Learning Deep Structured Semantic Models for Web Search using Clickthrough Data

Po-Sen Huang et al. CIKM'13

이현성

Index

- Motivation
- Model description
  - Overview
  - Word hashing
  - MLP Layer
  - Ranking documents and learning
- Experiment and discussion
- Extension of DSSM
- Conclusion

Motivation

Motivation

- Information Retrieval (Document Searching and Retrieval)
  - Main category in Data Mining (almost independent discipline)

- Example
  - Documents can be any length from one word to thousands.
  - A query is a special type of document

q := ‘ant ant bee’

d_1 := ‘dog bee dog hog dog ant dog’

d_2 := ‘cat gnu dog eel fox’

Motivation

- Problem definition
  - Let d to be a documents and D to be set of all documents
    Let q to be a query.

    q, d are sentences, or text
    You can consider text either sequence, or multiset.

  - For given query q
    Return a set of relevant documents of size k D_k
    Where $D_k := \{d \in D \mid sim(d, q)\ is\ high\ enough\}$

  - You need to define similarity measure between a document d and a query q
    - thus rank documents given query q

On calculating $sim(d, q)$

- We need a model f such that

Model $f : d \rightarrow \mathbb{R}^d$

Then we can calculate similarity of non-numeric entities in transformed space

$$sim(d_1, d_2) := cos\big(f(d_1), f(d_2)\big) = \frac{f(d_1)^T f(d_2)}{|f(d_1)||(f(d_2)|}$$

This model f have many names

Hashing, Embedding, Mapping, Transformation, Projection

On calculating $sim(d, q)$

- Traditional methods are limited
  - TF-IDF, BM25 (counting based vector mapping)
    - They fail to catch semantic information.
      Same theme can be represented in different words.
    - *기본적으로, 두 document에서 같은 단어가 자주 나오면 비슷하다고 판단하는 방법*
  - LSA(PCA), LDA (unsupervised statistical inference)
    - These models are trained in an unsupervised manner.
      Their Objective function are only loosely coupled with calculating $sim(d, q)$.
    - *이 종류의 모델이 optimize하는 goal이랑 문서 간의 similarity는 별로 상관이 없는 것으로 알려져 있다.*
      (그래서 생각보다 성능이 별로)

https://janav.wordpress.com/2013/10/27/tf-idf-and-cosine-similarity/
this page explains tf-idf in depth

https://www.youtube.com/watch?v=NcC1XyD5dvA
Very detailed explanation on Latent Semantic Analysis (PCA)

https://www.youtube.com/watch?v=FkckgwMHP2s
David Blei (who created LDA) explains LDA and other probabilistic topic models(difficult)

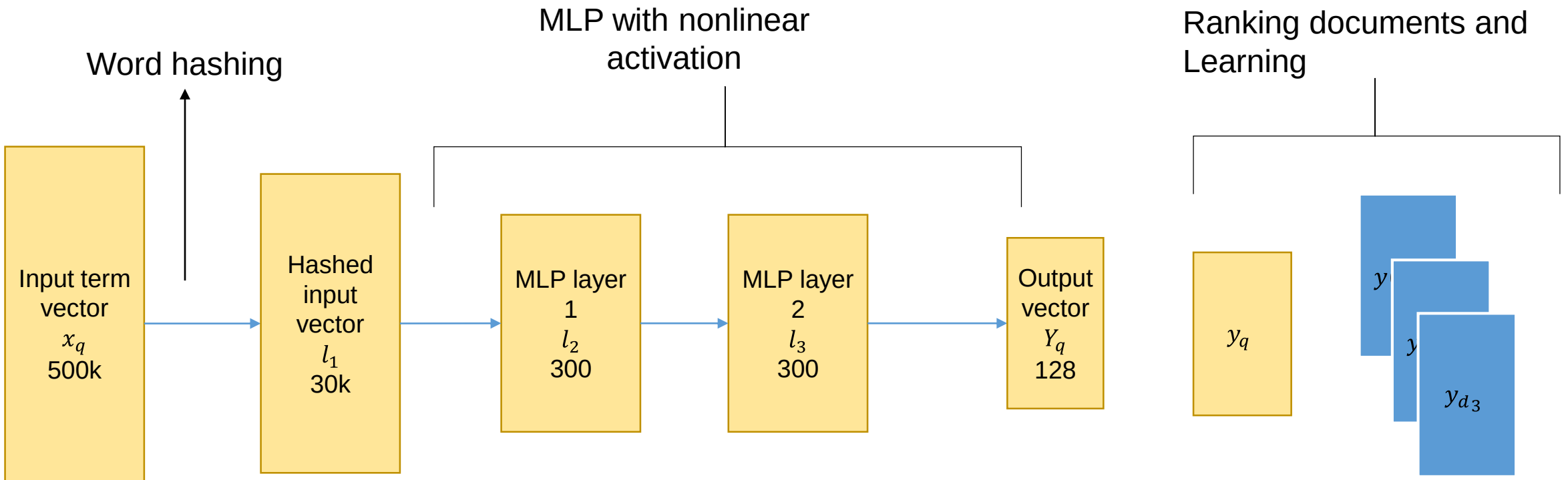
Cheng Xhai SIGIR '17 tutorial (교수님이 메일로 공유함)

Semantic feature를 이해할 수 있으면서
Supervised learning인 text similarity model을 한번 만들어 보자!

(기존에 그런 모델이 없는 건 아닌데... 계산도 적게 하는 모델이면 더 좋다)

Model Description

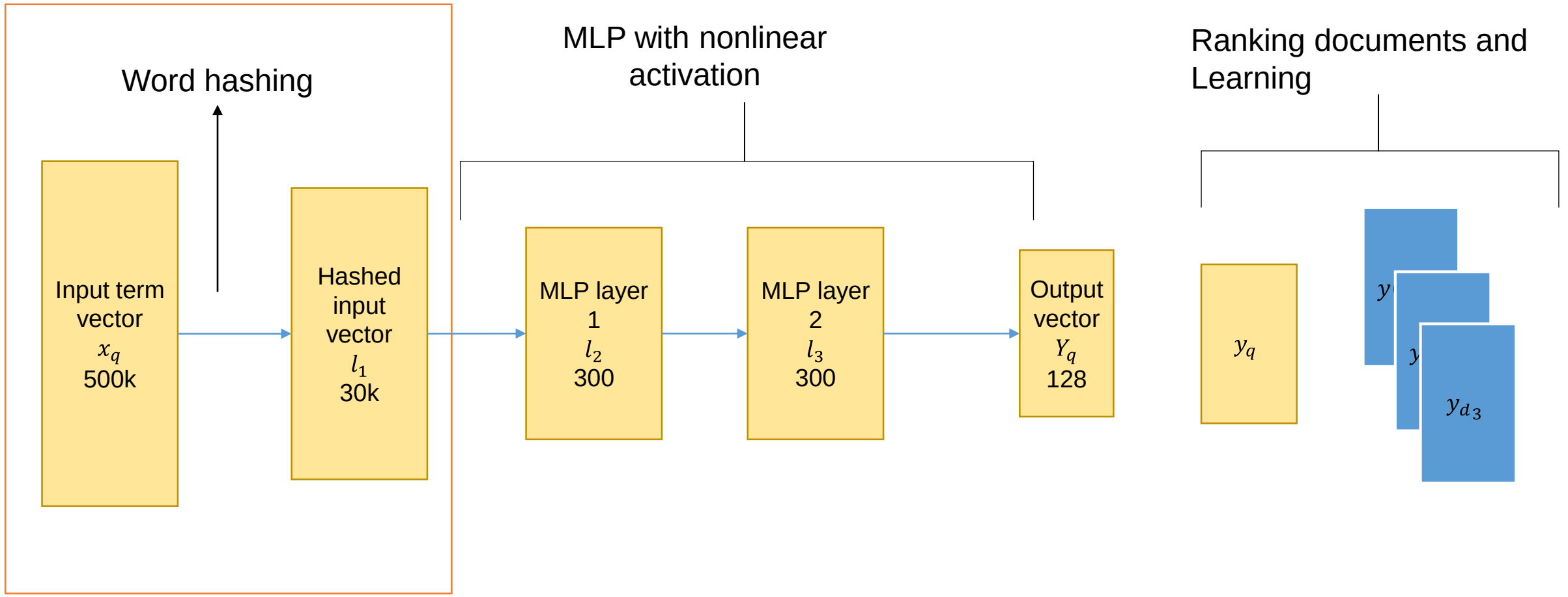
The DSSM Model is made up to three parts



Functional representation of the DSSM Model

$$M(q;\Theta)$$

$$M : q \in D \rightarrow y \in \mathbb{R}^D$$


Word hashing
MLP with nonlinear activation
Ranking documents and Learning
Input term vector
x_q
500k
Hashed input vector
l_1
30k
MLP layer 1
l_2
300
MLP layer 2
l_3
300
Output vector
Y_q
128
y_q
y
y
y_{d_3}

Letter-n-gram hashing

- The input of DSSM model are queries and documents mapped on so-called *n-gram* spaces instead of traditional word spaces.

example)

'good', -> '#good#'

Then it is mapped into a list of trigrams:

  - *#go, goo, ood, od#*

- As the number of characters are limited, the number of possible n-grams are limited as well, which is much smaller compared to number of words available.

Letter-n-gram hashing

Term frequency

d_1 := “data mining”
d_2 := “data is good data”

Term Freq	Data	Mining	Good	Is	Any	Other	Words
d_1	1	1	0	0	0	0	0
d_2	2	0	1	1	0	0	0

Term frequency input vector x

Word trigram frequency

d_1 := "#da dat ata ta# #mi min ini nin ing ng#"

d_2 := "#da dat ata ta# #is is# #go goo ood od# #da dat ata ta# #is"

token Freq	#da	Dat	Ata	Ta#	Other	Words
d_1	1	1	1	1	…	…
d_2	2	2	2	2	…	…

Word trigram vector l_1

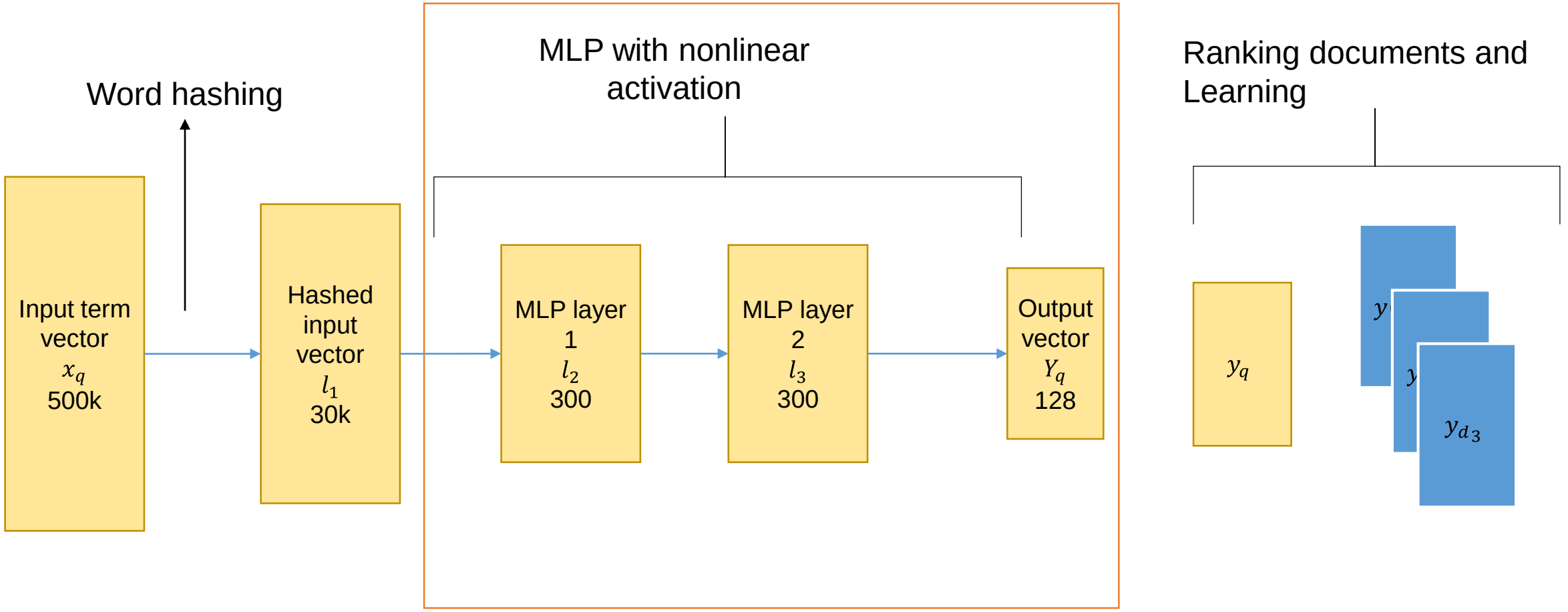
Term-frequency vector로 정의된 document를 trigram space에 매핑

Letter-n-gram hashing enables training large-scale dataset

- The number of English words are at least 500k.
- But, the number of trigrams $\leq 29^3 \cong 30k$

- Two different words could have same n-gram letter vector representation.
  - (but in trigram case, collision is neglible)

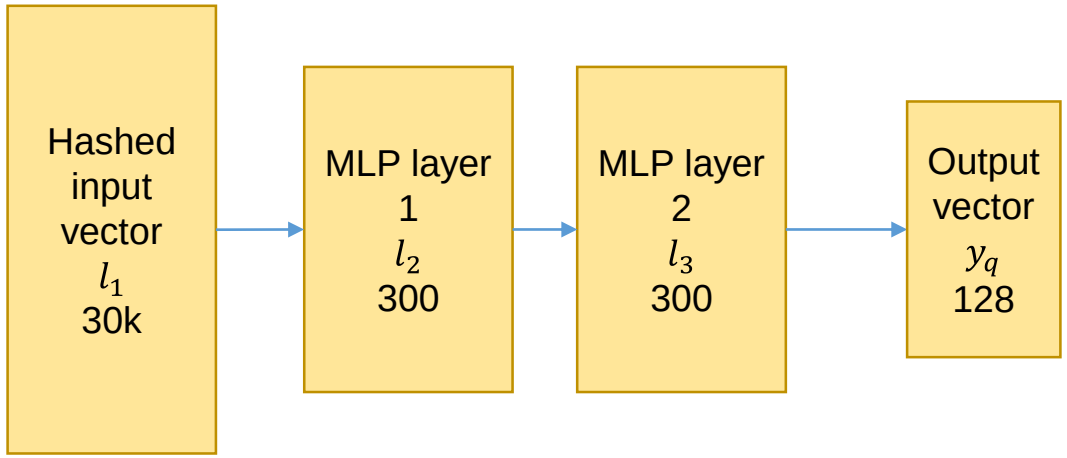
	Letter-trigram	
Word size	Token size	Collision
40k	10306	2
500k	30621	22


Word hashing
MLP with nonlinear activation
Ranking documents and Learning
Input term vector
x_q
500k
Hashed input vector
l_1
30k
MLP layer 1
l_2
300
MLP layer 2
l_3
300
Output vector
Y_q
128
y_q
y
y
y_{d_3}

Multilayer with nonlinear activation

- I assume that all of us are familiar with the concept of multilayer perceptron.

$$l_2 = f(W_2 l_1 + b_2)$$
$$l_3 = f(W_3 l_2 + b_3)$$
$$y = f(W_o l_3 + b_o)$$



Where $f(x) := \tanh(x)$

$$= \frac{2}{1 + \exp(-2x)} - 1$$

$$sim(q, d) := cosine(y_q, d_q)$$

$$= \frac{y_q^T y_d}{|y_q|\,|y_d|}$$

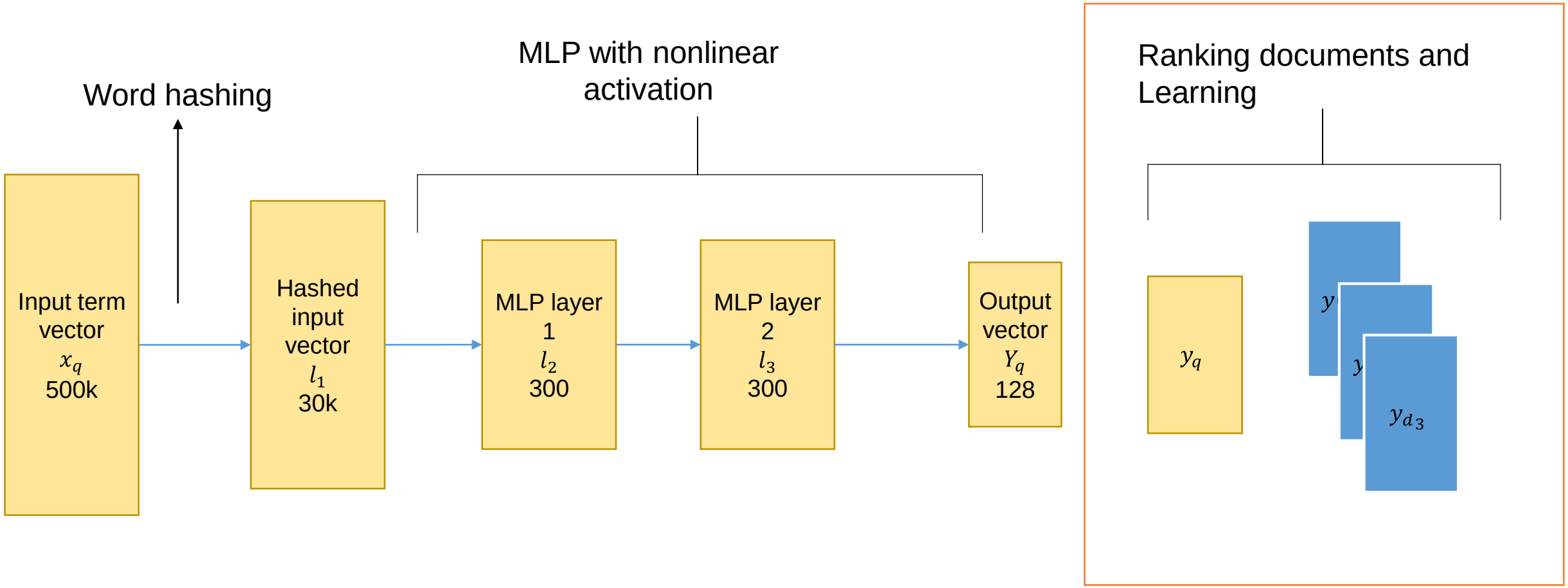
The model embeds a document into a vector

- Now we know how the model M calculates similarity between two document q, d

$$sim(q,d) \coloneqq cosine(y_q, d_q)$$
$$= \frac{y_q^T y_d}{|y_q|\,|y_d|}$$

Where $y_q = \mathrm{M(q)}, \mathrm{y_d} = \mathrm{M(d)}$

- Then, how to train this model?


Word hashing
MLP with nonlinear activation
Ranking documents and Learning
Input term vector
x_q
500k
Hashed input vector
l_1
30k
MLP layer 1
l_2
300
MLP layer 2
l_3
300
Output vector
Y_q
128
y_q
y
y
y_{d_3}

Training the model

- For each query q, We know answer document(user actually clicked) d^+ from clickthrough data.

$P(d^+|q)$:= the prob. that the document d^+ to be relevant given the query q

So, if we observe query-answer pair (q, d^+),
update $P(d^+|q)$ to be higher.

$$P(d^+|q) := \frac{\exp\left(\gamma sim(q, d^+)\right)}{\sum_{d' \in D} \exp\left(\gamma sim(q, d')\right)}$$

γ is the smoothing factor in the softmax function

In practice, for each query-answer pair, we approximate D by including d^+ and four randomly selected unclicked document.

$$\overline{D} = \{d^+, d_1, d_2, d_3, d_4\}$$

Training the model

Assuming i.i.d, our likelihood for given dataset is defined as

$$L(\Lambda) = \Pi_{(q,d^+)} P(d^+|q)$$

Maximizing likelihood function $L(\Lambda)$ is equivalent to minimizing following Negative log-likelihood

$$nll(\Lambda) \coloneqq -\log\big(L(\Lambda)\big) = -\sum_{(q,d^+)} \log\big(P(d^+|q)\big)$$

$$obj(X|\Lambda) \coloneqq argmin_{\Lambda}(nll)$$

Connection between loss function and similarity

$$nll(\Lambda) = -\log\big(P(d^+|q)\big)$$
$$= -\log\left(\frac{\exp\big(\gamma sim(q,d^+)\big)}{\sum_{d'\in D}\exp\big(\gamma sim(q,d')\big)}\right) = \log\left(\frac{\sum_{d'\in D}\exp\big(\gamma sim(q,d')\big)}{\exp\big(\gamma sim(q,d^+)\big)}\right)$$
$$= \log\left(\frac{\exp\big(\gamma sim(q,d^+)\big) + \sum_{d'\in D-\{d^+\}}\exp\big(\gamma sim(q,d')\big)}{\exp\big(\gamma sim(q,d^+)\big)}\right)$$
$$= \log\left(1 + \sum_{d'\in D-\{d^+\}}\frac{\exp\big(\gamma sim(q,d')\big)}{\exp\big(\gamma sim(q,d^+)\big)}\right)$$

Query q와 relevant한 document d^+간의 similarlity $sim(q,d^+)$는 높이고,

Irrelevant한 document d'와 query q 사이의 similarity $sim(q,d^-)$는 낮춘다.

Experiments

Dataset and evaluation protocol

Large-scale evaluation dataset	
Source	One-year query log of a commercial search engine (maybe Bing)
# queries	16,510 (on average, each query is associated 15 relevant documents)
Relevance scale	[0,1,2,3,4] (0 means not relevant, 4 means most relevant
Training	Binary information (whether user clicked document or not) [0,1] It is not directly related with relevance scale. Used (approximately 100m) sampled query-document pairs whose documents are popular and rich click information
Preprocessing	All words are white-space tokenized and lowercased. No stemming is performed.
Evaluation method	2-fold cross validation : first set and second set (1) Use first set as training, and find best parameter using second set. (2) Use second set as training, and evaluate test score using first set. (3) Evaluation results are mean value of result (1) and (2)
Ranking Metric	NDCG truncated at 1,3, and 10

Baselines

- Counting based relevance models
  - TF-IDF, BM-25
- Unsupervised probabilistic topic models or ML models
  - LSA, PLSA, DAE, BLTM-PR
- Supervised ML models
  - WTM, DPM
- I omit detailed explanation of these models.
  - I think each model itself is not much important.
  - I think what is of importance is supposed model outperforms various types of models

Overall results

Models	NDCG@1	NDCG@3	NDCG@10
TF-IDF	0.319	0.382	0.462
BM25	0.308	0.373	0.455
LSA	0.298	0.372	0.455
PLSA	0.295	0.371	0.456
DAE	0.310	0.377	0.459
BLTM	0.337	0.403	0.480
WTM	0.332	0.400	0.478
DPM	0.329	0.401	0.479
DSSM w.o word hashing	0.342	0.410	0.486
DSSM with no hidden layer	0.357	0.422	0.495
DSSM with two hidden layer	***0.362***	***0.425***	***0.498***

Green : Counting based relevance models
Yellow : Unsupervised probabilistic topic models or ML models
Gray : Supervised ML models
Blue : Proposed Models

Extension of DSSM

Following slides are from

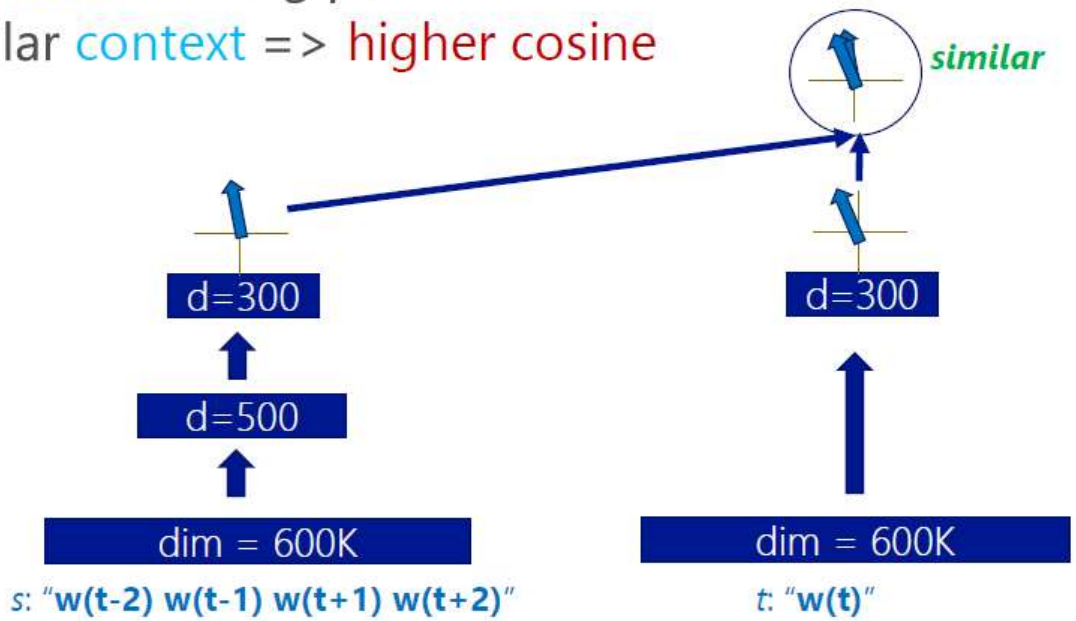
Deep Learning for Natural Language Processing: Theory and Practice (Tutorial), He and Gao et al. CIKM '14.

DSSM learns many kinds of semantic similarities between two entities *X* and *Y*

Tasks	*X*	*Y*
Web search	***Search query***	***Web documents***
Ad selection	***Search query***	***Ad keywords***
Entity ranking	***Mention (highlighted)***	***Entities***
Recommendation	***Doc in reading***	***Interesting things in doc or other docs***
Machine translation	***Sentence in language A***	***Translations in language B***
Nature User Interface	*Command (text/speech)*	*Action*
Summarization	*Document*	*Summary*
Query rewriting	*Query*	*Rewrite*
Image retrieval	*Text string*	*Images*
...	...	...

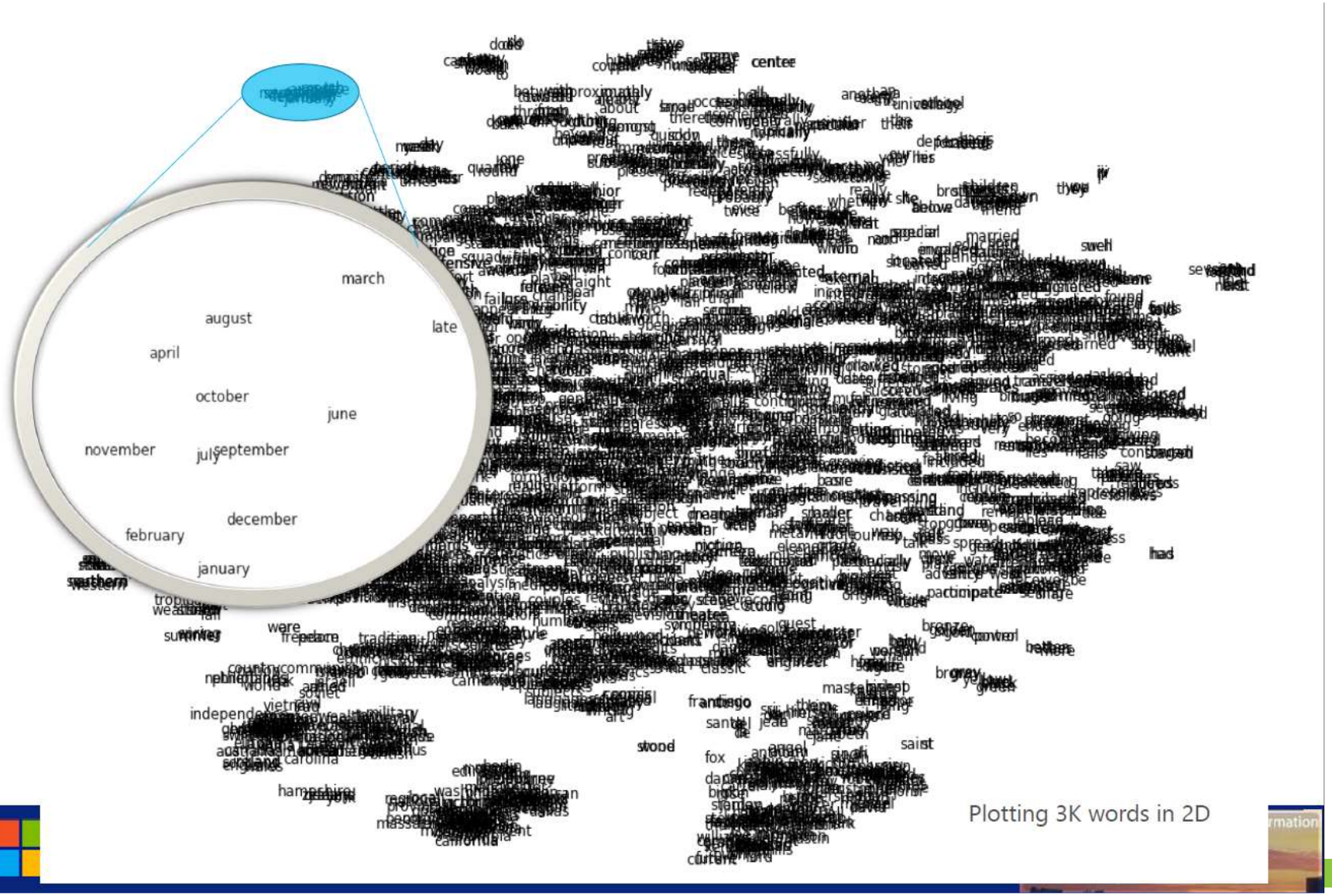
DSSM: learning words' meaning

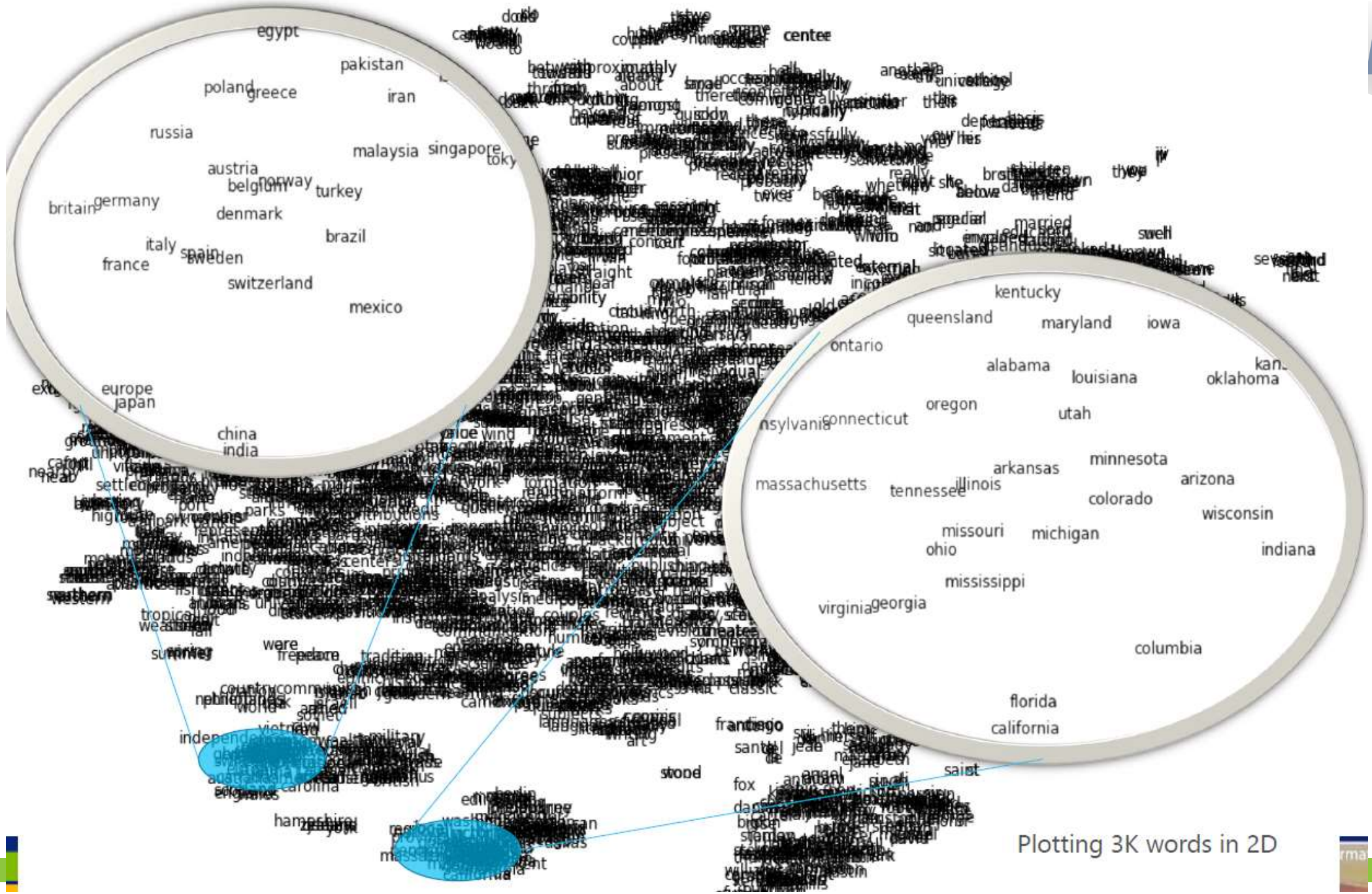
- Learn a word's semantic meaning by means of its neighbors (context)
  - Construct context <-> word training pair for DSSM
  - Similar words with similar context => higher cosine
- Training Condition:
  - 600K vocabulary size
  - 1B words from Wikipedia
  - 300-dimentional vector



You shall know a word by the company it keeps
(J. R. Firth 1957: 11)

[Song, He, Gao, Deng, Shen, 2014]


march
august
late
april
october
june
november
july
september
december
february
january
Plotting 3K words in 2D



Plotting 3K words in 2D

Evaluation on the word analogy task

The dataset contains 19,544 word analogy questions:

Semantic questions, e.g.,: "Athens is to Greece as Berlin is to ?"

Syntactic questions, e.g.,: "dance is to dancing as fly is to ?"

Model	Dim	Size	Accuracy Avg.(sem+syn)
SG	300	1B	61.0%
CBOW	300	1.6B	36.1%
vLBL	300	1.5B	60.0%
ivLBL	300	1.5B	64.0%
GloVe	300	1.6B	70.3%
DSSM	**300**	**1B**	**71.4%**

(i)vLBL results are from (Mnih et al., 2013); skip-gram (SG) and CBOW results are from (Mikolov et al., 2013a,b); GloVe are from (Pennington, Socher, and Manning, 2014)

Conclusion

- embed two entities into same semantic space and find similarity of two entities
  - equivalent content-based recommendation

- not very complicated, but show significant improvements in several areas

- 느낀점 :
  최근에 NLP에서 Sequential modelling(RNN)이 대세가 된 것 같았는데
  Bag of words 모델도 좋은 성능을 보여줘서 신기했다.