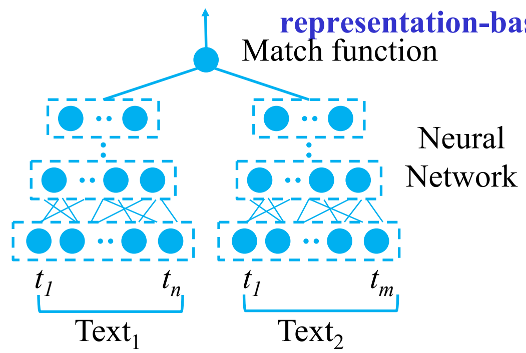
DSSM is a type of representation-based neural IR model

- Build an abstract representation $\Phi(T)$ for each text T
- Match the abstract representations for T_1 and T_2

There are many models

- E.g., DSSM, C-DSSM
- E.g., ARC-I, ARC-II

representation-based



Deep Structured Semantic Models (DSSM): Training

Trained using a query log

- T_1: query
- T_2: document title
- Clicked documents were treated as relevant
- Non-relevant documents were selected "randomly"
  - Randomly from the collection would be unrealistic
  - Randomly from the top N (e.g., N=20 or N=100) would be fine

(Huang, et al., 2013)

Deep Structured Semantic Models (DSSM): Training

Query: apple pie

Top titles with clicks (✓)

1. Perfect apple pie recipe | Pillsbury
2. Apple pie recipe | Taste of home
3. Apple pie by Grandma Ople
4. Scrumptious apple pie recipe
5. Apple pie recipe | Food network
6. ✓ Apple pie recipe | NYT cooking
7. ✓ Apple pie – Martha Stewart
8. Apple pie – Wikipedia
9. …

Training data

- p (T_6 | q) > p (T_1 | q)
- p (T_6 | q) > p (T_2 | q)
- p (T_6 | q) > p (T_3 | q)
- …
- p (T_7 | q) > p (T_1 | q)
- p (T_7 | q) > p (T_2 | q)
- p (T_7 | q) > p (T_3 | q)
- …

Pairwise training

Deep Structured Semantic Models (DSSM): Training

"Trained using gradient-based numerical optimization"

- Minimize the loss function

$$L(\Lambda) = -\log \prod_{(Q,D^+)} P(D^+|Q) \qquad P(D|Q) = \frac{\exp(\gamma R(Q,D))}{\sum_{D' \in \boldsymbol{D}} \exp(\gamma R(Q,D'))}$$

(Maximize scores of clicked docs, ignore unclicked docs)

The learning algorithm is not our focus

- Assume that it uses many preference pairs…

  p (T_6 | q) > p (T_1 | q)

  to learn weights that give higher scores to clicked documents

(Huang, et al., 2013)

Deep Structured Semantic Models (DSSM): Testing

Methodology

- 16,510 English queries from a commercial search engine
- Re-rank top 15 documents/query from another ranker
- Human relevance assessments on a scale of 0-4
- 2-fold cross-validation
- Metrics: NDCG @1, @3, @10

Experimental results

- 10-15% better than BM25 and several unsupervised models
  - Not surprising: Supervised is expected to beat unsupervised
- In our experiments, SVM-Rank > DSSM > BM25

(Huang, et al., 2013)

19

19

Deep Structured Semantic Models (DSSM): Summary

Key ideas

- Orthographic → continuous term representation (unusual)
- No idf or document length
  - Perhaps not needed to re-rank the top 15 from a strong ranker
  - Perhaps not needed to re-rank titles (short,)

Why does it work?

- Unclear from their experiments and subsequent work
- They think it captures semantic structure, but don't say how
- **Note:** Just re-ranking the top 15 titles produced by another ranker
  - Could be learning site preferences, or …?

20

20

Outline

Introduction

Deep Structured Semantic Models (DSSM)

Deep Relevance Matching Model (DRMM)

Kernel-based Neural Ranking Model (K-NRM)

Convolutional Kernel-based Neural Ranking Model (Conv-KNRM)

BERT reranking

DeepCT

doc2query

Deep Relevance Matching Model (DRMM): Motivating Ideas

Much recent deep learning research uses word embeddings

- Represent a term by a weight vector (continuous representation)

Continuous representations are an old idea in IR

- LSI, LSA, PLSA, PIRCS, MatchPlus, …
  - Not terrible, but not as good as BM25, vector space, …
- Query term 'cat' matches document term 'kitten' ☺
- Query term 'cat' matches document term 'dog' ☹

Query & document terms that match exactly are a strong signal

- Prior work with continuous representations lost this signal

(Guo, et al., 2016)

Word2Vec

Word2vec is a popular method for creating continuous representations of terms

- Input: A lot of text
- Output: A vector-based term dictionary
  - Words that appear in similar contexts will have similar term vectors

cat	kitten
0.14	0.13
0.01	0.02
0.00	0.01
0.38	0.35
0.01	0.00
0.00	0.01
0.27	0.29
: :	: :
0.67	0.60
←300→	←300→

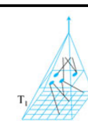
Examples of similar terms (English GoogleNews)

- apple: apples, pear, fruit, berry, pears, strawberry
- pie: pies, cake, slice, cheesecake, biscuit
- man: woman, boy, teenager, girl, robber, men
- cat: cats, dog, kitten, feline, beagle, puppy

23

23

Deep Relevance Matching Model (DRMM)



Key ideas

- Continuous representations of terms (word2vec)
- Measure the interaction between each pair of terms (q_i, d_j)
- For each query term q_i, bin interactions of different strengths
- Use a feed-forward network to combine signals for q_i
- Aggregate scores for q_i
- Modulate the influence of q_i ("gating")
- Linear combination to produce a score for (q_i, d_j)

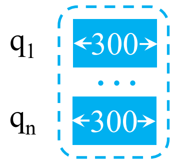
It's simpler than it sounds…

(Guo, et al., 2016)

24

24

Deep Relevance Matching Model (DRMM): Query Representation



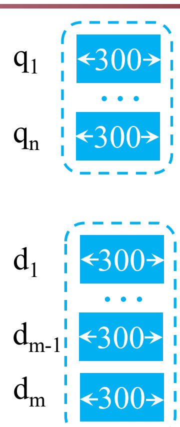
Embedding Layer

Use a continuous representation of query terms

- A 300-dimension vector for each term
- Standard word2vec

(Guo, et al., 2016)

Deep Relevance Matching Model (DRMM): Document Representation



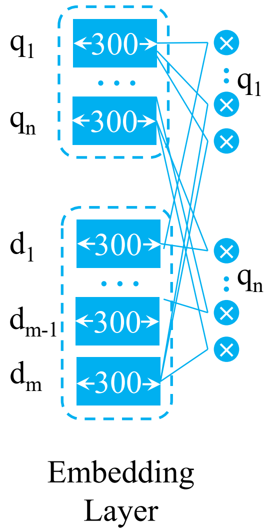
Embedding Layer

Use a continuous representation of document terms

- A 300-dimension vector for each term
- Standard word2vec

(Guo, et al., 2016)

Deep Relevance Matching Model (DRMM): Local Interactions



Compare each query term to each document term

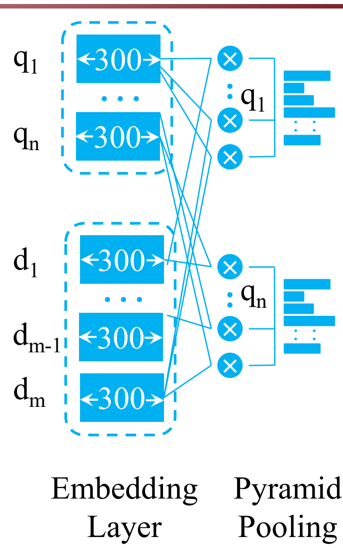
- Cosine similarity of 300-dimension embedding vectors for (q_i, d_j)
- Values are in range [-1, 1]

Note: This is an interaction model

- It considers many local interactions between q and d

(Guo, et al., 2016)

Deep Relevance Matching Model (DRMM): Pyramid (Histogram) Pooling



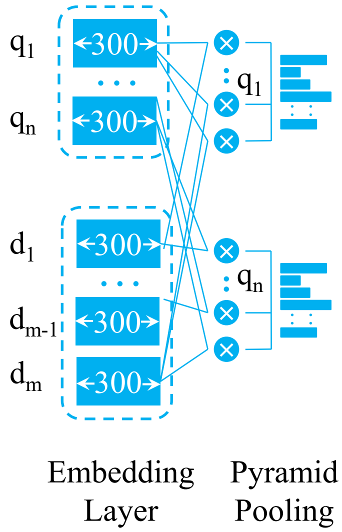
Bin values for (q_i, d_j) matches of different quality

- 1 bin for [1,1]
  - q_i and d_j match exactly
- b bins for [-1, 1)
  - q_i and d_j match softly
  - E.g., [-1, -0.8) … [0.8, 1.0)

How should values be binned?

(Guo, et al., 2016)

Deep Relevance Matching Model (DRMM): Pyramid (Histogram) Pooling

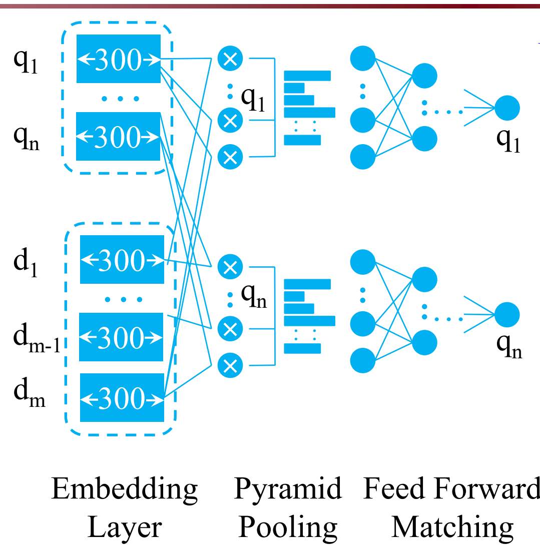


They tried 3 types of histograms

- Count matches in range (CH)
  – Number of matches to q_i in each range (e.g., [0.2, 0.4))
  – Essentially tf for each range
- Normalized count (NH)
  – Percentage of matches to q_i in each quality range
- Log of count (LCH)
  – log (tf) for each range (most effective method)

(Guo, et al., 2016)

Deep Relevance Matching Model (DRMM): Feed Forward Neural Network



Use a feedforward network to combine the scores from the 11 bins for q_i into a match score

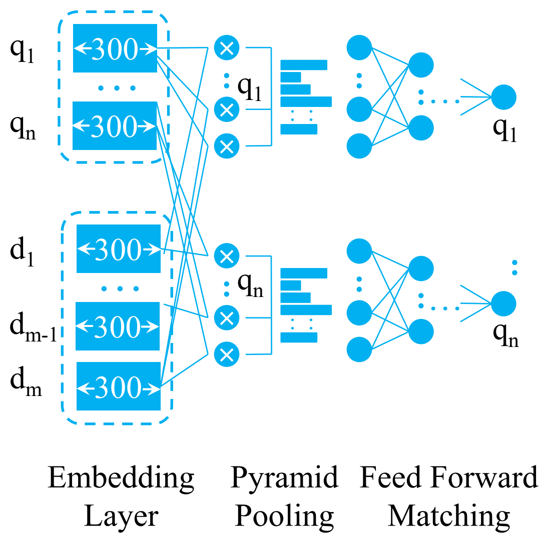
- 2 hidden layers

$$z_i^{(l)} = tanh(\boldsymbol{W}^{(l)} \boldsymbol{z}_i^{(l-1)} + \boldsymbol{b}^{(l)})$$

$i=1, \ldots, n, l=1, \ldots, L$

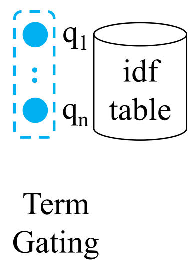
(Guo, et al., 2016)

Deep Relevance Matching Model (DRMM): Term Gating



Model term importance with gate weights

$$g_i = \frac{\exp\left(w\,\mathrm{idf}(q_i)\right)}{\sum_{j=1}^{n} \exp\left(w\,\mathrm{idf}(q_j)\right)}$$

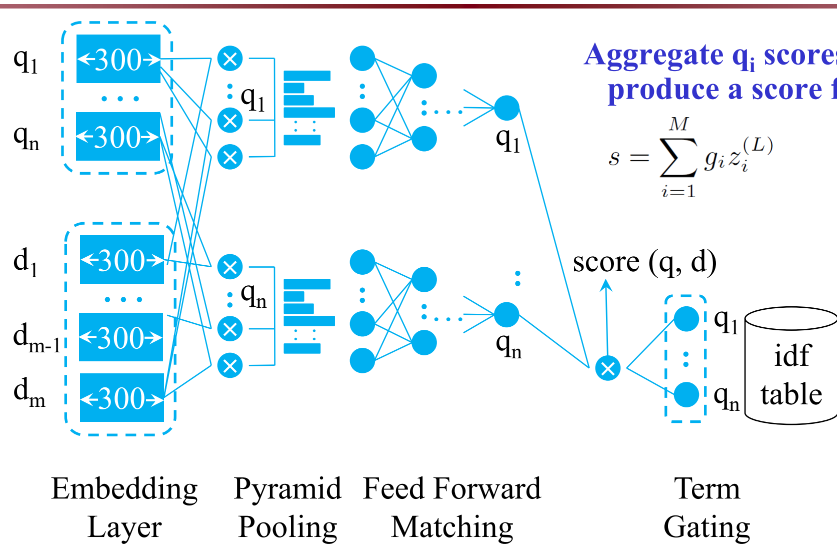


(Guo, et al., 2016)

Deep Relevance Matching Model (DRMM): Aggregation

Aggregate q_i scores to produce a score for q

$$s = \sum_{i=1}^{M} g_i z_i^{(L)}$$



(Guo, et al., 2016)

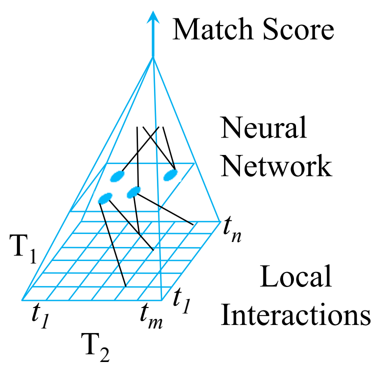
Deep Relevance Matching Model (DRMM): Type of Neural IR Model

DRMM is a type of interaction-based neural IR model

- Identify local matches between two pieces of text
  - – E.g., cosine similarity of term vectors
- Learn interaction patterns for matching
  - – Often hierarchical patterns
  - – E.g., convolutional neural network

There are many interaction-based models

- DRMM, DeepMatch, ARC-II
- MatchPyramid, K-NRM



33

33

Deep Relevance Matching Model (DRMM): Computational Complexity

Every query matches every document

- There are always soft-matches
- The computational cost is too high to be practical for initial retrieval

DRMM is used in a re-ranking pipeline

- Use an efficient algorithm (e.g., Indri) to create a ranking
- Use DRMM to re-rank the top *n* documents

34

34

Deep Relevance Matching Model (DRMM): Training

Pairwise training with hinge loss

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

d^+: Relevant documents

d^-: Non-relevant documents

Training data

- Robust04: 600K documents, 50 queries
- ClueWeb09-B: 34M documents, 150 queries

(Guo, et al., 2016)

Deep Relevance Matching Model (DRMM): Effectiveness

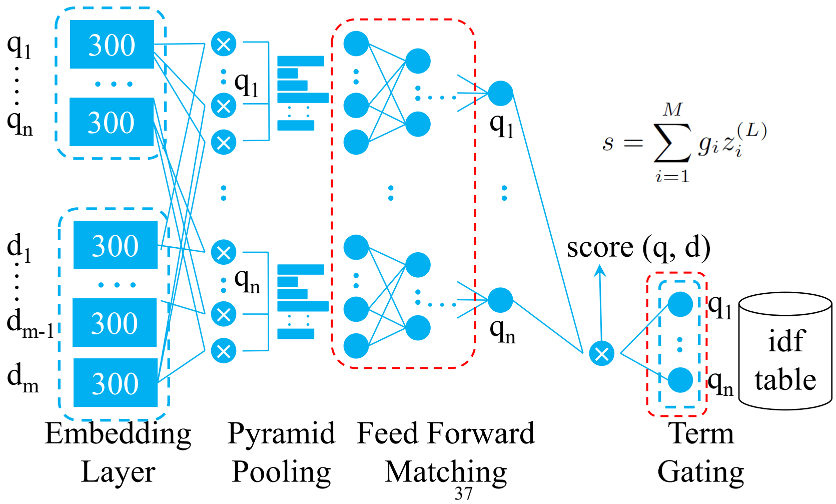
DRMM is more effective than Indri and BM25

- Supervised vs unsupervised … not surprising

Guo, et al. didn't compare to learning-to-rank systems (!)
… but we did

- DRMM is a little better than Rank-SVM
  - Should it be compared to a system that does query expansion?
- DRMM is about the same as Coordinate Ascent
  - A good list-wise LeToR algorithm

Deep Relevance Matching Model (DRMM): Where Does the Learning Occur?



(Guo, et al., 2016)

37

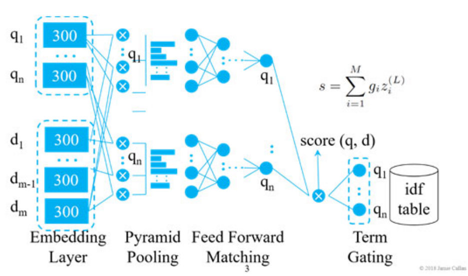
Deep Relevance Matching Model (DRMM): Where Does the Learning Occur?

DRMM learns how to combine evidence

- How to combine 'exact match', 'strong match' and 'weak match' signals

The word embeddings are static

- Learning cannot propagate weights through the histogram layer



38

(Guo, et al., 2016)

Deep Relevance Matching Model (DRMM): Similarities and Differences

Similarity to older models

- A bag-of-words model
- Exact-match of query terms to document terms
- log (tf)
- Idf
- Summation of scores for each query term

Differences with older models

- Exact- <u>and</u> soft-match of query terms and document terms
  - Continuous representations
- Binning for matches of different quality
  - A bin for exact matches
  - Bins for 'close' and 'far' matches
- Non-linear combination of match values of different quality

39

Outline

Introduction
Deep Structured Semantic Models (DSSM)
Deep Relevance Matching Model (DRMM)
Kernel-based Neural Ranking Model (K-NRM)
Convolutional Kernel-based Neural Ranking Model (Conv-KNRM)
BERT reranking
DeepCT
doc2query

40

For More Information

- W. R. Caid, S. T. Dumais, and S. I. Gallant. Learned vector-space models for document retrieval. Information Processing and Management, 31(3), pp. 419-429, 1995.
- J. Guo, Y. Fan, Q. Ai, and W. B. Croft. A deep relevance matching model for ad-hoc retrieval. CIKM 2017.
- B. Hu, Z. Lu, H. Li, and Q. Chen. Convolutional neural network architectures for matching natural language sentences. NIPS 2014.
- P.-S. Huang, X. He, J. Gao, L. Deng, A. Acero, and L. Heck. Learning deep structured semantic models for web search using clickthrough data. CIKM 2013.
- K. L. Kwok. A network approach to probabilistic information retrieval. ACM Transactions on Information Systems, 13(3), pp 324-353, 1995.
- L. Pang, Y. Lan, J. Guo, J. Xu, and X. Cheng. A deep investigation of deep IR models. SIGIR 2017 Neu-IR workshop.
- Y. Shen, X. He, J. Gao, L. Deng, and G. Mesnil. Learning semantic representations using convolutional neural networks for web search. WWW 2014.
- C. Xiong, Z. Dai, J. Callan, Z. Liu, and R. Power. End-to-end neural ad-hoc ranking with kernel pooling. SIGIR 2017.