Learning to Reweight Terms with Distributed Representations

Guoqing Zheng, Jamie Callan

School of Computer Science
Carnegie Mellon University

August 12, 2015

Outline

- **Goal:**
  Assign weights to query terms for better retrieval results
- **Motivation**
- **Method**
- **Evaluation**
- **Conclusions**

Motivation

- Assigning weights to terms is not a new problem

$$score(q, d) = \sum_{t \in q} s(t, q, C) f(t, d) \quad (1)$$

- Brief summary of existing representative methods:

Type	Method	Performance	Features	Model
Unsupervised	Constant	bad	-	-
	IDF	average	-	-
Supervised (Rank measure)	LtR-methods (WSD)	good	manual, external	non-convex
Supervised (Term recall)	Zhao et al	average	manual	requires SVD

- Can we have something that is directly term weight supervised, can lead to good retrieval performance without manual / expensive feature construction?

Probably yes?

- What might be a good choice of term weights? Term recall
- The probability of term t of query q appears in relevant documents R_q, can be estimated as $\frac{|R_{q,t}|}{|R_q|}$ if relevance judgments are available
- True term recall proves to be highly effective as shown by Zhao et al[3, 2]
- But which features are good to predict that?

Distributed word vectors

- Distributed word vectors learnt from a large corpora can be used to effectively measure semantic similarity

$$vec(\text{King}) - vec(\text{Man}) \approx vec(\text{Queen}) - vec(\text{Woman})$$

- Another important characteristic of word vector is the **addition property**, i.e., it's often semantically meaningful to construct word vectors for phrases from those of individual terms, such as representing

$$vec(\text{Chinese river}) \approx vec(\text{Chinese}) + vec(\text{river})$$

Idea in this work

- Intuitively, term recall measures the relative importance of a term w.r.t the whole query in determining document-query relevance
- With the **addition property** of word vectors, we can construct feature vectors for terms from their word vector representations, which resembles the above intuition of term recall
- For a term t in query q, we construct the feature vector $x(t)$ given the word vector representation $w(t)$ as

$$x(t) \equiv vec(t) - vec(q)$$

  where $vec(q)$ is the average word vector of all the terms in q.
- The above defined feature vector is essentially the offset vector of a term to the center of the entire query

Feature vector illustration

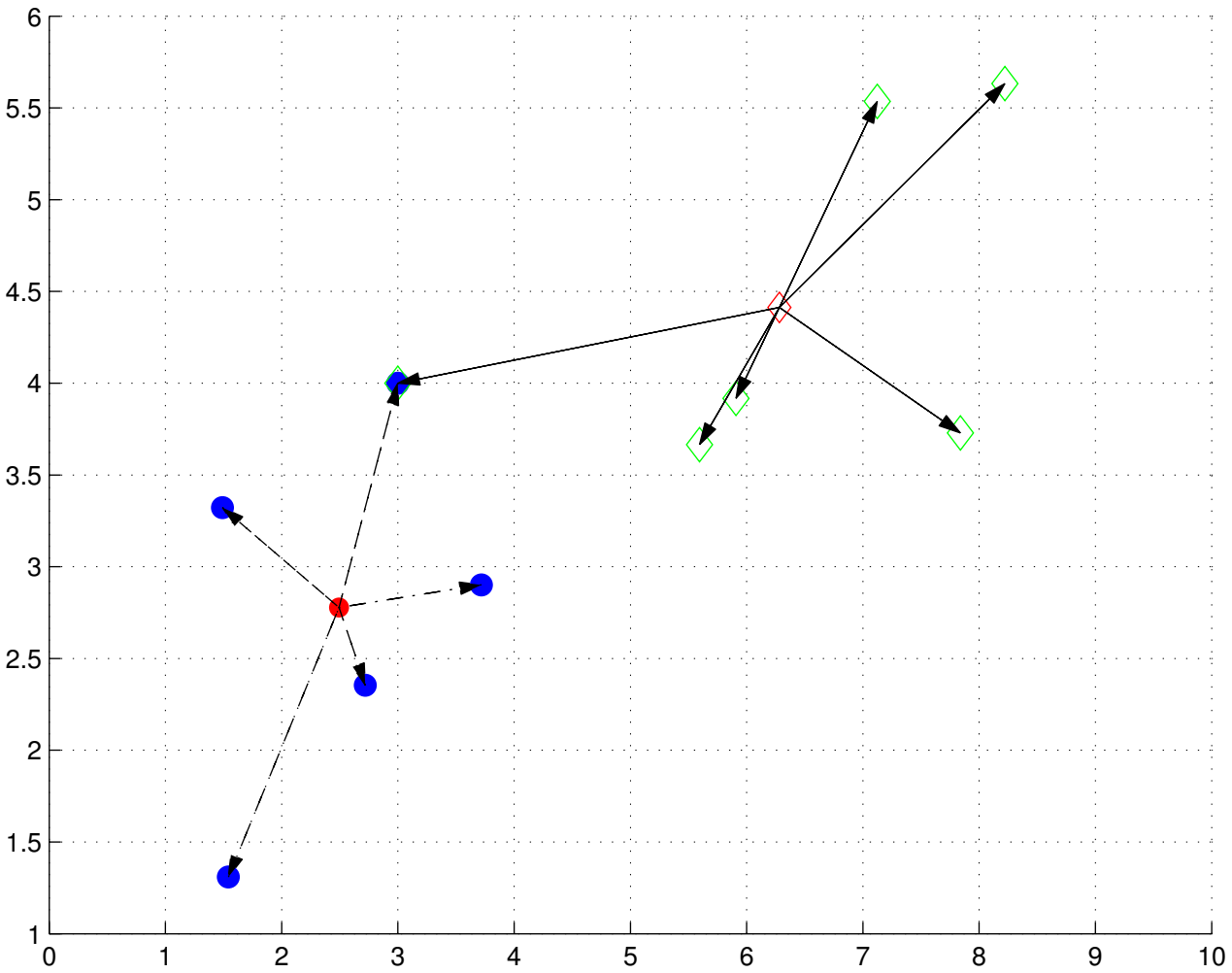


Figure: Demonstration of transforming global 2-dimensional word vectors into feature vectors local to the query.

Term recall learning with feature vectors

- Model term recall as a function of the feature vectors defined
- Preprocessing: transform term recall r from [0,1] to $y \in \mathbb{R}$ by a logit function $y = logit(r)$
- Models the transformed term recall y as a linear function of the feature vectors x
- Employ ℓ_1 norm regularization in learning (ℓ_2 norm also tried)
- LASSO optimization:

$$\hat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta} \in \mathbb{R}^p} \frac{1}{2} \sum_{i=1}^{M} \sum_{j=1}^{n_i} (y_{ij} - \boldsymbol{\beta}^\top \boldsymbol{x}_{ij})^2 + \lambda \|\boldsymbol{\beta}\|_1 \qquad (2)$$

- Predict term recall for testing query term with feature vector $\boldsymbol{x}_*$ as

$$\widehat{\mathbb{P}(t_*|R)} = \text{sigmoid}\left(\hat{\boldsymbol{\beta}}^\top \boldsymbol{x}_*\right) = \frac{\exp\{\hat{\boldsymbol{\beta}}^\top \boldsymbol{x}_*\}}{1 + \exp\{\hat{\boldsymbol{\beta}}^\top \boldsymbol{x}_*\}} \qquad (3)$$

Incorporating predicted weights to query models (I)

Base query models:

- Bag-of-Words (`BOW`):

  ```
  apple pie recipe
  ```

- Sequential Dependency model (`SD`):

  ```
  #weight(
  0.8 #combine( apple pie recipe )
  0.1 #combine( #1(apple pie)
                #1(pie recipe) )
  0.1 #combine(
                #uw8(apple pie)
                #uw8(pie recipe) ) )
  ```

Incorporating predicted weights to query models (II)

Term recall enhanced query models:

- DeepTR-BOW:

  #weight($\widehat{\mathbb{P}}(\texttt{apple}|R)$ apple
  $\widehat{\mathbb{P}}(\texttt{pie}|R)$ pie
  $\widehat{\mathbb{P}}(\texttt{recipe}|R)$ recipe)

- DeepTR-SD:

  #weight(
  0.8 #weight($\widehat{\mathbb{P}}(\texttt{apple}|R)$ apple
  $\widehat{\mathbb{P}}(\texttt{pie}|R)$ pie
  $\widehat{\mathbb{P}}(\texttt{recipe}|R)$ recipe)
  0.1 #combine(#1(apple pie)
  #1(pie recipe))
  0.1 #combine(
  #uw8(apple pie)
  #uw8(pie recipe)))

Evaluation

- Data sets

Collection	# Docs	# Words	TREC Topics
ROBUST04	528K	253M	351-450, 601-700
WT10g	1,692K	1,076M	451-550
GOV2	25,205K	24,007M	701-850
ClueWeb09B	29,038K	23,890M	1-200

- Baseline query models: Bag of Words (BOW), Sequential Dependency model (SD), Weighted Sequential Dependency model (WSD) [1]
- Retrieval models: Language model and BM25
- Evaluation metrics: P@10, MAP for all collections, NDCG@10 and ERR@20 for collections with graded relevance judgments

Evaluation

Word vector source:

- trained from all above collections with dimensions ranging from 100, 300, 500
- pre-trained by Google trained from Google News (100 billion words) with dimension 300

Experimental results

Retrieval results of `DeepTR-BOW` with language model (dimension=300)

Query Model	ROBUST04 MAP	WT10g MAP	GOV2 MAP	ClueWeb09B MAP
`BOW`	0.2512	0.1943	0.2488	0.0702
`SD`	0.2643	0.2032	0.2688	0.0745
`WSD`	0.2749	0.2260	0.2946	-
`DeepTR-BOW (Corpus)`	0.2591^{b} *(+3.2/-1.9)*	0.2103 *(+8.2/+3.5)*	0.2646^{b} *(+6.3/-1.6)*	0.0718 *(+2.2/-3.6)*
`DeepTR-BOW (GOV2)`	0.2650^{b} *(+5.5/+0.3)*	0.2111^{b} *(+8.7/+3.9)*	0.2646^{b} *(+6.3/-1.6)*	0.0741 *(+5.6/-0.5)*
`DeepTR-BOW (ClueWeb09B)`	0.2657^{b} *(+5.8/+0.5)*	0.2129 *(+9.6/+4.8)*	0.2685^{b} *(+7.9/-0.1)*	0.0718 *(+2.2/-3.6)*
`DeepTR-BOW (Google)`	0.2673^{b} *(+6.4/+1.2)*	0.2122^{b} *(+9.3/+4.5)*	0.2630^{b} *(+5.7/-2.2)*	0.0732 *(+4.2/-1.8)*

b : Statistically significant difference with `BOW`
s : Statistically significant difference with `SD`

$\Rightarrow$ Weighted BoW queries significantly outperforms unweigted BoW queries and are even comparable to SD queries.

Experimental results (cont.)

Retrieval results of DeepTR-SD with language model (dimension=300)

Query Model	ROBUST04	WT10g	GOV2	ClueWeb09B
	MAP	MAP	MAP	MAP
BOW	0.2512	0.1943	0.2488	0.0702
SD	0.2643	0.2032	0.2688	0.0745
WSD	0.2749	0.2260	0.2946	-
DeepTR-SD (Corpus)	0.2754^{b}_{s} *(+9.6/+4.2)*	0.2182^{b}_{s} *(+12.3/+7.4)*	0.2831^{b}_{s} *(+13.8/+5.3)*	0.0748 *(+6.5/+0.5)*
DeepTR-SD (GOV2)	0.2781^{b}_{s} *(+10.7/+5.2)*	0.2223^{b}_{s} *(+14.4/+9.4)*	0.2831^{b}_{s} *(+13.8/+5.3)*	0.0806^{b}_{s} *(+14.8/+8.2)*
DeepTR-SD (ClueWeb09B)	0.2810^{b}_{s} *(+11.9/+6.3)*	0.2279^{b}_{s} *(+17.3/+12.1)*	0.2890^{b}_{s} *(+16.2/+7.5)*	0.0748 *(+6.5/+0.5)*
DeepTR-SD (Google)	0.2842^{b}_{s} *(+13.1/+7.5)*	0.2256^{b}_{s} *(+16.1/+11.0)*	0.2814^{b}_{s} *(+13.1/+4.7)*	0.0780^{b}_{s} *(+11.0/+4.7)*

b : Statistically significant difference with BOW

s : Statistically significant difference with SD

$\Rightarrow$ DeepTR-SD significantly outperforms both BoW and SD queries.

Experimental results (cont.)

Retrieval results of DeepTR-BOW with BM25 (dimension=300)

Query Model	ROBUST04	WT10g	GOV2	ClueWeb09B
	MAP	MAP	MAP	MAP
BOW	0.2460	0.1783	0.2334	0.0566
SD	0.2546	0.1817	0.2409	0.0600
DeepTR-BOW (Corpus)	0.2566^{b} *(+4.3/+0.8)*	0.1903^{b} *(+6.7/+4.7)*	0.2430^{b} *(+4.1/+0.9)*	0.0593^{b} *(+4.7/-1.2)*
DeepTR-BOW (GOV2)	0.2561^{b} *(+4.1/+0.6)*	0.1910^{b} *(+7.1/+5.1)*	0.2430^{b} *(+4.1/+0.9)*	0.0596^{b} *(+5.3/-0.7)*
DeepTR-BOW (ClueWeb09B)	0.2590^{b} *(+5.3/+1.8)*	0.1932^{b} *(+8.3/+6.3)*	0.2462^{b} *(+5.5/+2.2)*	0.0593^{b} *(+4.7/-1.2)*
DeepTR-BOW (Google)	0.2600^{b} *(+5.7/+2.1)*	0.1922^{b} *(+7.8/+5.7)*	0.2449^{b} *(+4.9/+1.7)*	0.0584 *(+3.1/-2.7)*

b : Statistically significant difference with BOW

s : Statistically significant difference with SD

$\Rightarrow$ Similar results observed for BM25 retrieval.

Experimental results (cont.)

Retrieval results of DeepTR-SD with BM25 (dimension=300)

Query Model	ROBUST04	WT10g	GOV2	ClueWeb09B
	MAP	MAP	MAP	MAP
BOW	0.2460	0.1783	0.2334	0.0566
SD	0.2546	0.1817	0.2409	0.0600
DeepTR-SD (Corpus)	0.2595^{b}_{s} *(+5.5/+1.9)*	0.1944^{b}_{s} *(+9.0/+7.0)*	0.2478^{b}_{s} *(+6.1/+2.9)*	0.0613^{b}_{s} *(+8.2/+2.1)*
DeepTR-SD (GOV2)	0.2609^{b}_{s} *(+6.1/+2.5)*	0.1941_{s} *(+8.8/+6.8)*	0.2478^{b}_{s} *(+6.1/+2.9)*	0.0616^{b}_{s} *(+8.8/+2.6)*
DeepTR-SD (ClueWeb09B)	0.2630^{b}_{s} *(+6.9/+3.3)*	0.1951^{b}_{s} *(+9.4/+7.3)*	0.2502^{b}_{s} *(+7.2/+3.9)*	0.0613^{b}_{s} *(+8.2/+2.1)*
DeepTR-SD (Google)	0.2627^{b}_{s} *(+6.8/+3.2)*	0.1938_{s} *(+8.7/+6.7)*	0.2461^{b} *(+5.4/+2.2)*	0.0604 *(+6.8/+0.7)*

b : Statistically significant difference with BOW

s : Statistically significant difference with SD

$\Rightarrow$ Similar results observed for BM25 retrieval.

Word vector dimensionality

vectors trained from ClueWeb09B, Language Model retrieval

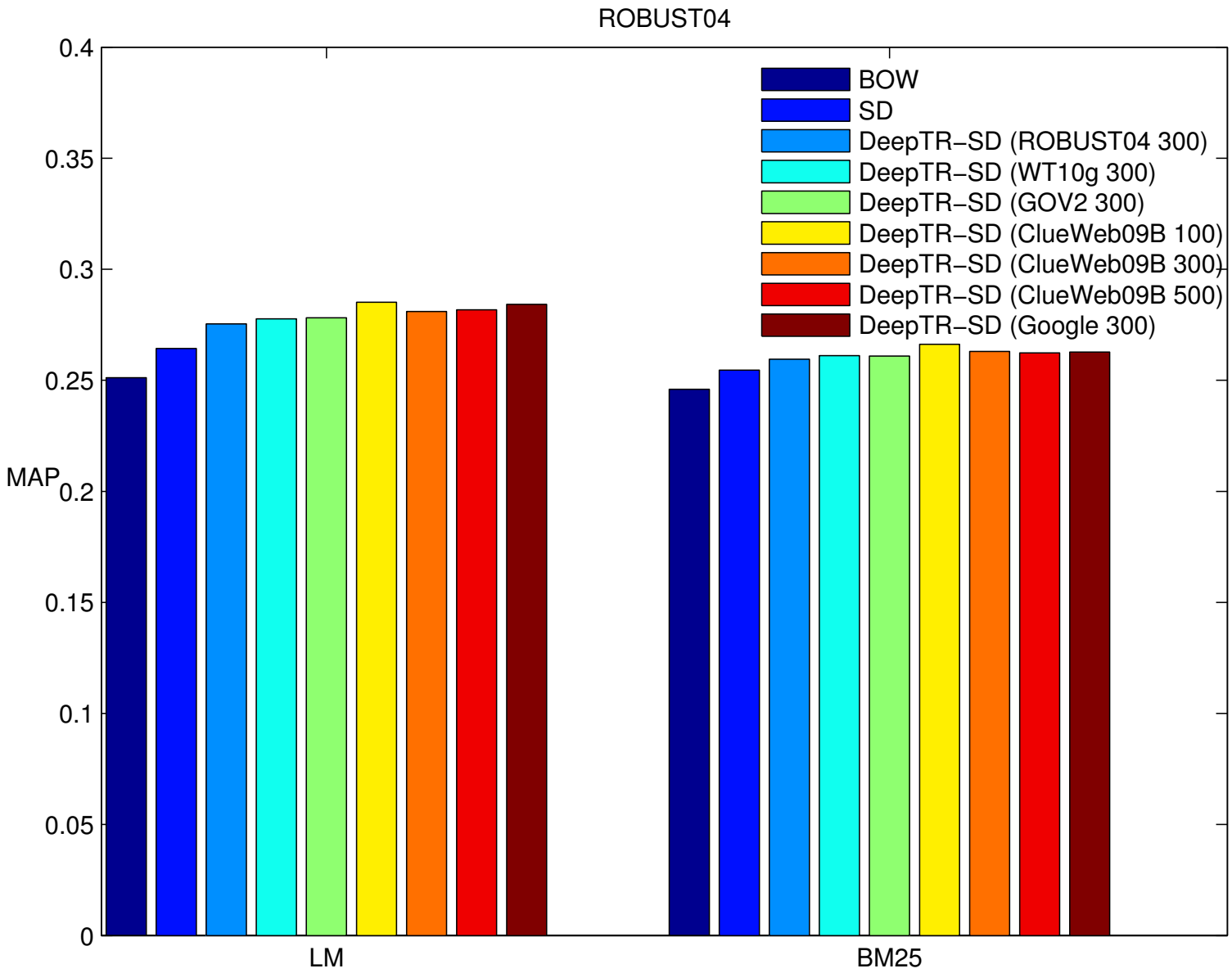
Query Model	ROBUST04 MAP	WT10g MAP	GOV2 MAP	ClueWeb09B MAP
BOW	0.2512	0.1943	0.2488	0.0702
SD	0.2643	0.2032	0.2688	0.0745
WSD	0.2749	0.2260	0.2946	-
DeepTR-BOW (100)	0.2681^{b} *(+6.7/+1.4)*	0.2239^{b} *(+15.3/+10.2)*	0.2667^{b} *(+7.2/-0.8)*	0.0727 *(+3.6/-2.4)*
DeepTR-BOW (300)	0.2657^{b} *(+5.8/+0.5)*	0.2129 *(+9.6/+4.8)*	0.2685^{b} *(+7.9/-0.1)*	0.0718 *(+2.2/-3.6)*
DeepTR-BOW (500)	0.2679^{b} *(+6.6/+1.4)*	0.2179^{b} *(+12.1/+7.2)*	0.2655^{b} *(+6.7/-1.2)*	0.0726 *(+3.4/-2.5)*
DeepTR-SD (100)	0.2851^{b}_{s} *(+13.5/+7.9)*	0.2373^{b}_{s} *(+22.1/+16.8)*	0.2848^{b}_{s} *(+14.4/+5.9)*	0.0778^{b}_{s} *(+10.8/+4.5)*
DeepTR-SD (300)	0.2810^{b}_{s} *(+11.9/+6.3)*	0.2279^{b}_{s} *(+17.3/+12.1)*	0.2890^{b}_{s} *(+16.2/+7.5)*	0.0748 *(+6.5/+0.5)*
DeepTR-SD (500)	0.2817^{b}_{s} *(+12.2/+6.6)*	0.2306^{b}_{s} *(+18.7/+13.5)*	0.2860^{b}_{s} *(+14.9/+6.4)*	0.0781^{b}_{s} *(+11.3/+4.9)*

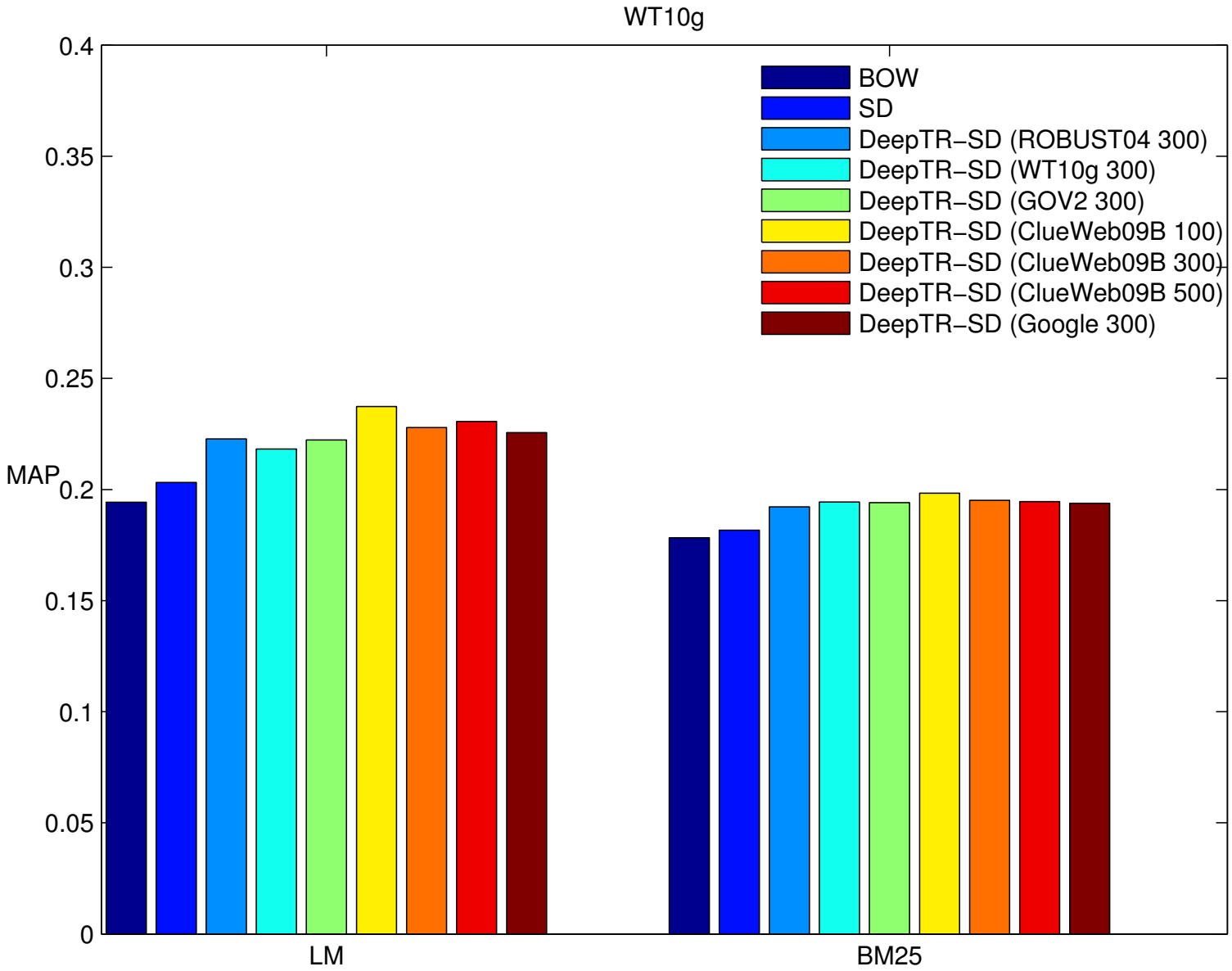
b : Statistically significant difference with BOW

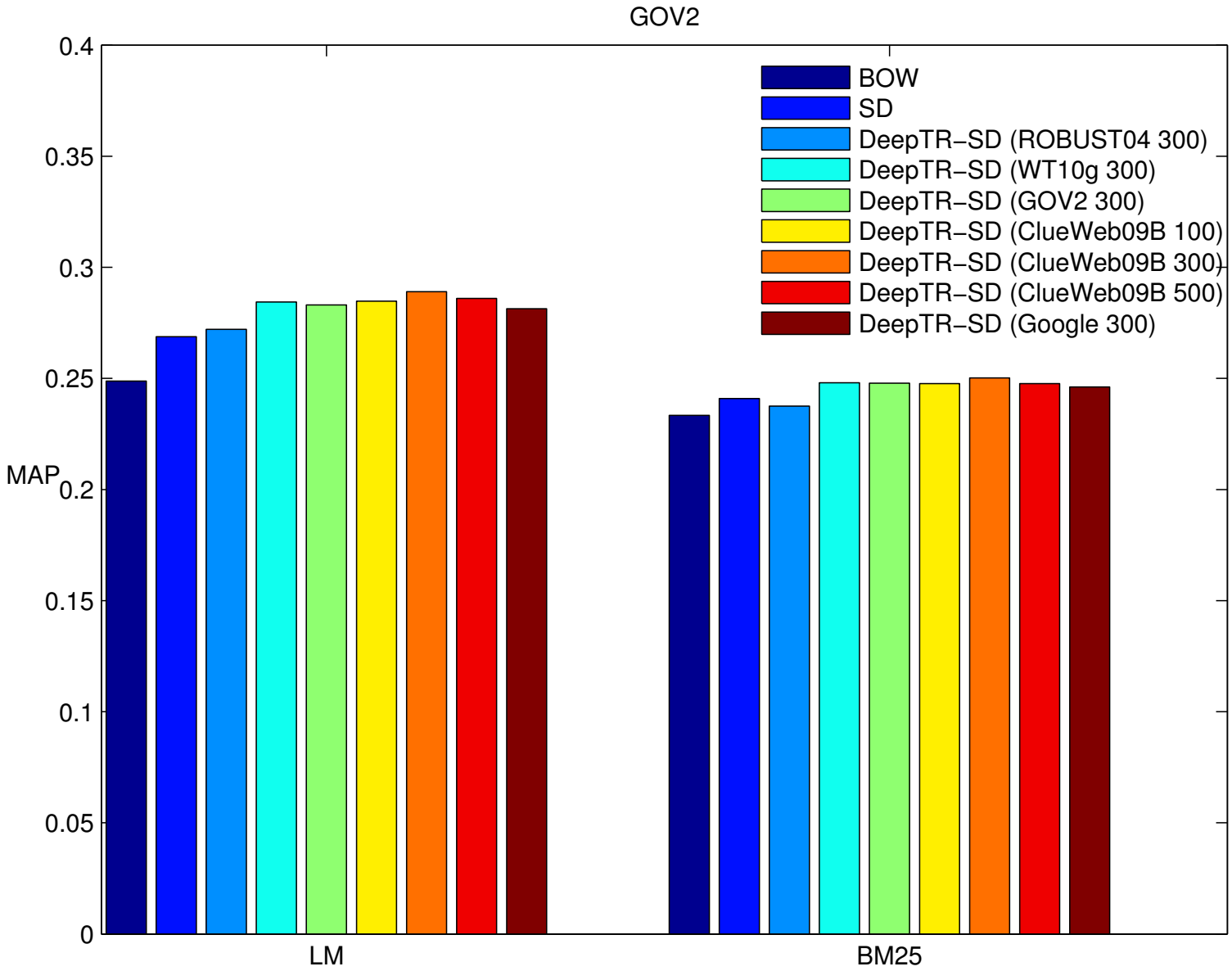
s : Statistically significant difference with SD

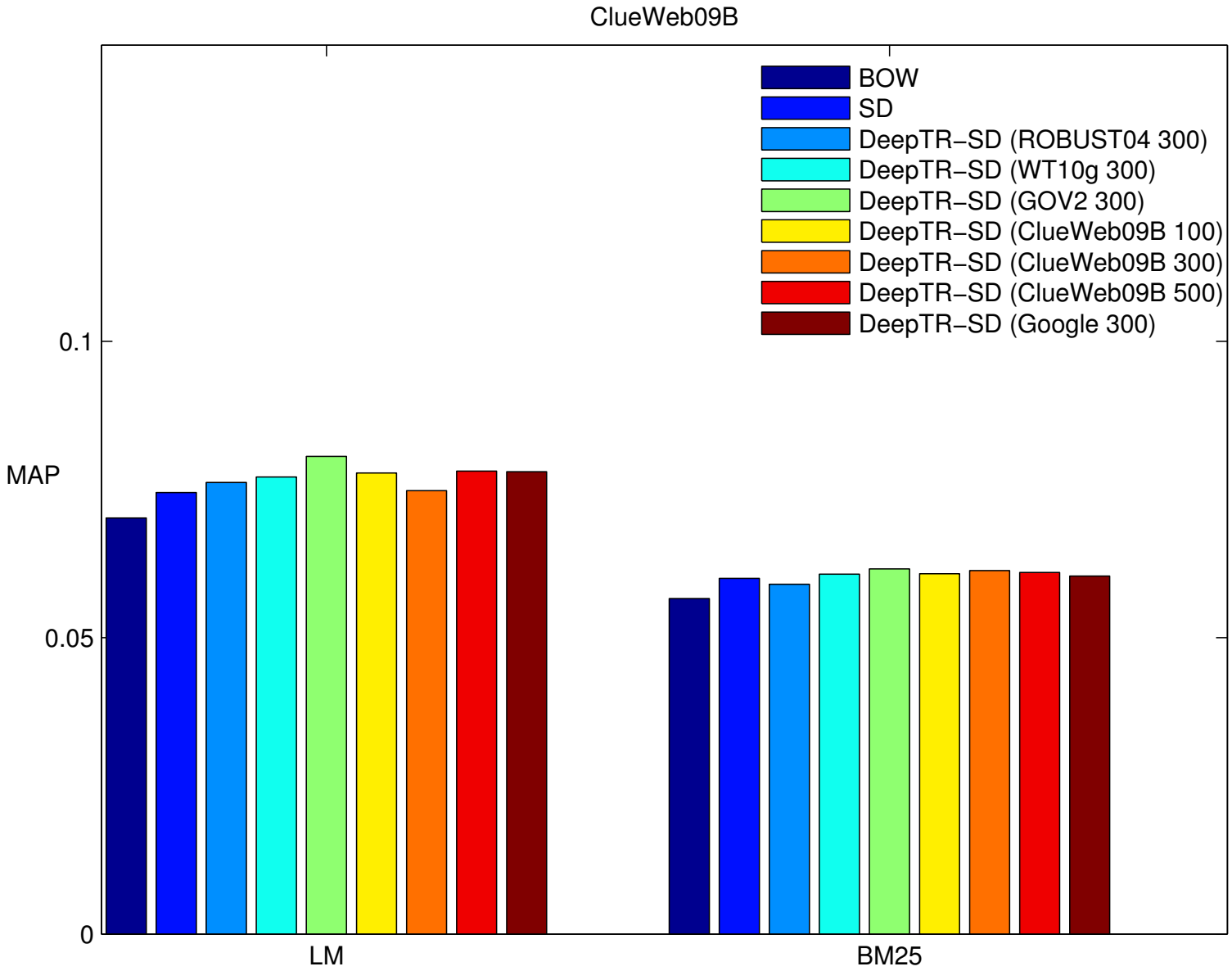
$\Rightarrow d = 100$ works slightly better

Experimental results (cont.)

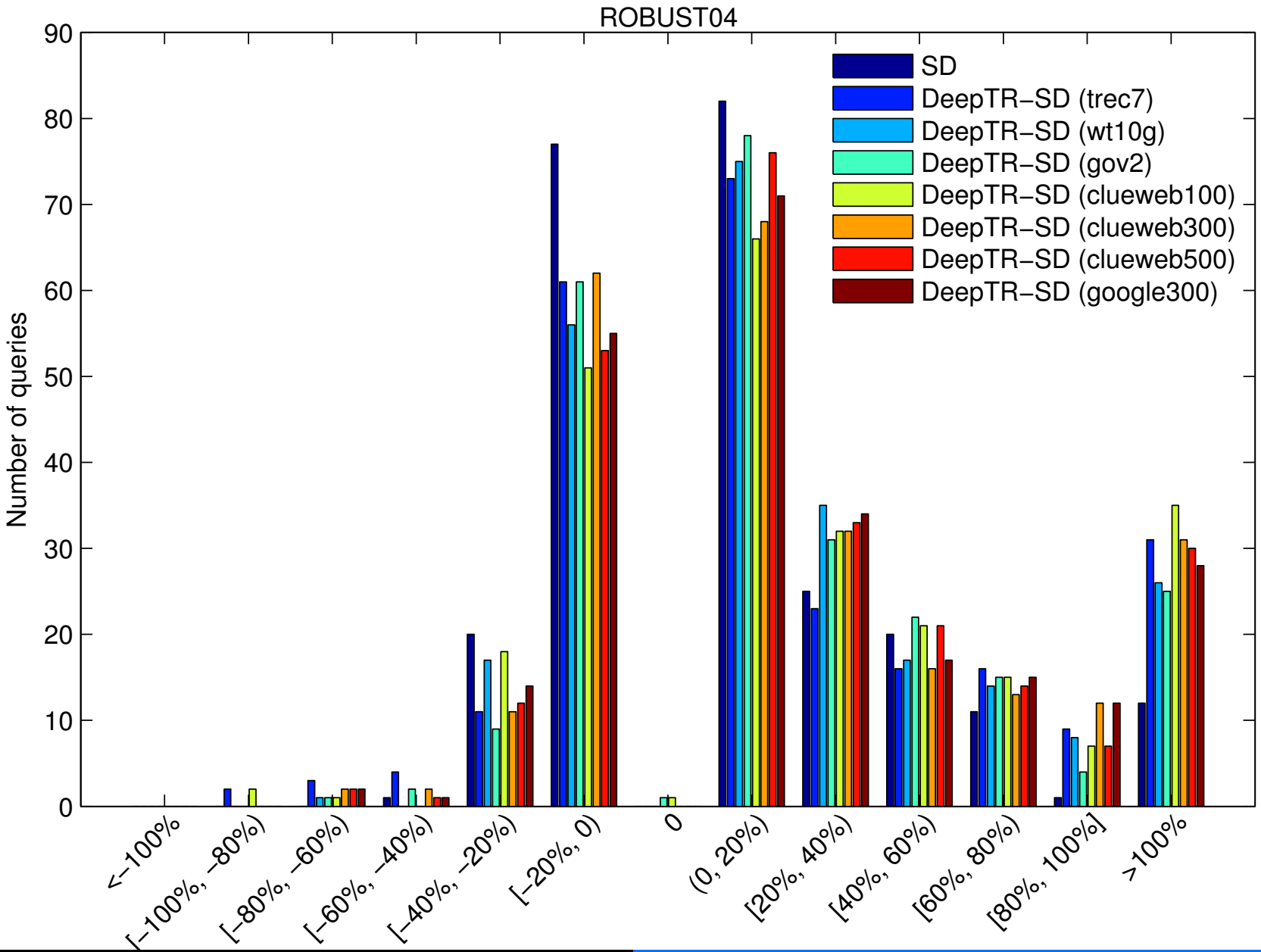


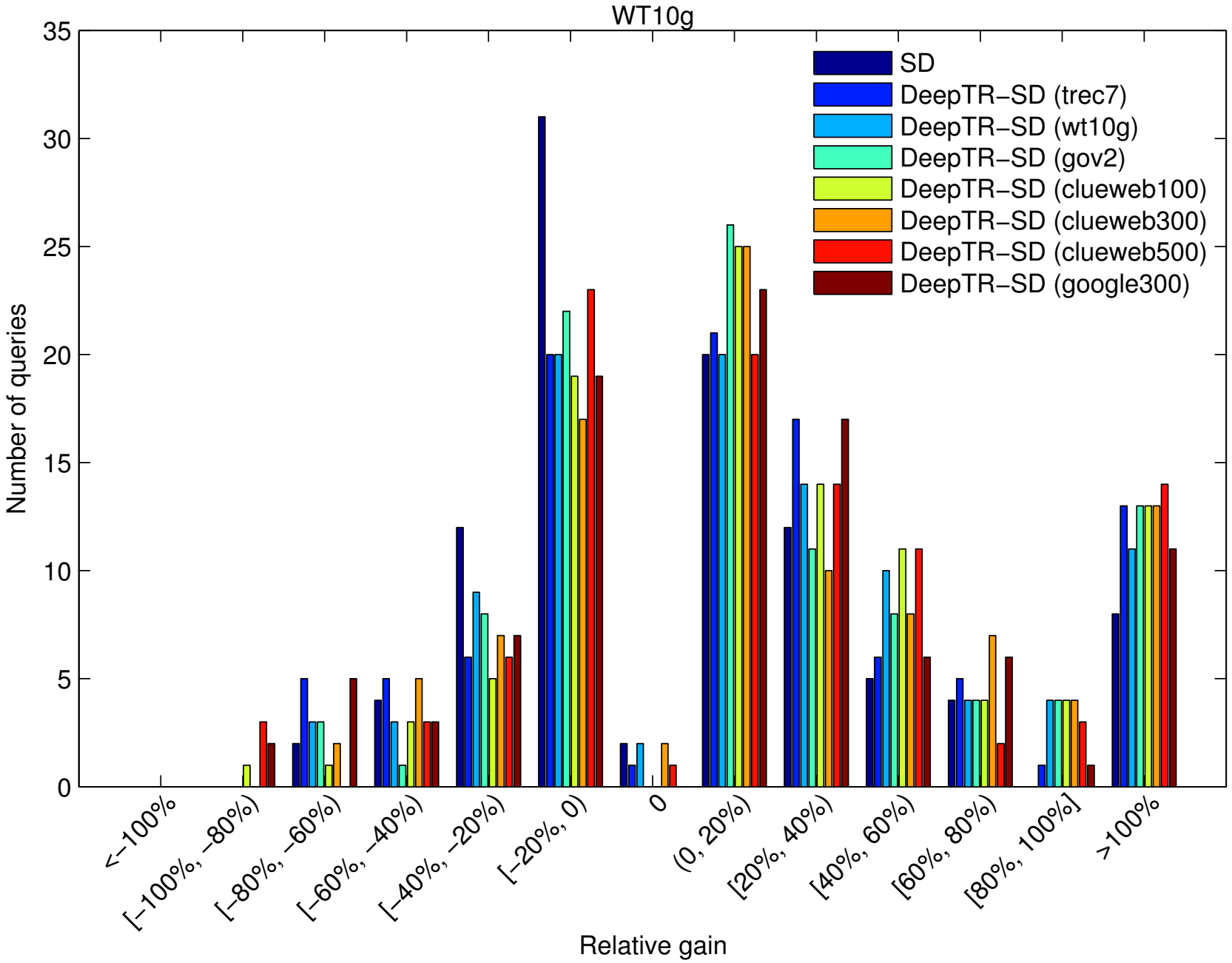

WT10g
MAP
0.4
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
LM
BM25
BOW
SD
DeepTR–SD (ROBUST04 300)
DeepTR–SD (WT10g 300)
DeepTR–SD (GOV2 300)
DeepTR–SD (ClueWeb09B 100)
DeepTR–SD (ClueWeb09B 300)
DeepTR–SD (ClueWeb09B 500)
DeepTR–SD (Google 300)

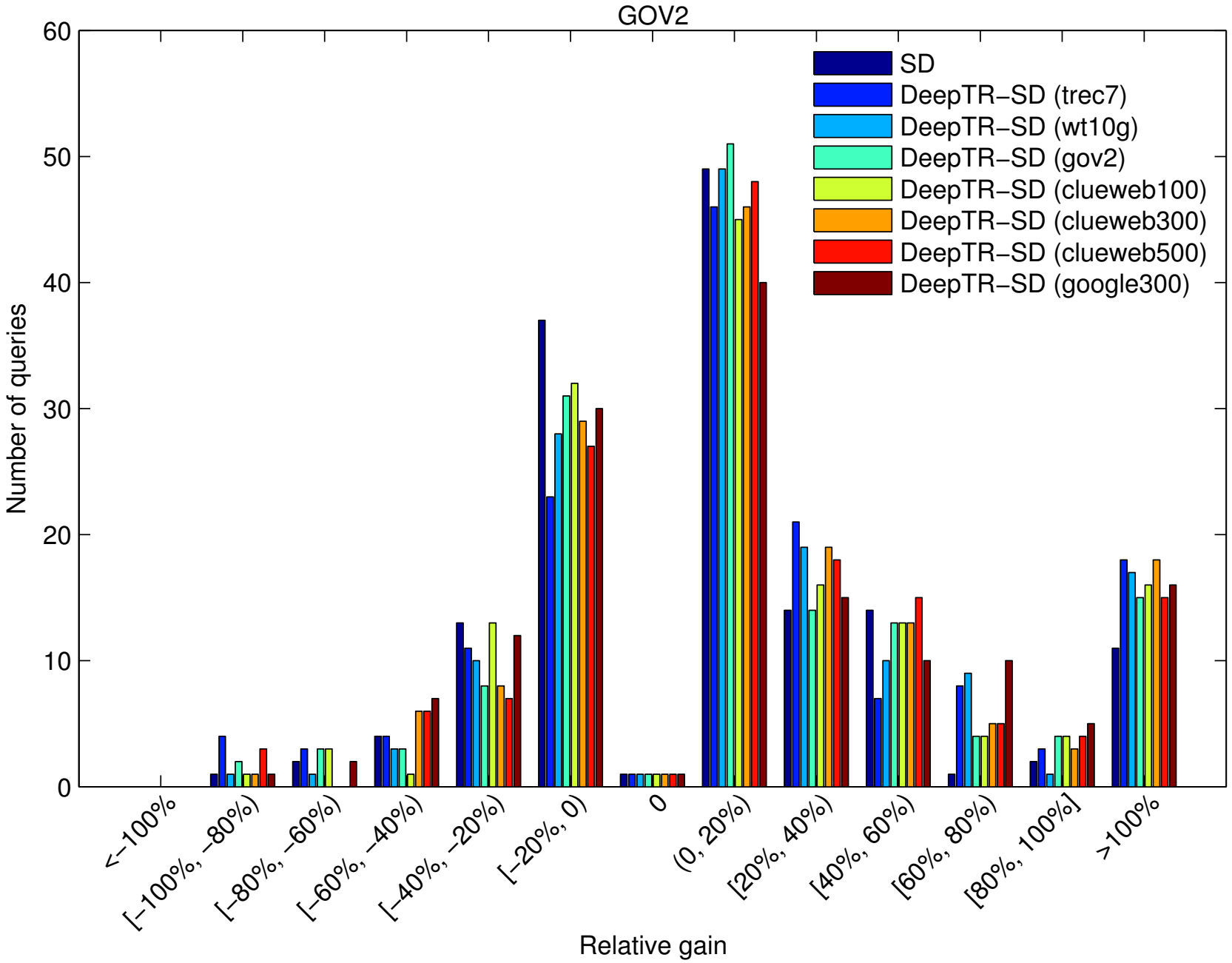

GOV2
MAP
0.4
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
LM
BM25
BOW
SD
DeepTR–SD (ROBUST04 300)
DeepTR–SD (WT10g 300)
DeepTR–SD (GOV2 300)
DeepTR–SD (ClueWeb09B 100)
DeepTR–SD (ClueWeb09B 300)
DeepTR–SD (ClueWeb09B 500)
DeepTR–SD (Google 300)

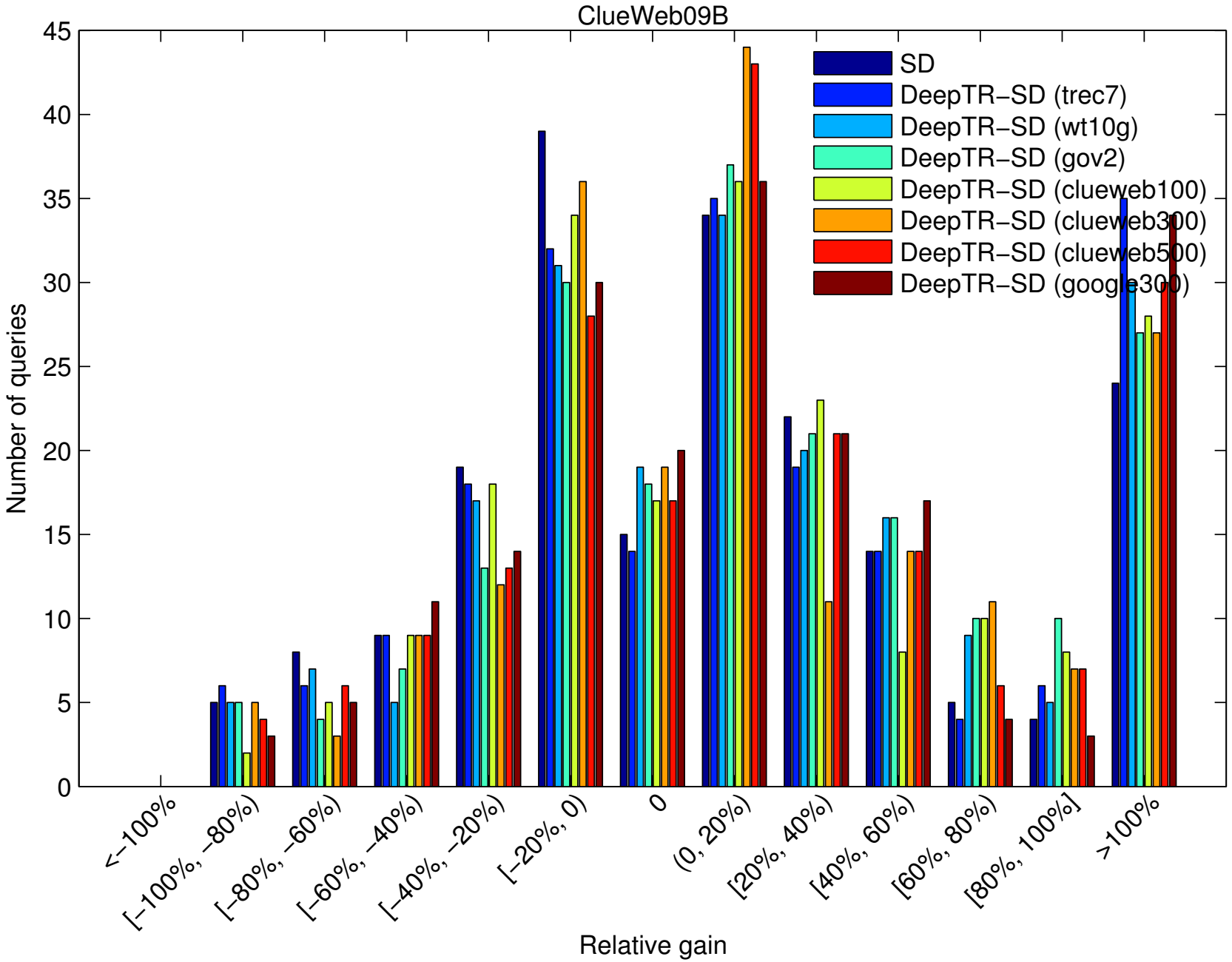

ClueWeb09B
BOW
SD
DeepTR–SD (ROBUST04 300)
DeepTR–SD (WT10g 300)
DeepTR–SD (GOV2 300)
DeepTR–SD (ClueWeb09B 100)
DeepTR–SD (ClueWeb09B 300)
DeepTR–SD (ClueWeb09B 500)
DeepTR–SD (Google 300)
MAP
0.1
0.05
0
LM
BM25

Robustness

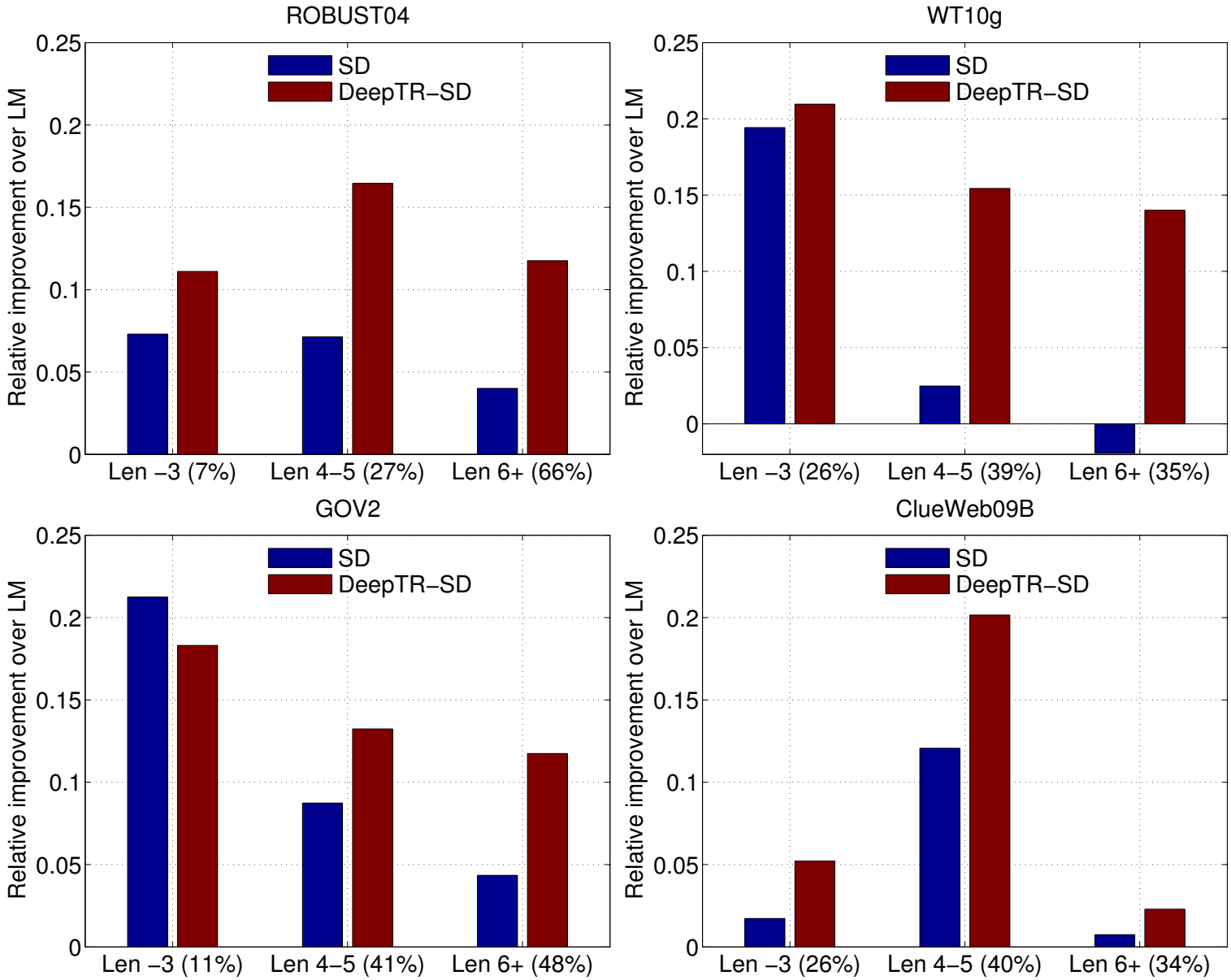

ROBUST04
Number of queries
0
10
20
30
40
50
60
70
80
90
SD
DeepTR–SD (trec7)
DeepTR–SD (wt10g)
DeepTR–SD (gov2)
DeepTR–SD (clueweb100)
DeepTR–SD (clueweb300)
DeepTR–SD (clueweb500)
DeepTR–SD (google300)
<–100%
[–100%, –80%)
[–80%, –60%)
[–60%, –40%)
[–40%, –20%)
[–20%, 0)
0
(0, 20%)
[20%, 40%)
[40%, 60%)
[60%, 80%)
[80%, 100%]
>100%


WT10g
SD
DeepTR–SD (trec7)
DeepTR–SD (wt10g)
DeepTR–SD (gov2)
DeepTR–SD (clueweb100)
DeepTR–SD (clueweb300)
DeepTR–SD (clueweb500)
DeepTR–SD (google300)
Number of queries
0
5
10
15
20
25
30
35
<−100%
[−100%, −80%)
[−80%, −60%)
[−60%, −40%)
[−40%, −20%)
[−20%, 0)
0
(0, 20%)
[20%, 40%)
[40%, 60%)
[60%, 80%)
[80%, 100%]
>100%
Relative gain


GOV2
SD
DeepTR–SD (trec7)
DeepTR–SD (wt10g)
DeepTR–SD (gov2)
DeepTR–SD (clueweb100)
DeepTR–SD (clueweb300)
DeepTR–SD (clueweb500)
DeepTR–SD (google300)
Number of queries
0
10
20
30
40
50
60
<−100%
[−100%, −80%)
[−80%, −60%)
[−60%, −40%)
[−40%, −20%)
[−20%, 0)
0
(0, 20%)
[20%, 40%)
[40%, 60%)
[60%, 80%)
[80%, 100%]
>100%
Relative gain


ClueWeb09B
Number of queries
Relative gain
SD
DeepTR–SD (trec7)
DeepTR–SD (wt10g)
DeepTR–SD (gov2)
DeepTR–SD (clueweb100)
DeepTR–SD (clueweb300)
DeepTR–SD (clueweb500)
DeepTR–SD (google300)
0
5
10
15
20
25
30
35
40
45
<−100%
[−100%, −80%)
[−80%, −60%)
[−60%, −40%)
[−40%, −20%)
[−20%, 0)
0
(0, 20%)
[20%, 40%)
[40%, 60%)
[60%, 80%)
[80%, 100%]
>100%

Query length



Graded relevance

Query Model	Language Model Retrieval			
	GOV2		ClueWeb09B	
	NDCG@20	ERR@20	NDCG@20	ERR@20
BOW	0.3732	0.1493	0.1597	0.1253
SD	0.4059	0.1607	0.1694	0.1243
DeepTR-SD (GOV2)	0.4121^{b}	0.1626^{b}	0.1743^{b}	0.1312_{s}
DeepTR-SD (ClueWeb09B)	0.4146^{b}	0.1614^{b}	0.1663	0.1255
DeepTR-SD (Google)	0.4112^{b}	0.1600	0.1698	0.1302

Query Model	BM25 Retrieval			
	GOV2		ClueWeb09B	
	NDCG@20	ERR@20	NDCG@20	ERR@20
BOW	0.3789	0.1487	0.1188	0.1075
SD	0.3940	0.1554	0.1294	0.1059
DeepTR-SD (GOV2)	0.4008_{s}^{b}	0.1590_{s}^{b}	0.1307	0.1068
DeepTR-SD (ClueWeb09B)	0.4033_{s}^{b}	0.1587_{s}^{b}	0.1305	0.1067
DeepTR-SD (Google)	0.4010_{s}^{b}	0.1586_{s}	0.1298	0.1050

b : Statistically significant difference with BOW
s : Statistically significant difference with SD

Conclusions and future work

- Novel (simple) framework for learning term weightes from features directly constructed from distributed word vectors
- Extensive experiments with two retrieval models on four test collections demonstrates the effectiveness
- Completing the table:

Type	Method	Performance	Features	Model
Unsupervised	Constant	bad	-	-
	IDF	average	-	-
Supervised (Rank measure)	LtR-methods (WSD)	good	manual, external	non-convex
Supervised (Term recall)	Zhao et al	average	manual	requires SVD
Supervised (Term recall)	Proposed work	good	automatic	convex, cheap

- Future directions include predicting term recalls for bigrams and proximity terms, other possible ways to construct feature vectors from distributed vectors, using word vectors for query expansion, theoretical justifications, etc.

Acknowledgements and QA

- Reproducibility: codes, preprocessed queries, experimental setups are available
- Contact: gzheng@cs.cmu.edu
- Thanks to SIGIR for the student travel grant
- Questions?