Learning Deep Structured Semantic Models for Web Search using Clickthrough Data

Po-Sen Huang[1], Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, Larry Heck
Microsoft Research, Redmond, WA, USA

Presented at CIKM, Oct. 2013

[1]P. Huang is with UIUC. He was an intern with MSR when this work was done.

Background of Web Search

- Traditionally, search engines retrieve web documents by matching terms in documents with those in a search query – **lexical matching**
- However, lexical matching can be suboptimal due to language discrepancy between documents and queries
  - E.g., a concept can often be expressed using different vocabularies and language styles
- Need to bridge the lexical gaps between queries and documents – **semantic matching**

Related work on semantic modeling for IR

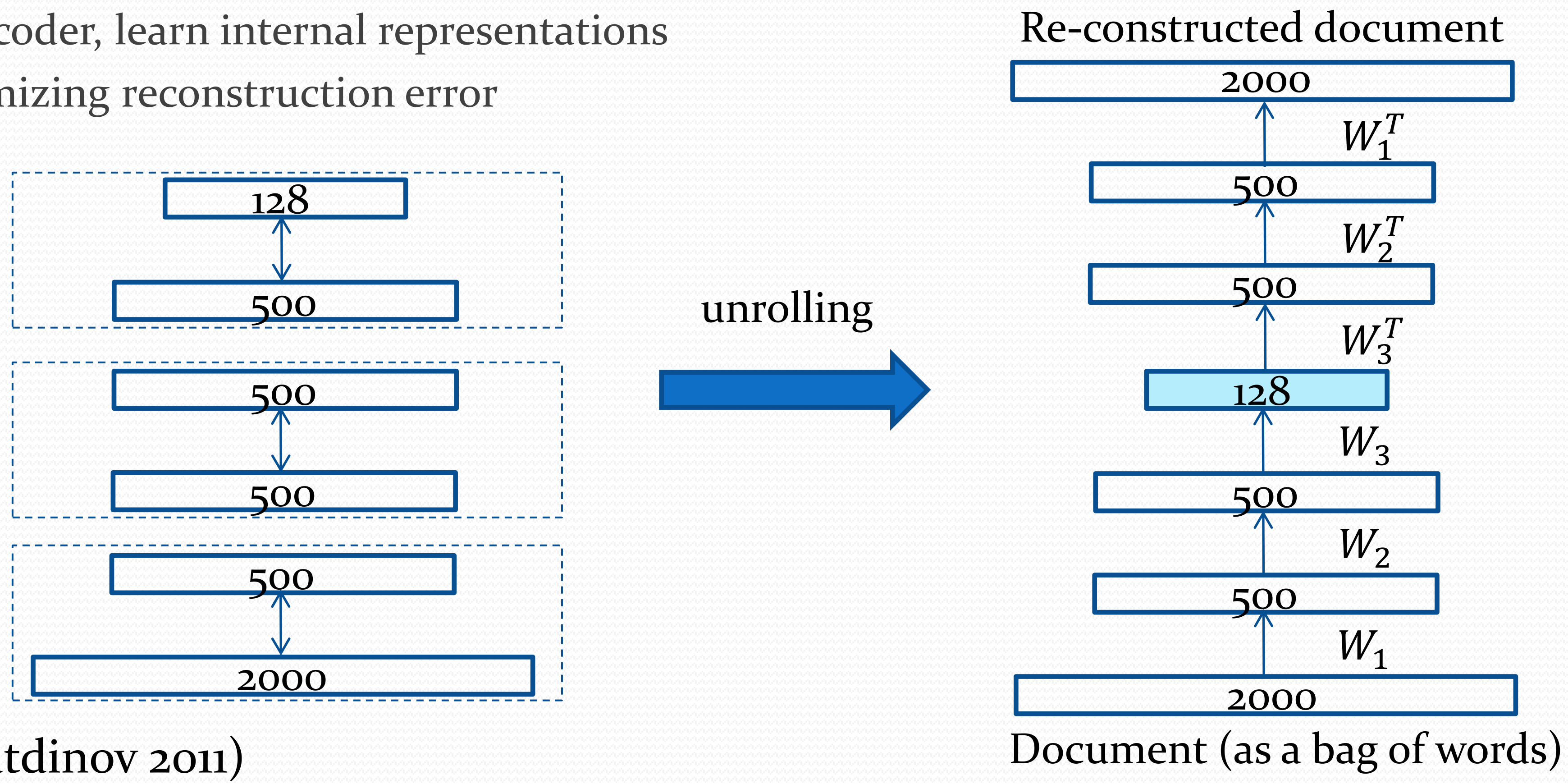
- Document retrieval based on semantic content
  - Deal with lexicon mismatch between search queries and web documents
- Early approaches
  - Latent Semantic Analysis (LSA) and its varieties (Deerwester et al., 1990)
    - LSA extracts abstract semantic content using SVD
  - Many extensions exist: PLSA, LDA, etc.
- Recent improvements:
  - Go deeper: e.g., semantic hashing (Hinton and Salakhutdinov 2011)
  - Go beyond documents: e.g., using click signals (Gao et al. 2010; Gao et al. 2011)

Previous work: Clickthrough Log based models

- State of the art document ranking approaches that use models trained on clickthrough data.
  - Oriented PCA (Diamantaras et al., 1996)
  - Word Translation Model (Gao et al. 2010)
  - Bilingual Topic Model (Gao et al. 2011)
  - Discriminative Projection Model (Yih et al. 2011; Gao et al. 2011)
- However,
  - expressive power could be limited by using linear model
  - Not scalable, model size increases rapidly along vocabulary size

Previous work: Deep auto encoder

- Training
  - Step1: RBM layer-wise pre-training, initialize weights
  - Step2: Deep auto-encoder, learn internal representations through minimizing reconstruction error

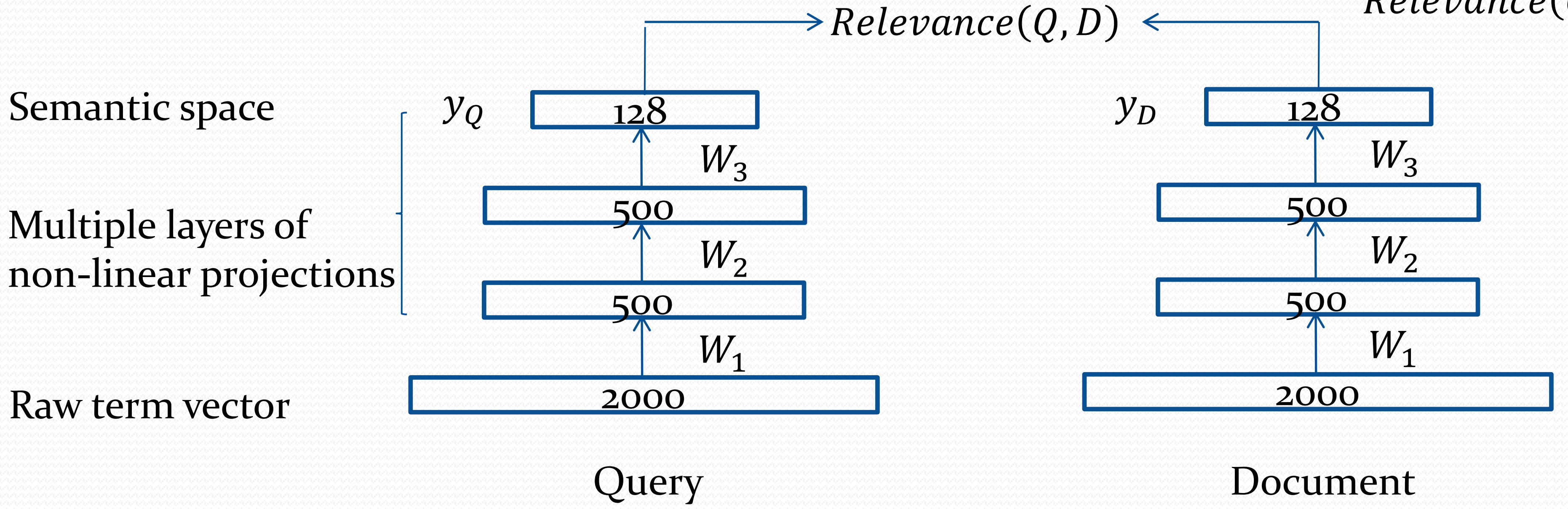


(Hinton and Salakhutdinov 2011)

Previous work: Deep auto encoder (II)

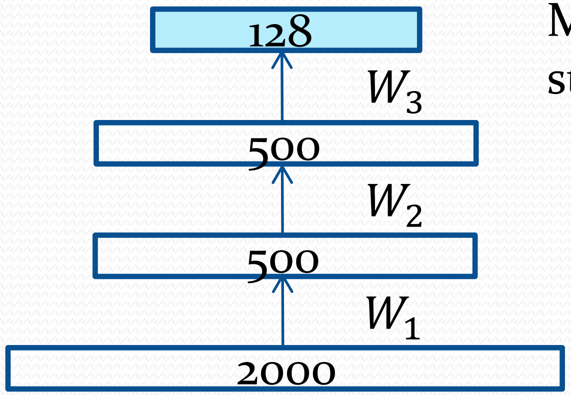
- Testing
  - Project both query and document to a common semantic space
  - Measure the relevance of Q and D in that space directly

$$Relevance(Q,D) = Cosine(y_Q, y_D)$$



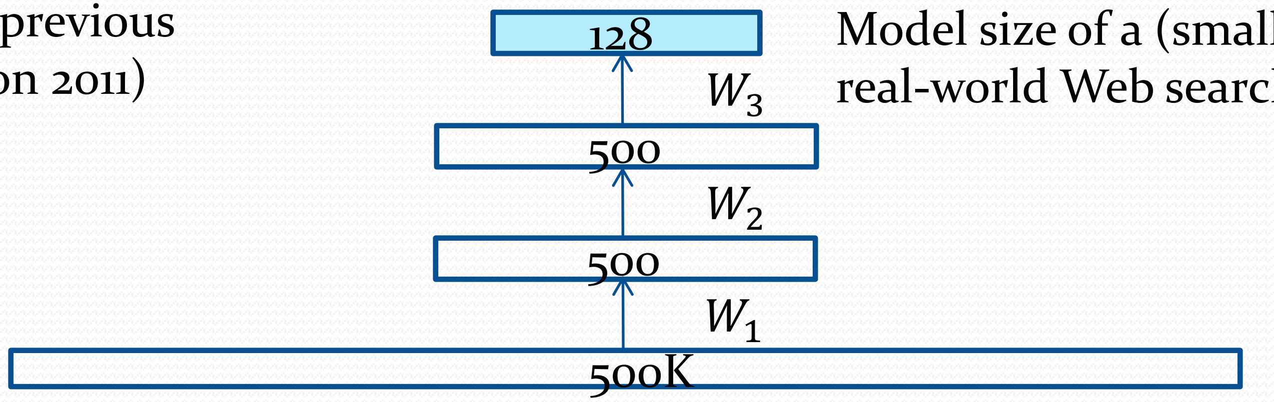
Problems of DAE

- Mismatched *learning objective*
  - Model is trained by reconstructing the document, not for relevance measure
- Lack of *scalability*
  - Model size increases rapidly along the vocabulary size



Model size in previous studies (Hinton 2011)

~1million parameters



Model size of a (small) real-world Web search task

250 million parameters

Learning semantic representations from Web and search logs

- The goal of deep semantic representation for web search
  - Map docs/queries/entities/… to a common semantic space for inference

- Our solution: Deep Structured Semantic Models (DSSM)
  - Using the ***tri-letter*** based word hashing for scalable word representation
  - Using the ***deep neural net*** to extract high-level semantic representations
  - Using the ***click signal*** to guide the learning

Tri-letter: a scale-able word representation

- Tri-letter based Word Hashing of "cat"
  - -> #cat#
  - Tri-letters: #-c-a, c-a-t, a-t-#.

- Compact representation
  - |Voc| (500K) → |TriLetter| (30K)
- Generalize to unseen words
- Robust to misspelling, inflection, etc.

$$x\,(cat) = \begin{bmatrix}0\\ \vdots \\ 1 \\ \vdots \\ 0\end{bmatrix} \leftarrow$$

The index of word *cat* in the vocabulary

$$f\,(cat) = \begin{bmatrix}0\\ \vdots \\ 1 \\ \vdots \\ 1 \\ \vdots \\ 1 \\ \vdots \\ 0\end{bmatrix}$$

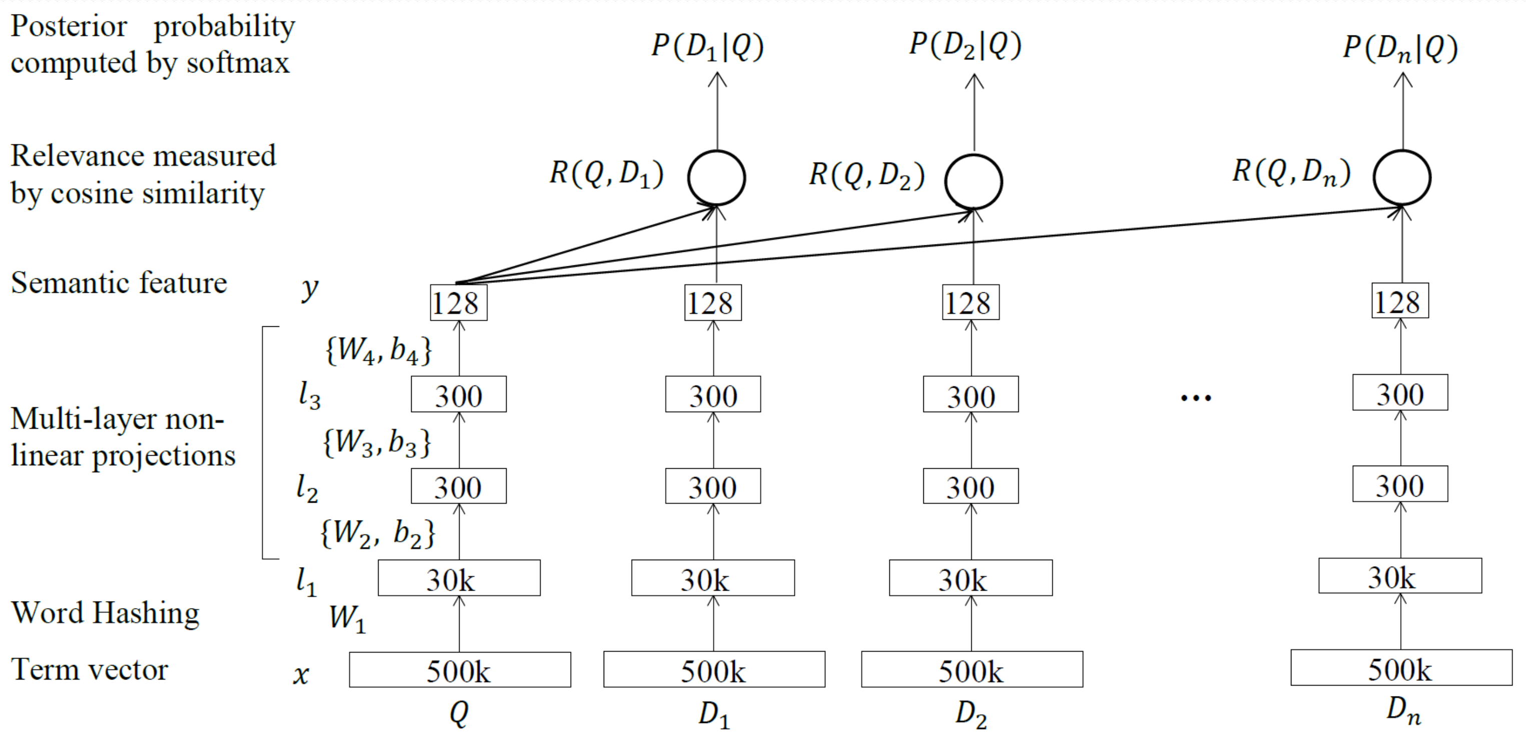
Indices of *#-c-a*, *c-a-t*, *a-t-#* in the letter-tri-gram list, respectively.

Word hashing by n-gram of letters

- Collision:
  - What if different words have the same word hashing vector?
  - Statistics
    - 22 out of 500K words collide
    - Collision Example: #bananna# <- > #bannana#

Vocabulary size	Unique tri-letter observed in voc	Number of Collisions
40K	10306	2
500K	30621	22

Deep Structured Semantic Model (DSSM)



Use **deep neural nets** for semantic representation extraction

Use **tri-letter** based word hashing to handle any unseen words

Maximize the **cosine similarity** between the query and the clicked doc

[Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, Larry Heck, “Learning Deep Structured Semantic Models for Web Search using Clickthrough Data,” in CIKM 2013]

Training DSSM

- Optimization: SGD (w/ minibatch)
- Objective: Cosine loss defined on the clickthrough data
  - For each query Q, there is a set of documents $\boldsymbol{D}$
    - $\boldsymbol{D} = \{D^+, D_1^-, \dots, D_N^-\}$ includes the clicked doc D^+, and a set of unclicked docs collected via sampling
    - $R(Q,D) = Cosine(y_D, y_Q)$
    - $P(D|Q) = \frac{\exp(\gamma R(Q,D))}{\sum_{D' \in \boldsymbol{D}} \exp(\gamma R(Q,D'))}$
  - $\text{loss}(Q, \boldsymbol{D}) = -log P(D^+|Q)$

Implementation Details

- Select parameters based on cross validation
- Randomly choose 4 competitors (similar performance as selecting based on TF-IDF ranking)
- We fixed the architecture to be
  - TriLetter-300-300-128
- Tanh() as the activation function
- Random initialization – pretraining does not make much difference
- Use stochastic gradient descent to optimize the training objective
- Control learning rate

NDCG results on a real-world Web search task

Models	NDCG@1	NDCG@3	NDCG@10
BM25	30.8	37.3	45.5
Previous Shallow/Deep Semantic Models, trained on doc collection (unsupervised)			
LSA (Deerwester et al., 1990)	29.8	37.2	45.5
PLSA (Hofmann 1999)	29.5	37.1	45.6
Deep Auto-Encoder (Hinton et al., 2011)	30.6	37.4	45.6
Previous Semantic Models trained on click logs (supervised)			
DPM (w/ S2Net (Yih et al., 2011))	32.9	40.1	47.9
Word Translation Model (Gao et al, 2010)	33.2	40.0	47.8
Bilingual Topic Model (Gao et al., 2011)	33.7	40.3	48.0
Our deep structured semantic model trained on click logs (supervised)			
DSSM (this work)	36.2	42.5	49.8

(Please refer to our CIKM13 paper for more details)

Visualization

- $\hat{x} = argmax_x\ (h(x))$

Car	*Holiday*	*Video*	*Hunting*	*System*
automotive	happy	youtube	bear	systems
wheels	lyrics	videos	hunting	protect
cars	musical	dvd	texas	platform
auto	halloween	downloads	colorado	efficiency
car	eastern	movie	hunter	oems
vehicle	festival	cd	tucson	systems32

Table 1: Examples of words with high activation at the same nodes.

Summary

- Proposed a deep structured semantic model (DSSM) for web search

  Tri-letter based word representation

  deep neural net based semantic model

  Cosine-similarity based loss function defined on click log

- Significant gains over previous approaches
  - 5 pt NDCG gain compared with BM25
  - 3 pt gain compare with state of the art latent semantic models (BLTM, MT, DPM, etc.)

References

- Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., and Kuksa, P., 2011. Natural language processing (almost) from scratch. *JMLR*, vol. 12.
- Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T., and Harshman, R. 1990. Indexing by latent semantic analysis. *JASIS*.
- Deng, L., He, X. and Gao, J. 2013. Deep stacking networks for information retrieval. In *ICASSP.*
- Deng, L., Yu, D. and Platt, J. 2012 Scalable stacking and learning for building deep architectures In *ICASSP.*
- Gao, J., He, X., and Nie, J-Y. 2010. Clickthrough-based translation models for web search: from word models to phrase models. In *CIKM*.
- Gao, J., Toutanova, K., Yih., W-T. 2011. Clickthrough-based latent semantic models for web search. In *SIGIR*.
- He, X. and Deng, L. 2013. Speech-centric information processing: an optimization-oriented approach. *Proc of the IEEE*.
- Hinton, G. and Salakhutdinov R. 2011. Discovering Binary Codes for Fast Document Retrieval by Learning Deep Generative Models. Topics in Cognitive Science, Vol 3, pp 74-91.
- Hofmann, T. 1999. Probabilistic latent semantic indexing. In *SIGIR*.
- Huang, P-S., He, X., Gao, J., Deng, L., Acero, A. and Heck, L. 2013. Learning deep structured semantic models for web search using clickthrough data. In *CIKM 2013*.
- Hutchinson, B., Deng, L. and Yu, D. 2013. Tensor deep stacking networks *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Word hashing by n-gram of letters

- Collision:
  - What if different words have the same word hashing vector?
  - Statistics
    - 22 out of 500K words collide
    - Collision Example: #bananna# <- > #bannana#

Vocabulary	Type	Unique Key	Collision
40K	Bigram	1107	18
	Trigram	10306	2
500K	Bigram	1607	1192
	Trigram	30621	22