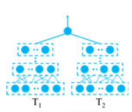
11-442 / 11-642 / 11-742:
Search Engines

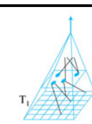
Learning to Rank: Neural Models

Jamie Callan and Zhuyun Dai
Carnegie Mellon University
callan@cs.cmu.edu

1



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)**
- Re-ranking with BERT
- DeepCT
- doc2query
- Summary

2

Conv-KNRM: Motivating Ideas

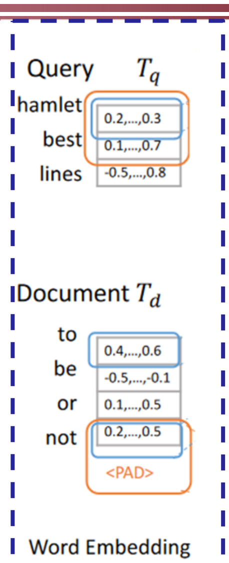
Borrow as many good ideas as possible from K-NRM

Extend the model to support n-grams

- Compose unigram embeddings to create n-gram embeddings
  - ‘deep’ + ‘learning’ vs. deep_learning
- Allow matching between n-grams of different lengths
  - E.g., ‘deep learning’ and ‘convolutional neural network’)

(Dai, et al., 2018)

Conv-KNRM: Text Representation

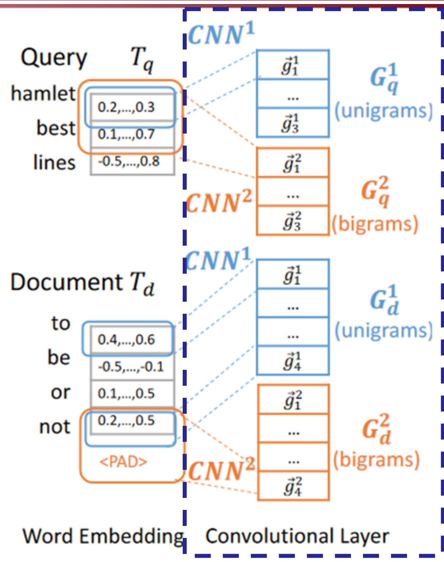


Represent query and document terms with embeddings

- 300 dimensions, as in DRMM and K-NRM
- Initialize with word2vec

(Dai, et al., 2018)

Conv-KNRM:
Text Representation

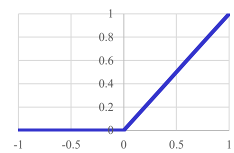


To form one n-gram of length *h* use a convolutional filter to combine *h* term embeddings

- E.g., for terms $T_i, T_{i+1}, \ldots, T_{i+h}$
- $v = w_v \cdot T_{i:i+h} \qquad v \in \mathbb{R}$
- F filters produce a vector V
  - F different combinations

The n-gram embedding of $T_{i:i+h}$

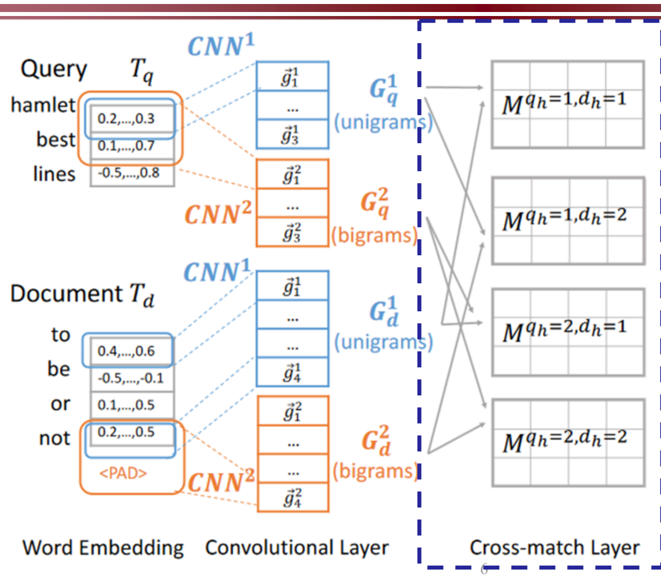
- $g_i^h = relu\left(w^h \cdot V^h + b^h\right)$
- $relu\ (\mathbf{x}) = \max\ (\mathbf{0}, \mathbf{x})$
  - "rectifier linear unit"



(Dai, et al., 2018)

5

Conv-KNRM:
Local Interactions



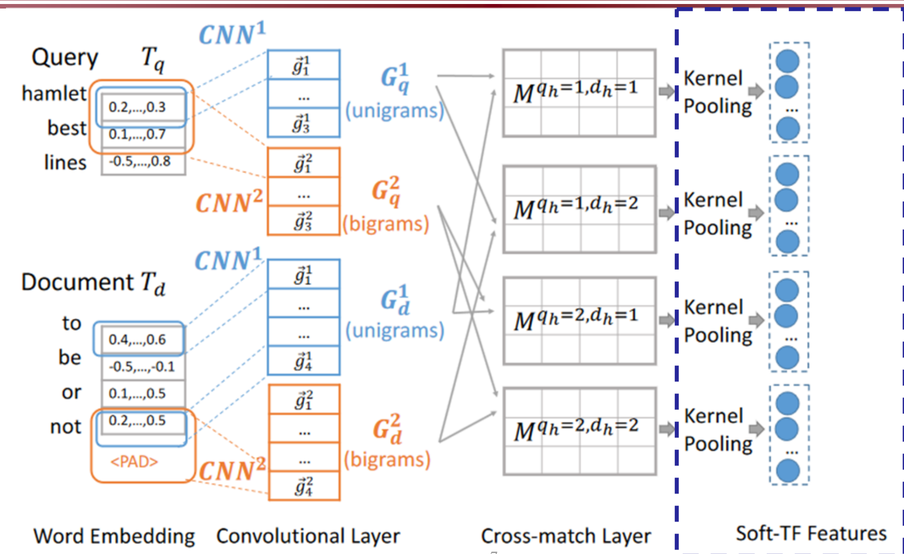
A cross-match layer allows n-grams of different lengths to match

- Similar to K-NRM's translation matrix
- 'cat movies' can match 'silly kitten videos'

(Dai, et al., 2018)

6

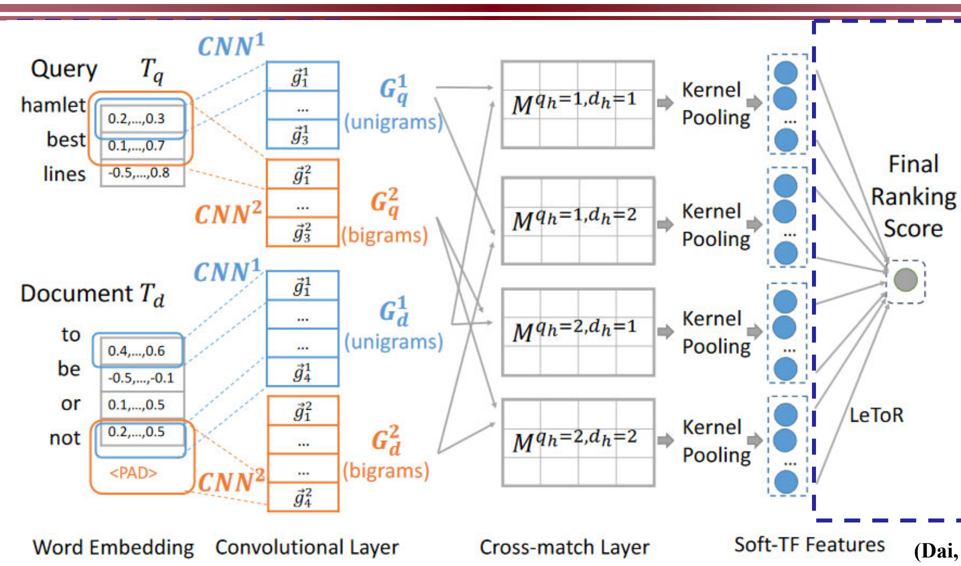
Conv-KNRM: Pooling



(Dai, et al., 2018)

7

Conv-KNRM: Aggregation



(Dai, et al., 2018)

8

Conv-KNRM

Conv-KNRM is a type of interaction-based neural IR model

- Identify <u>local matches</u> between two pieces of text
- Learn interaction patterns for matching

Conv-KNRM is used in a re-ranking pipeline

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

Trained in the same way as K-NRM

- Pairwise training with hinge loss

(Dai, et al., 2018)

Conv-KNRM: Effectiveness

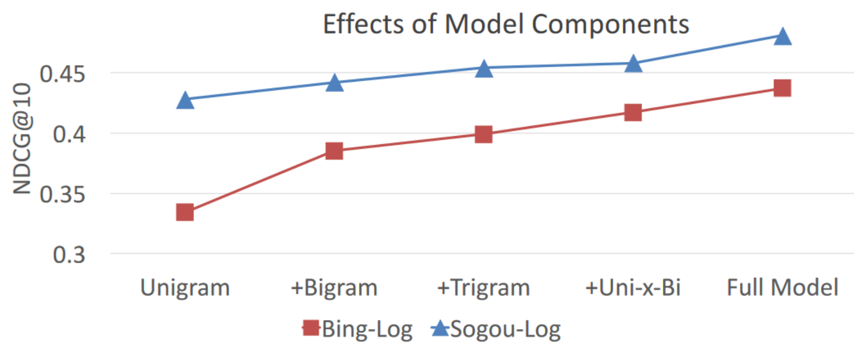
Conv-KNRM is more effective than K-NRM and several other baselines

Method	Sogou-Log MRR		Bing-Log MRR	
BM25	0.228	−33%	0.102	−61%
RankSVM	0.224	−34%	0.207	−22%
Coor-Ascent	0.242	−29%	0.208	−22%
DRMM	0.234	−31%	0.200	−25%
CDSSM	0.232	−32%	0.212	−20%
MP	0.240	−29%	$0.244^{\dagger\ddagger\S}$	−8%
K-NRM	$0.338^{\dagger\ddagger\S\P}$	−−	$0.265^{\dagger\ddagger\S\P}$	−−
Conv-KNRM	$0.358^{\dagger\ddagger\S\P*}$	+5%	$0.354^{\dagger\ddagger\S\P*}$	+34%

(Dai, et al., 2018)

Conv-KNRM: Effectiveness

How do the different model components contribute?



(Dai, et al., 2018)

Conv-KNRM: Sources of Effectiveness

N-grams

- A convolutional approach to forming n-grams
- Fewer parameters to train than discrete n-grams
  - E.g., 'white' + 'house' vs. white_house

Cross-matching n-grams of different lengths

- E.g., 'deep learning' matches 'convolutional neural networks'
- Another use of soft matching
- This is the most important part of Conv-KNRM

(Dai, et al., 2018)

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)
Re-ranking with BERT
DeepCT
doc2query
Summary

BERT

BERT is a recent neural language modeling architecture

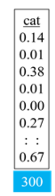
- Typical tasks
  - Classify sentence — e.g., sentiment classification
  - Classify a pair of sentences — e.g., similar sentences
  - Annotate a sentence — e.g., part-of-speech tagging
  - Annotate a pair of sentences — e.g., question answering
- It has had a major impact on all areas of language processing
  - Information retrieval (search engines)
  - Natural language processing
  - Machine translation
- This lecture covers some of its uses in search

(Devlin, et al., 2018)

BERT:
Input

BERT uses ideas that we have seen before

- A small'ish finite vocabulary
  - Lowercase: 'BERT' → 'bert'
  - 30,000 *word pieces*: 'unaffable' → 'un' '##aff' '##able'
    - » This is a form of morphological analysis (like stemming)
    - » Reduces the risk of out-of-vocabulary problems
- Word pieces (terms) are represented by embeddings
  - E.g., word2vec
  - We saw this idea in DRMM and K-NRM

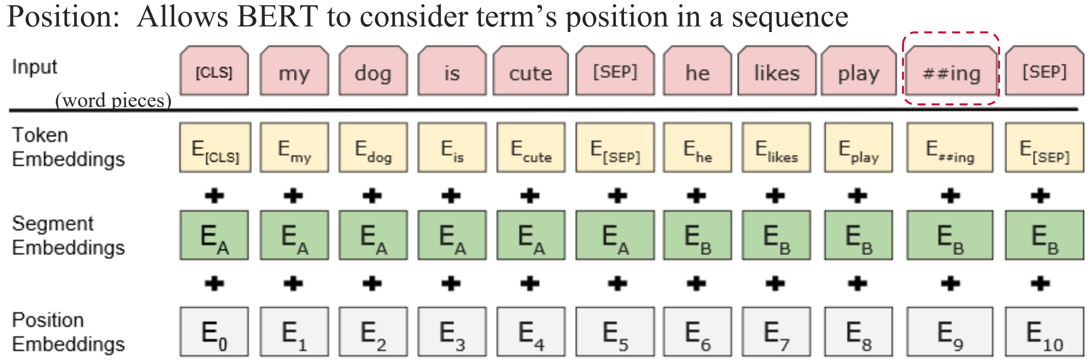


(Devlin, et al., 2018)

BERT:
Input

Token representation: Sum of three embeddings

- Token: Word pieces, [CLS] (class), [SEP] (separates segments)
- Segment: A unique text id for each text segment (e.g., 'A', 'B')
- Position: Allows BERT to consider term's position in a sequence

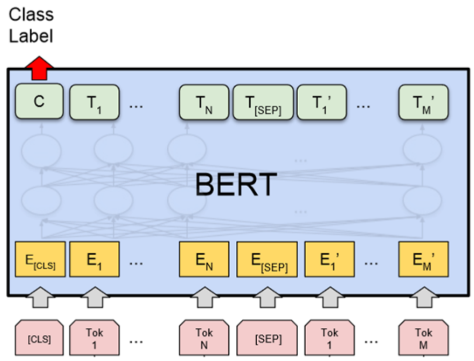


(Devlin, et al., 2018)

BERT

The input sequence passes through a feedforward network

- Each layer builds a new contextualized representation for each token
  - Including the task-specific class token



E.g., 12 layers

(Devlin, et al., 2018)

BERT: Common Pretrained Models

Commonly-used pretrained BERT models

Size	Layers	Hidden Size	Attention Heads	Parameters
Tiny	2	128	2	4M
Mini	4	256	4	11M
Small	4	512	4	29M
Medium	8	512	8	42M
Base	12	768	12	110M
Large	24	1024	16	340M

- Note the number of parameters
- Usually more parameters means higher accuracy, requires more training data, and longer training time

(Devlin, et al., 2018)

Re-Ranking with BERT

Re-ranking with BERT is similar to re-ranking with LeToR

- Use your favorite ranker to generate an initial ranking for q
  - E.g., BM25 or Indri
- For each document d_i in the top n
  - Use BERT to compare q, d_i
  - Use the comparison result to calculate p(relevant | q, d_i)
- Re-rank the top n documents by the new p(relevant | q, d_i) scores

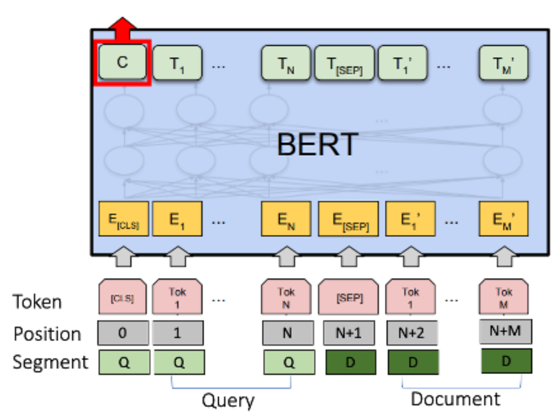
Easy!

(Dai and Callan, 2019)

Re-Ranking with BERT

BERT output is a sequence of token embeddings

- The special [CLS] token
- The tokens in q and d_i



BERT Re-ranker

(Dai and Callan, 2019)

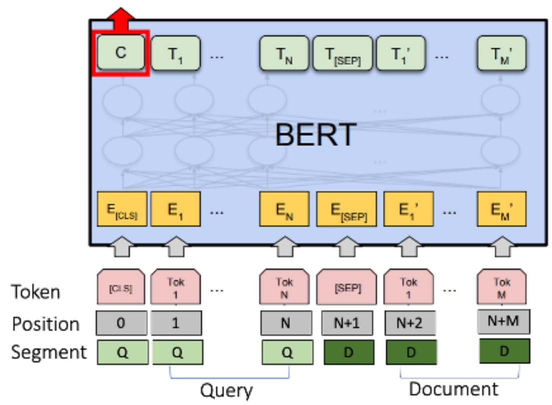
Re-Ranking with BERT

BERT output is a sequence of token embeddings

- The special [CLS] token
- The tokens in q and d_i

We want p(relevant | q, d_i)

- How are embeddings used to calculate p(relevant | q, d_i)?



BERT Re-ranker

(Dai and Callan, 2019)

21

21

Re-Ranking with BERT

BERT output is a sequence of token embeddings

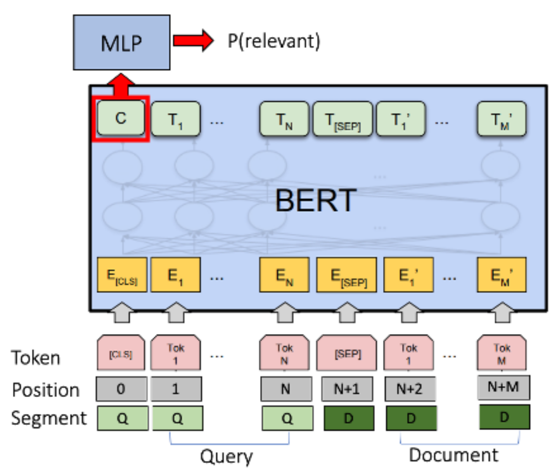
- The special [CLS] token
- The tokens in q and d_i

We want p(relevant | q, d_i)

- How are embeddings used to calculate p(relevant | q, d_i)?

Treat as a classification problem

- The [CLS] embedding is a feature vector
- Use it to predict { relevant, not relevant }
  - Multi-Layer Perceptron (MLP)



BERT Re-ranker

(Dai and Callan, 2019)

22

22

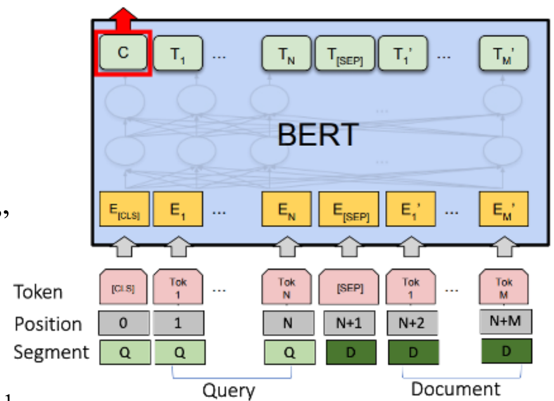
Re-Ranking with BERT: Training

Training occurs in three places

1. Train BERT

- Unsupervised training using a corpus
- E.g., masked language model (MLM)
  - » Predict the masked word

    "to be or not to ▇ , that is the question"
- Result: A general language model
  - » A task-neutral model
- Most people use pretrained models
  - » E.g., the pretrained models shown earlier



BERT Re-ranker

(Dai and Callan, 2019)

23

23

Re-Ranking with BERT: Training

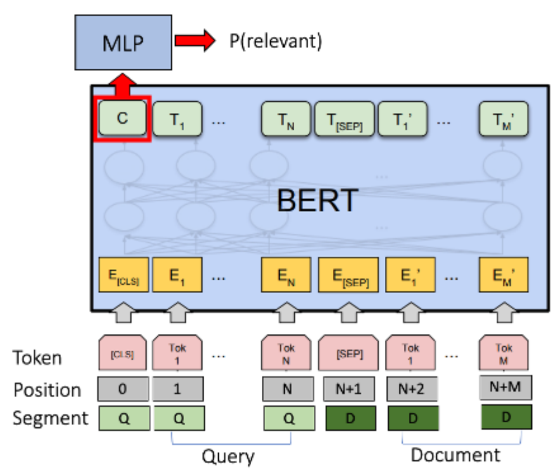
Training occurs in three places

2. Train the MLP

- Freeze BERT
- Treat the CLS embedding from the final layer as a feature vector
- Train the MLP
  - » Pointwise training

    (q, d, relevance value)
  - » Pairwise training data

    (q, d_{rel}, d_{nonrel})



BERT Re-ranker

(Dai and Callan, 2019)

24

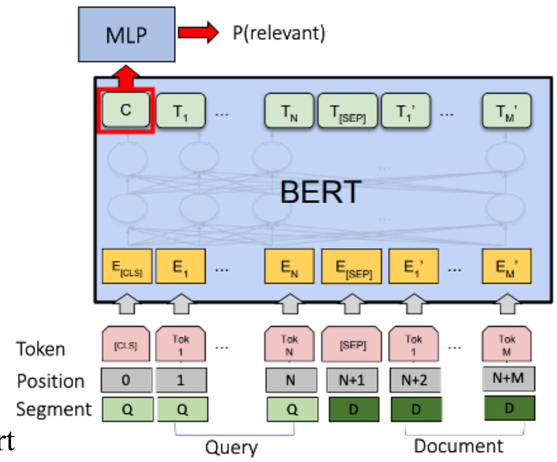
24

Re-Ranking with BERT: Training

Training occurs in three places

3. Fine-tune BERT (optional)

- Allow error signals from the MLP to backpropagate through BERT
  - » Requires a lot of training data
  - » E.g., a large search log
- Purpose: Tune the language model for a specific task such as search
  - » We saw this with K-NRM
  - » E.g., push ‘dog’ and ‘cat’ farther apart



BERT Re-ranker

(Dai and Callan, 2019)

Re-Ranking with BERT: Training

BERT fine-tuning is usually done with cross-entropy loss

Loss $= -\sum_{d\in R} log\big(predicted(d)\big) - \sum_{d\in NR} log\big(1 - predicted(d)\big)$

Note: The loss function is unrelated to the task

- This is true for all pointwise training procedures
- Search needs improved NDCG or MAP
- Thus, reducing loss may not improve search accuracy
  - We know this, but we tend to forget it
  - A common complaint: “Loss is reduced. Why is NDCG getting worse???”

True Score(d): 1

Predicted Score(d)

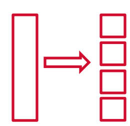
Re-Ranking with BERT: Practical Issues

BERT has high computational costs

- Typically run on a GPU or TPU
  - – Expensive hardware
- High memory requirements
- Memory depends on the length of the input sequence
  - – [CLS] $q_1 \ldots q_n$ [SEP] $d_1 \ldots d_m$
  - – Usually the sequence length is limited ≤ 512 tokens
    - » Longer sequences require unreasonable memory
  - – What if d is a long document?

Re-Ranking with BERT: Practical Issues

Controlling memory costs

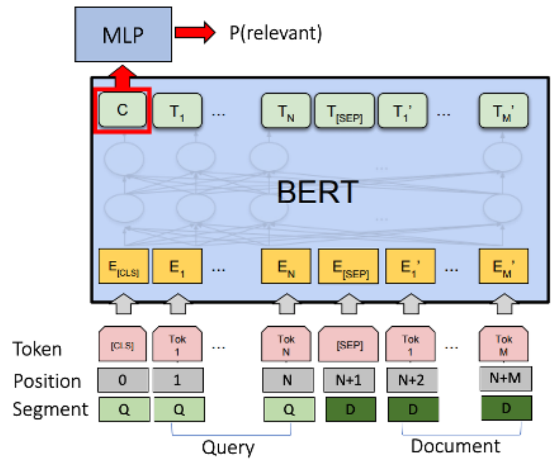


- Divide documents into passages
  - – Using document markup: E.g., <p> tags
  - – Every n words: E.g., n=150-300 is typical for English
  - – Add the document title to each passage to provide context
- Calculate a score for each passage
- Use passage scores to calculate a document score (‘pooling’)
  - – FirstP: Use the score of the first passage
  - – MaxP: Use the score of the highest scoring passage
  - – SumP: Add the passage scores
  - – …

(Dai and Callan, 2019)

Re-Ranking with BERT: Text Representation Still Matters

- Tokens → word pieces
- Divide documents into <u>passages</u>
  - Reduce computational cost
- Add the <u>title</u> to each passage
  - [CLS] q [SEP] title [SEP] passage [SEP]
  - Provides brief document-level context
- Tune BERT with a large <u>search log</u>
  - Effects similar to K-NRM
- Train the MLP for a specific dataset
  - E.g., ClueWeb09, Robust04



BERT Re-ranker

(Dai and Callan, 2019)

Re-Ranking with BERT

How well does it work?

	Model	Robust04 Title	Robust04 Description	ClueWeb09-B Title	ClueWeb09-B Description	
		nDCG@20				
Heuristic	BOW	0.417	0.409	0.268	0.234	
	SDM	0.427	0.427	0.279	0.235	
LTR	RankSVM	0.420	0.435	0.289	0.245	
	Coor-Ascent	0.427	0.441	**0.295**	0.251	
Neural	DRMM	0.422	0.412	0.275	0.245	
	Conv-KNRM	0.416	0.406	0.270	0.242	(no search log)
	BERT-FirstP	0.444†	0.491†	0.286	**0.272**†	
	BERT-MaxP	**0.469**†	**0.529**†	0.293	0.262†	
	BERT-SumP	0.467†	0.524†	0.289	0.261	
Query length:		**Short**	**Long**	**Short**	**Long**	

(Dai and Callan, 2019)

Re-Ranking with BERT

BERT re-rankers make better use of the text

- Continuous (soft) match among query & document terms
- The model is sequential (n-grams for varying n are built-in)
- Better understanding of the importance of each term in context
- Better with long queries with mixed term quality

Qry Type	Avg Len	nDCG@20 SDM		Coor-Ascent		BERT-MaxP	
Title	3	0.427		0.427		0.469	
Desc	14	0.404	- 5%	0.422	- 1%	0.529	+ 13%
Narr	40	0.278	- 35%	0.424	- 1%	0.487	+ 4%

Robust04

(Dai and Callan, 2019)

31

31

Re-Ranking with BERT

BERT re-rankers are the state-of-the-art for accurate ranking

- In most competitive situations, BERT re-rankers win

BERT re-rankers are not yet practical in some industry settings

- Requires expensive hardware
- Requires higher technical skill
- Best practices for long documents are still evolving
- Sensitive to training conditions
  - It is easy to get poor results with BERT re-rankers
  - Why? You did something wrong. What? It's not clear.

The state-of-the-art is changing … an important area to watch

32

32

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)

Re-ranking with BERT

DeepCT

doc2query

Summary

Understanding the Text

IR has a long history of shallow text understanding

- Index terms
  - Related concepts: Case, morphology
  - Unimportant concepts: Stopwords
  - Text structure: Bag-of-words, distance (n-grams, windows)
- Retrieval models
  - Importance: Term weighting based on term frequency

The older models are becoming uncompetitive

- Can they be updated to remain competitive?

Understanding the Text

BERT shows that frequency signals are no longer enough

In some cases, an upset **stomach** is the result of an allergic reaction to a certain type of food. It also may be caused by an irritation. Sometimes this happens from consuming too much alcohol or caffeine. Eating too many fatty foods or too much food in general may also cause an upset **stomach**.	passage$_1$
All parts of the body (muscles , brain, heart, and liver) need energy to work. This energy comes from the food we eat. Our bodies digest the food we eat by mixing it with fluids(acids and enzymes) in the **stomach**. When the **stomach** digests food, the carbohydrate (sugars and starches) in the food breaks down into another type of sugar, called glucose.	passage$_2$

- Each passage contains 'stomach' twice (tf=2)
- 'stomach' is more important in passage$_1$
- Color coding shows BERT's assessment of term importance

(Dai and Callan, 2020)

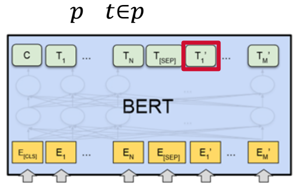
Understanding the Text: DeepCT

Better text understanding can improve older retrieval models

1. Train a BERT re-ranker, as before
2. Then train a model to map token embeddings to importance scores (token *t* in passage *p*)

$$\hat{y}_{t,p} = \vec{w}T_{t,p} + b$$

$$loss_{MSE} = \sum_{p}\sum_{t \in p}\left(y_{t,p} - \hat{y}_{t,p}\right)^2$$



(Dai and Callan, 2020)

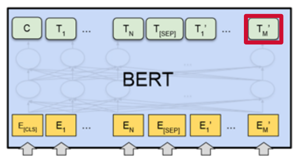
Understanding the Text:
DeepCT

Better text understanding can improve older retrieval models

1. Train a BERT re-ranker, as before
2. Then train a model to map token embeddings to importance scores (token *t* in passage *p*)

$$\hat{y}_{t,p} = \vec{w}T_{t,p} + b$$

$$loss_{MSE} = \sum_{p} \sum_{t \in p} (y_{t,p} - \hat{y}_{t,p})^2$$



(Dai and Callan, 2020)

Understanding the Text:
DeepCT

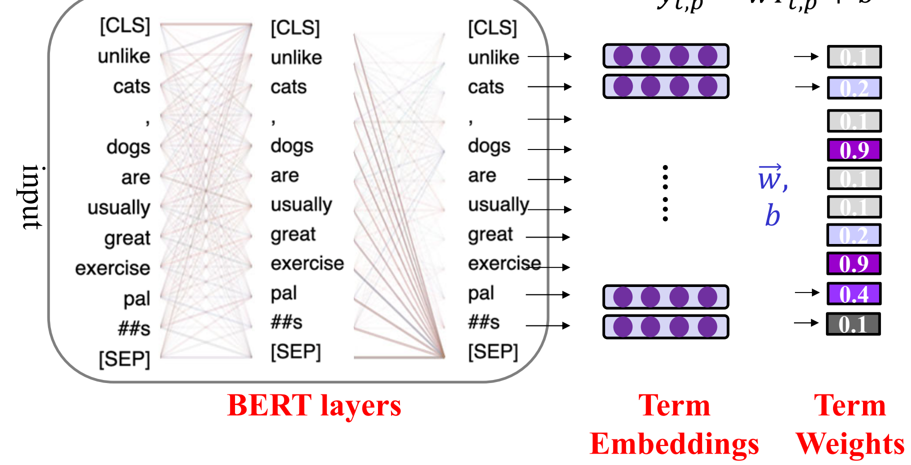
Better text understanding can improve older retrieval models

1. Train a BERT re-ranker, as before
2. Then train a model to map token embeddings to importance scores (token *t* in passage *p*)
   – There are several ways to generate training labels for importance of t ∈ p
      » Supervised: $QTR(t, d) = \frac{|Q_{d,t}|}{|Q_d|}$ (query term recall)
         • Q_d: Queries that consider *d* relevant
         • $Q_{d,t}$: Queries in $Q_{d,t}$ that contain *t*
      » Unsupervised: 1 if t is in the title, 0 otherwise
      » Unsupervised: % of inlinks to d that contain t

(Dai and Callan, 2020)

Understanding the Text: DeepCT

$$\hat{y}_{t,p} = \vec{w}T_{t,p} + b$$



(Dai, 2020)

Understanding the Text: DeepCT

Supervised (query term recall)

$$loss_{MSE} = \sum_{p} \sum_{t \in p} (y_{t,p} - \hat{y}_{t,p})^2$$

unlike	0.0	0.1
cats	0.1	0.2
,	0.0	0.1
dogs	1.0	0.9
are	0.0	0.1
usually	0.2	0.1
great	0.1	0.2
exercise	0.8	0.9
pal	0.4	0.4
##s	0.0	0.1

Unsupervised (title terms)

$$loss_{MSE} = \sum_{p} \sum_{t \in p} (y_{t,p} - \hat{y}_{t,p})^2$$

unlike	0.0	0.1
cats	0.0	0.2
,	0.0	0.1
dogs	1.0	0.9
are	0.0	0.1
usually	0.0	0.1
great	0.0	0.2
exercise	1.0	0.9
pal	0.0	0.4
##s	0.0	0.1

(Dai, 2020)

Understanding the Text: DeepCT

"Unlike cats, dogs are usually great exercise pals. Many breeds enjoy running and hiking, and will happily trek along on any trip. Exercise time varies…"

When a token occurs more than once, use its highest score

- E.g., 'exercise' in this example

(Dai, 2020)

Understanding the Text: DeepCT

Better text understanding can improve older retrieval models

3. Transform scores to positive integer importance weights
   – Scores tend to fall in [0, 1]
   – $100x$, $\sqrt{100x}$, …
4. In a typical inverted index, replace tf with importance weights
   – E.g., in your QryEval inverted lists
5. Rank with BM25 or Indri as usual

(Dai and Callan, 2020)

Understanding the Text: DeepCT

Better text understanding can improve older retrieval models

	Ranking Method	dev MRR@10		test MRR@10	
Single-Stage	Official BM25	0.167	-30%	0.165	-31%
	Doc2Query BM25 (Nogueira et al., 2019)	0.215	-12%	0.218	-9%
	DeepCT-Index BM25	0.243	–	0.239	–
Multi-Stage	Feature-based LeToR	0.195	-20%	0.191	-20%
	K-NRM (Xiong et al., 2017b)	0.218	-10%	0.198	-17%
	Duet V2 (Mitra and Craswell, 2019)	0.243	+0%	0.245	+2%
	Conv-KNRM (Dai et al., 2018)	0.247	+2%	0.247	+3%
	FastText+Conv-KNRM (Hofstätter et al., 2019)	0.277	+14%	0.290	+21%
	BERT Re-Ranker (Nogueira and Cho, 2019)	0.365	+50%	0.359	+50%

MS MARCO, official evaluation

(Dai and Callan, 2020)

Understanding the Text

Better text understanding can improve older retrieval models

- Better text understanding → better term importance weights
  - Also seen in Nogueira, et al., 2019
- Better initial retrieval → better BERT reranking (not shown)
  - 3 of the top 4 MARCO leaderboard systems use DeepCT

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)
Re-ranking with BERT
DeepCT
doc2query
Summary

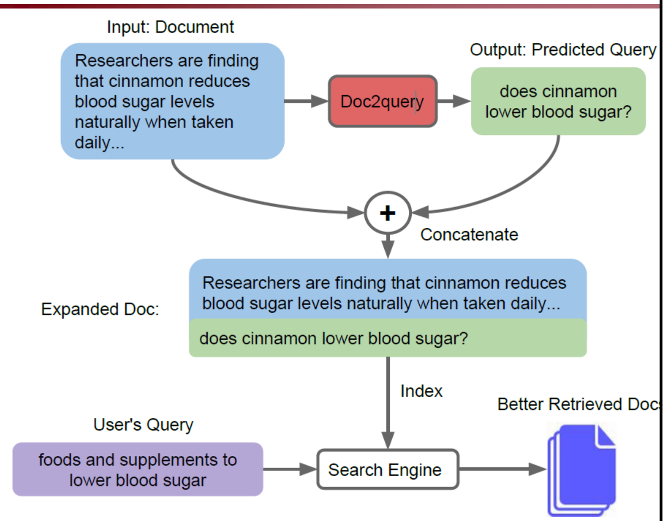
doc2query

For each document d in the corpus

- Automatically generate questions that d can answer
- Add these questions to d
  - – Append to the end of d
  - – *document expansion*

Use the expanded documents to build an ordinary inverted index

Use BM25 for first-stage retrieval



(Nogueira, et al., 2019)

doc2query: Examples

Document: July is the hottest month in Washington DC with an average temperature of 27C (80F) and the coldest is January at 4C (38F) with the most daily sunshine hours at 9 in July. The wettest month is May with an average of 100mm of rain.

Target query: what is the temperature in washington

Predicted query: weather in washington dc

Document: The Delaware River flows through Philadelphia into the Delaware Bay. It flows through and aqueduct in the Roundout Reservoir and then flows through Philadelphia and New Jersey before emptying into the Delaware Bay.

Target query: where does the delaware river start and end

Predicted Query: what river flows through delaware

(Nogueira, et al., 2019)

47

47

doc2query: Examples

- why was the manhattan project important
- why was the manhattan charter
- what did the manhattan project do
- what was the importance of manhattan
- what was the importance of manhattan communication
- why was the manhattan project created
- what was the manhattan project
- why was the manhattan project important
- why was manhattan an important factor
- what was the result of the manhattan
- what is the manhattan project
- : : : : : : : : :

- what is vascular)
- what is vascular) material
- what is vascular) in plants
- what is a Phloem
- what is vascular) in photosynthesis
- what are vascular plants
- what is vascular plants
- what is a vascular plant
- what do vascular plants do
- what are vascular plants
- what is a vascular plant
- what are xylem and vascular plants
- : : : : : : : : :

(https://github.com/nyu-dl/dl4ir-doc2query)

48

48

doc2query: Examples

- what was the impact of the civil war
- what was the impact of the american industrial
- what was the impact of the civil industrial
- what was the impact of the american civil war
- what was the result of the industrial of the industrial
- what did the treaty of Exclusion do
- what was the treaty of Exclusion
- what did the Congress treaty do
- what did the treaty of Exclusion immigration. do
- how much did the treaty of Exclusion affect
- : : : : : : : : :

- what is costa rica
- what is costa rica known for
- what is costa rica prime
- what is conducive
- what is costa rica known for?
- Rica: Medical
- what is Medical Medical
- what is Medical Medical made of
- what is Medical
- what is Medical in costa rica
- what is a medical services,
- what is Medical in services,
- : : : : : : : : :

(https://github.com/nyu-dl/dl4ir-doc2query)

doc2query

How are queries generated?

- Train: Use (q, $d_{relevant}$) pairs to train a sequence-to-sequence transformer model
  - $d_{relevant} \rightarrow q$
  - Datasets
    - » MS MARCO Dev set for training (6,900 queries)
    - » TREC CAR (3M queries)
- Test: Predict 10 queries per document
  - Use top-k sampling

A later version uses the T5 transformer, which generates better queries

- Better queries enables using 40 queries per document, which is much more effective

(Nogueira, et al., 2019)

doc2query

doc2query improves BM25 by about 15% (and docT5query is better – 25%?)

What are the effects?

- **Term reweighting:** tf is increased for some terms
  - 69% of the non-stopword terms in generated queries were already in the document
  - "Researchers find that living with cats reduces allergies in children." →
    "Do cats reduce allergies in children?"
- **Reduced vocabulary mismatch:** new terms are added to the document
  - 31% of the non-stopword terms in generated queries were not in the document
  - "Researchers find that living with cats reduces allergies in children." →
    "Are kittens healthy for kids?"

(Lin, et al., 2020)
(Nogueira, et al., 2019)

51

51

doc2query

What are the main sources of improvement?

	Method	MS MARCO Passage MRR@10	MS MARCO Passage Recall@1k
(1)	Original Text	0.184	0.853
(2a)	+ Expansion w/ New Terms	0.195	0.907
(2b)	+ Expansion w/ Copied Terms	0.221	0.893
(2c)	+ Expansion w/ Copied Terms + New Terms	0.277	0.944
(3)	Only Expansion Terms (Without Original Text)	0.263	0.927

The effect seems more complex than typical query expansion

- Not just term reweighting and adding related vocabulary
- The expansion queries appear to summarize the document quite well
- The effects are not well-understood, but they appear to be consistent

(Lin, et al., 2020)

52

52

doc2query

doc2query is compatible with other familiar techniques

- Use your favorite initial ranker (BM25, Indri, VSM)
- Pseudo relevance feedback
- BERT reranking
  - Better initial ranking produces better re-ranking

(Lin, et al., 2020)

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)
Re-ranking with BERT
DeepCT
doc2query
Summary

Summary

Continuous representations are popular again

- Lexical (DSSM), conceptual (DRMM, K-NRM, Conv-KNRM)

Two main types of architectures

- Representation-based vs. interaction-based

Integration of exact-match and soft-match signals

- Older systems were discrete or continuous, not both
- The combination seems effective and reliable (robust)

Some architectures require much training data, some don't

- E.g., trained (much data) vs static (little data) embeddings

Summary

No feature engineering … but much network engineering

- Ignore the hype … not necessarily less work

Poor understanding of how well and why the system works

- Early neural rankers compared to weak baselines (i.e., not LeToR)
- What is the contribution of different parts of the network?
- Did the system learn (good), or did it memorize (less good)?
  - Neural ranking systems are good at memorizing data

Some research embeds familiar ideas in complicated networks

- log (tf), idf, proximity, multiple bags-of-words (title, body, …)

Summary

Vocabulary understanding tailored for search tasks

- Continuous representations tuned for search tasks

Contextualized language models trained from massive data

- Better understanding leads to better text similarity

Document understanding tailored for search tasks

- Better shallow representations
  - Reweight terms: DeepCT
  - Document expansion: doc2query (and doc2query extensions)

57

57

Summary

Better text understanding can improve older retrieval models

Why is this important?

- These models are still used widely
  - Alone, and as the first stage of re-ranking architectures
- Text understanding in these models hasn't changed in a long time
  - Much research, but little change in the state of the art
- Deep text analysis + efficient matching with inverted indexes
  - Moves a computationally complex task to indexing
  - Encourages us to explore hybrid indexing strategies

There is more opportunity here than many people realized

58

58

Summary

Right now the importance of neural ranking is unclear

- We are seeing convincing wins over strong LeToR systems in some settings
  - LeToR is still best in many situations
- Discrete BoW representations are being updated or replaced
  - After 50 years … wow!
- Contextual information is required to be successful
  - Document context
  - Search (user) context
- Strong opinions on both sides … much debate

It isn't clear what the next generation of systems will look like

References

- Z. Dai and J. Callan. Deeper text understanding for IR with contextual neural language modeling. SIGIR 2019.
- Z. Dai and J. Callan. Context-aware sentence/passage term importance estimation for first stage retrieval. arXiv:1910.10687. 2019.
- J. Dalton, C. Xiong, and J. Callan. CAsT 2019: The Conversational Assistance Track overview. TREC 2019.
- J. Devlin, M.-W. Chang, K. Lee, K. Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. arxiv: 1810.04805. 2018.
- J. Lin, R. Nogueira, and A. Yates. Pretrained transformers for text ranking: BERT and beyond. arxiv 2010.06467. 2020.
- R. Nogueira, W. Yang, J. Lin, and K. Cho. Document expansion by query prediction. arxiv 1904.08375. 2019.