SIGIR 2018 Tutorial
July 8, 2018
Ann Arbor, USA

Deep Learning for Matching in Search and Recommendation

Jun Xu
Chinese Academy of Sciences

Xiangnan He
National University of Singapore

Hang Li
Bytedance AI Lab

Slides: http://comp.nus.edu.sg/~xiangnan/sigir18-deep.pdf

Outline of Tutorial

- Unified View of Matching in Search and Recommendation
- Part 1: Traditional Approaches to Matching
- Part 2: Deep Learning Approaches to Matching
- Summary

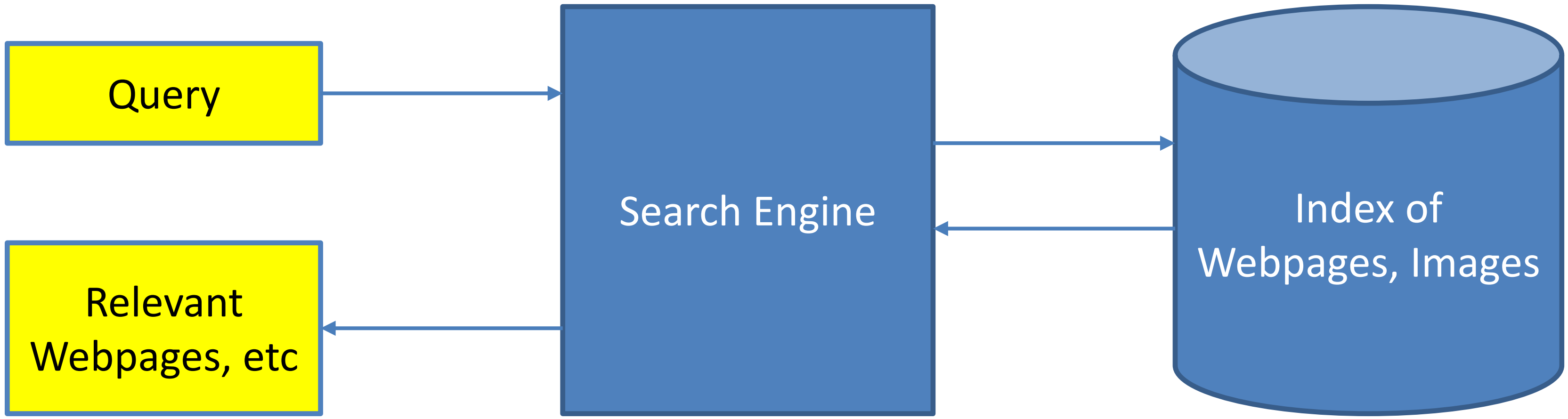
Slides: http://comp.nus.edu.sg/~xiangnan/sigir18-deep.pdf

Overview of Search Engine

Information pull: a user pulls information by making a specific request

User intent is explicitly reflected in query:
- Keywords, questions

Content is in
- Webpages, images, …



Key challenge: query-document semantic gap

Example of Query-Document Mismatch

Query	Document	Term matching	Semantic matching
seattle best hotel	seattle best hotels	partial	yes
pool schedule	swimming pool schedule	partial	yes
natural logarithm transformation	logarithm transformation	partial	yes
china kong	china hong kong	partial	no
why are windows so expensive	why are macs so expensive	partial	no

Same Search Intent Different Query Representations
Example: “Distance between Sun and Earth”

“how far” earth sun	average distance from the earth to the sun
“how far” sun	how far away is the sun from earth
average distance earth sun	average distance from earth to sun
how far from earth to sun	distance from earth to the sun
distance from sun to earth	distance between earth and the sun
distance between earth & sun	distance between earth and sun
how far earth is from the sun	distance from the earth to the sun
distance between earth sun	distance from the sun to the earth
distance of earth from sun	distance from the sun to earth
“how far” sun earth	how far away is the sun from the earth
how far earth from sun	distance between sun and earth
how far from earth is the sun	how far from the earth to the sun
distance from sun to the earth	

Same Search Intent Different Query Representations
Example: “Youtube”

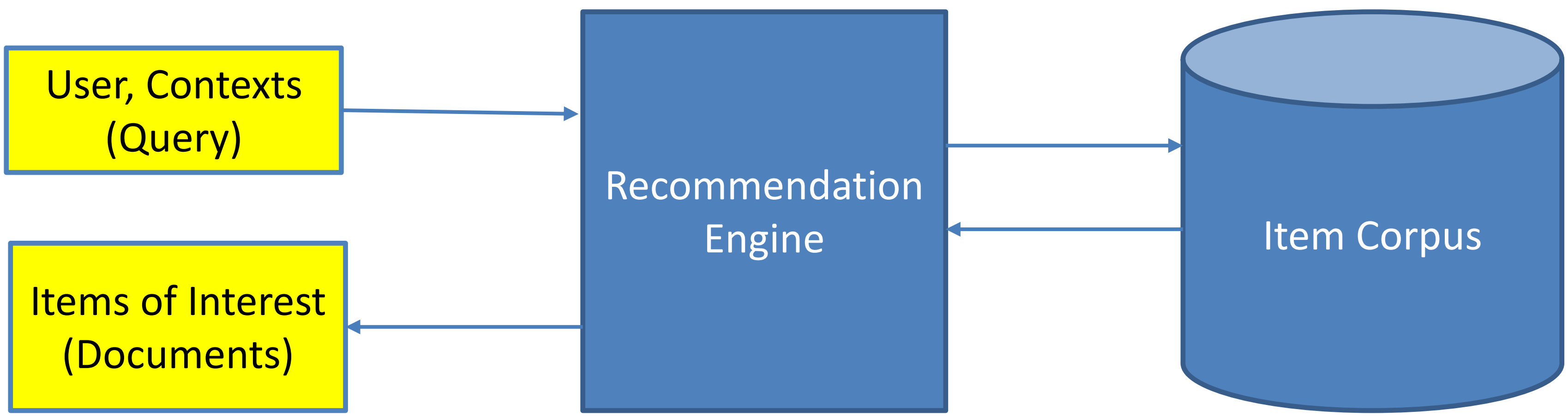
yutube	yuotube	yuo tube
ytube	youtubr	yu tube
youtubo	youtuber	youtubecom
youtube om	youtube music videos	youtube videos
youtube	youtube com	youtube co
youtub com	you tube music videos	yout tube
youtub	you tube com yourtube	your tube
you tube	you tub	you tube video clips
you tube videos	www you tube com	wwww youtube com
www youtube	www youtube com	www youtube co
yotube	www you tube	www utube com
ww youtube com	www utube	www u tube
utube videos	utube com	utube
u tube com	utub	u tube videos
u tube	my tube	toutube
outube	our tube	toutube

Overview of Recommendation Engine

Information push: the system pushes information to a user by guessing the user interest

User Interest is implicitly reflected in:
- Interaction history
- Demographics
- Contexts

Items can be:
- Products, news, movies, videos, friends …



Key challenge: user-item semantic gap

- Even severe than search, since user and item are two **different types of entities** and are represented by different features

Example of User-Item Semantic Gap

Movie Recommendation

?

User Profile (query):
- User ID
- Rating history
- Age, gender
- Income level
- Time of the day

…….

Item Profile (document):
- Item ID
- Description
- Category
- Price
- Image

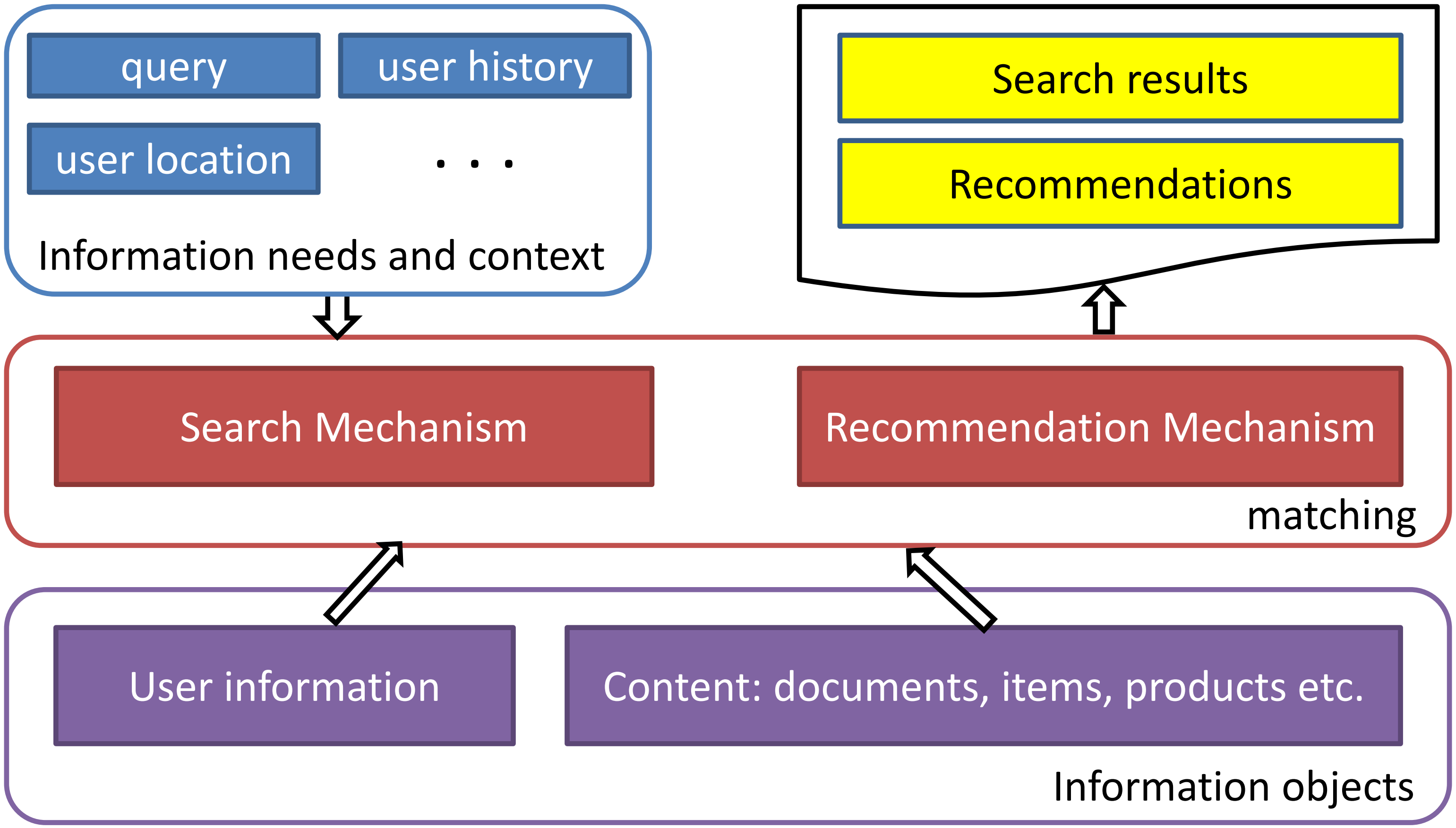
…….

There may be no overlap between user features and item features
Matching cannot be done on the superficial feature level!

Information Providing Mechanisms of Search and Recommendation (Hector et al., CACM' 11)

	Search	Recommendation
Delivery model	Pull	Push or pull
Beneficiary (priority)	User	User and provider
Unexpected good?	No	Yes
Collective knowledge	Maybe	Maybe
Query available	Yes	Maybe
Context dependent	Maybe	Maybe

Unified View on Matching in Search and Recommendation (Hector et al, CACM’11)



Common goal: matching a context (may or may not include an explicit query) to a collection of information objects (product descriptions, web pages, etc.)

Difference for search and recommendation: features used for matching!

Semantic Gap is Biggest Challenge in both Search and Recommendation

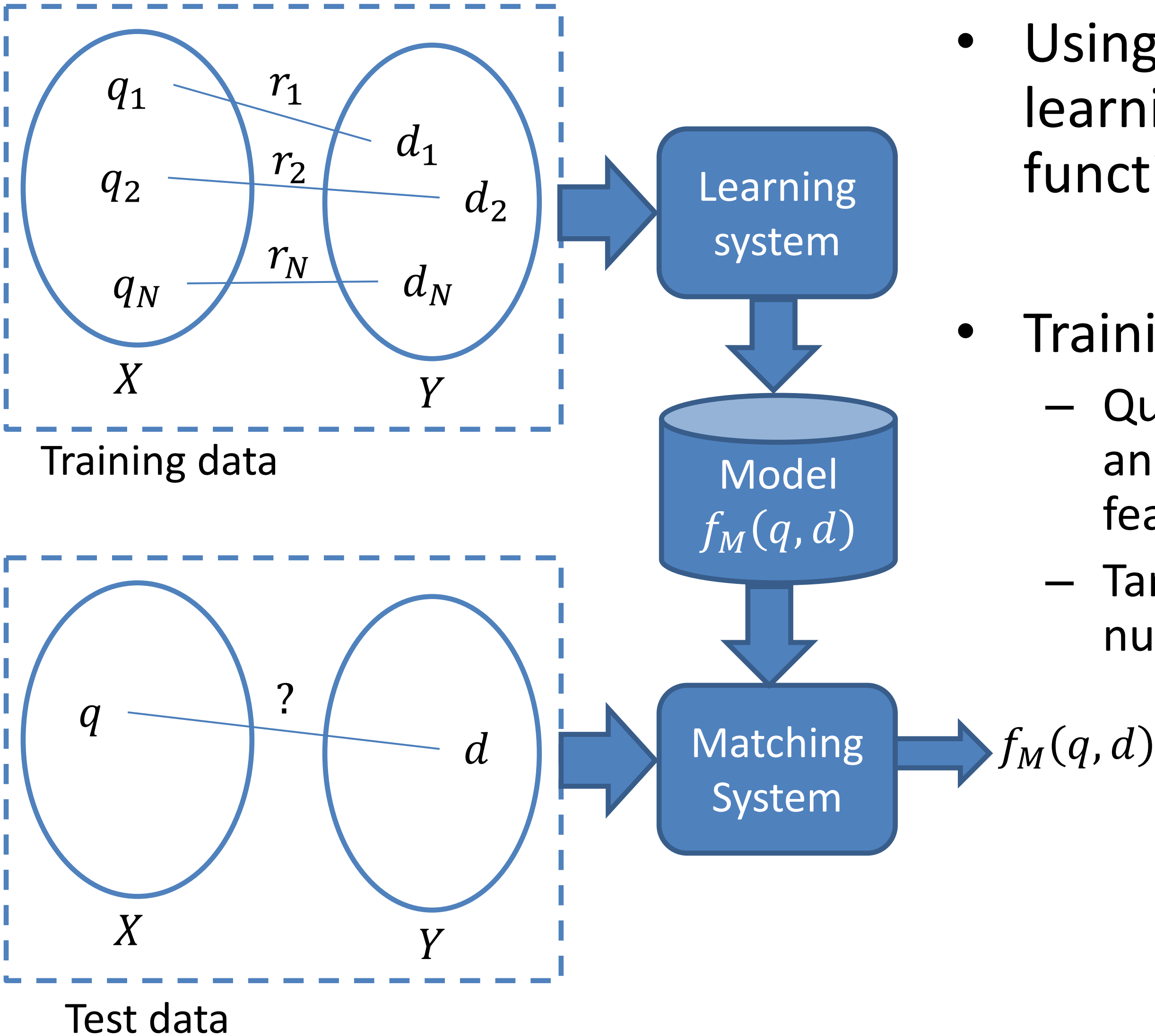
Query-document Mismatch

- Same intent can be represented by different queries (representations)
- Search is still mainly based on term level matching
- Query document mismatch occurs, when searcher and author use different representations

User-item Semantic Gap

- Features are used to represent a user and an item may be totally different (e.g., ID feature)
- Even when they partially overlap in features, it is insensible to conduct direct matching

Machine Learning for Matching



- Using relations in data for learning the matching function $f_M(q,d)$ or $P(r|q,d)$
- Training data $\{(q_i, d_i, r_i)\}_{i=1}^N$
  - Queries and documents (users and items) represented with feature vectors or ID's
  - Target can be binary or numerical values

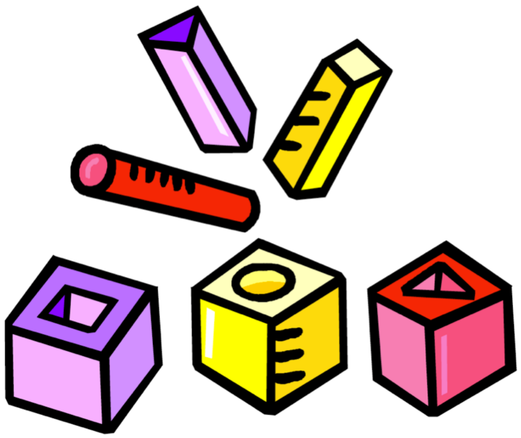
Organization of the Tutorial

- Unified View of Matching in Search and Recommendation (Jun Xu)
- Part 1: Traditional Approaches to Matching
  - Traditional matching models for search (Jun Xu)
  - Traditional matching models for recommendation (Xiangnan He)
- Part 2: Deep Learning Approaches to Matching
  - Overview (Jun Xu)
  - Deep matching models for search (Jun Xu)
  - Deep matching models for recommendation (Xiangnan He)
- Summary (Xiangnan He)

Outline of Tutorial

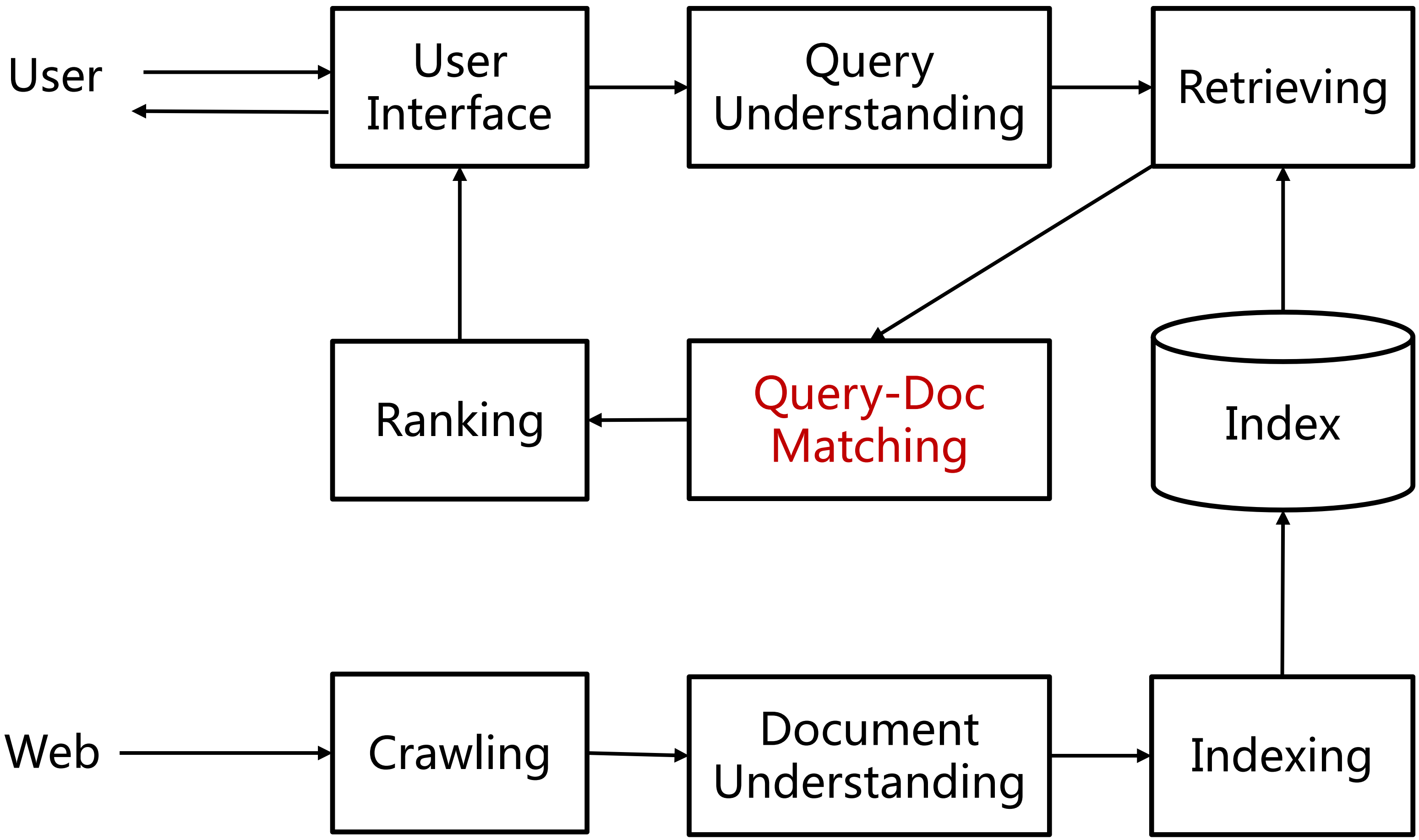
- Unified View of Matching in Search and Recommendation
- Part 1: Traditional Approaches to Matching
  - Traditional matching models for search
  - Traditional matching models for recommendation
- Part 2: Deep Learning Approaches to Matching
- Summary

Slides: http://comp.nus.edu.sg/~xiangnan/sigir18-deep.pdf



QUERY-DOCUMENT MATCHING

Overview of Web Search Engine



Key Factors for Query-Document Matching

Query:
Down the ages noodles and dumplings were famous Chinese food

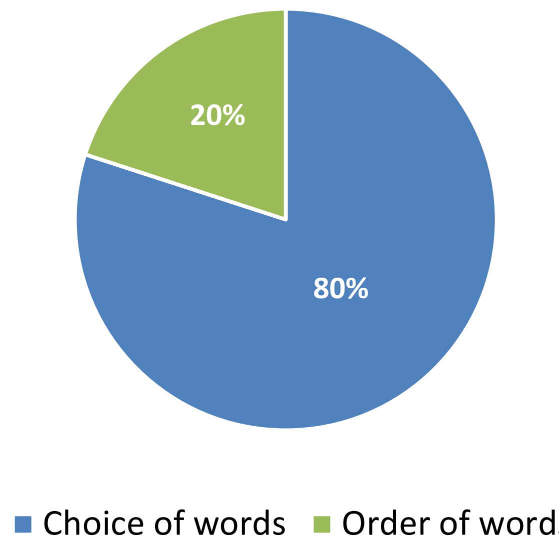
Document:
... down the ages dumplings and noodles were popular in China ...

- Bridging the semantic gap between words
  - Semantically similar words: famous ~ popular, Chinese ~ China
- Capturing order of words
  - N-grams: “down the ages” ~ “down the ages”
  - N-terms: “noodles and dumplings” ~ “dumplings and noodles”

……

Information from Choice of Words and Order of Words (Ross, '02)

- Assume:
  - Size of vocabulary = 10,000
  - Average sentence length = 20
- Rough contributions of information in bits
  - From the selection of words: $\log_2(10000^{20})$
  - From the order of words: $\log_2(20!)$
- "Over 80% of the potential information in language being in the choice of words without regard to the order in which they appear"
  - 80%: choice of words
  - 20%: order of words



Relation between Matching and Ranking

- In traditional IR: ranking = matching

  $f(q,d) = f_{BM25}(q,d)$ or $f(q,d) = f_{LMIR}(d|q)$

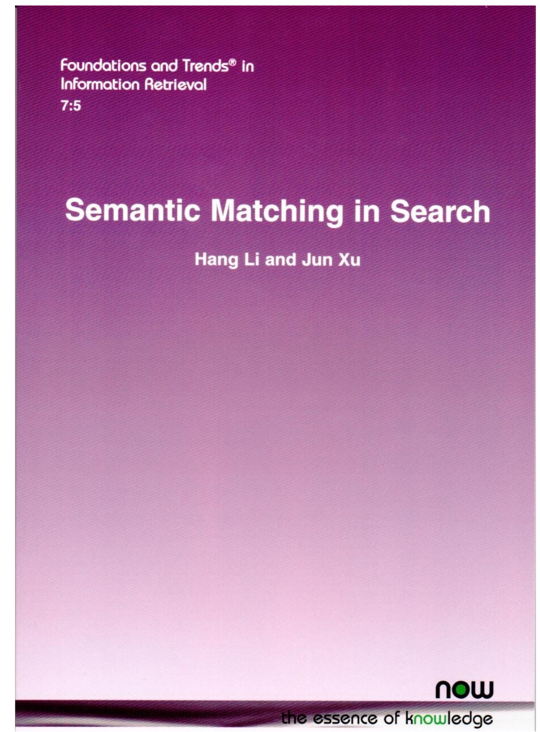
- In Web search: ranking and matching become separated
  - Learning to rank: matching as features for ranking

    $f(q,d) = f_{BM25}(q,d) + \mathrm{PageRank}(d) + \cdots$

	Matching	**Ranking**
Prediction	Matching degree between a query and a document	Ranking list of documents
Model	$f(q,d)$	$f(q,\{d_1,d_2,\cdots\})$
Challenge	Mismatch	Correct ranking on the top

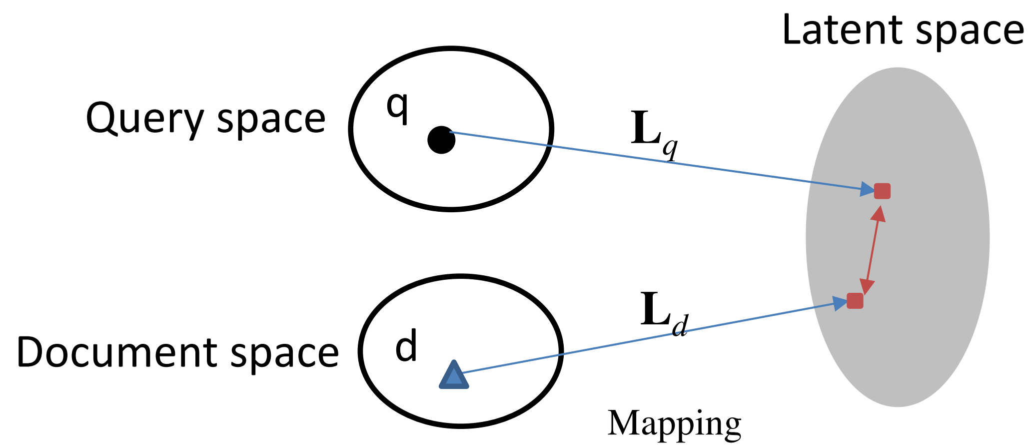
Traditional Approaches to Query-Document Semantic Matching

- Matching by query formulation
- Matching with term dependency
- Matching with topic model
- Matching in latent space model
- Matching with translation model

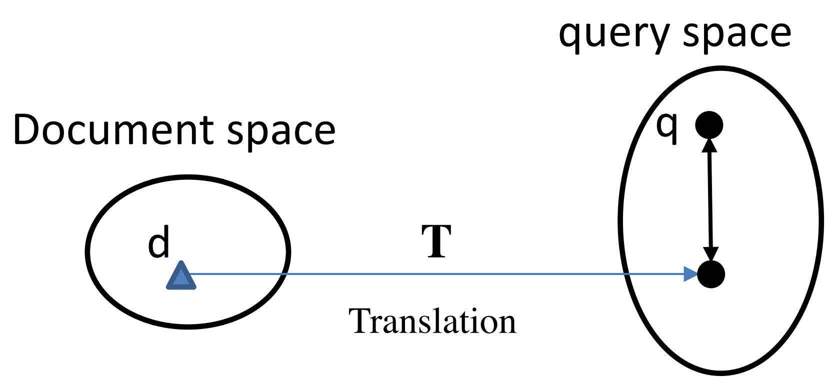


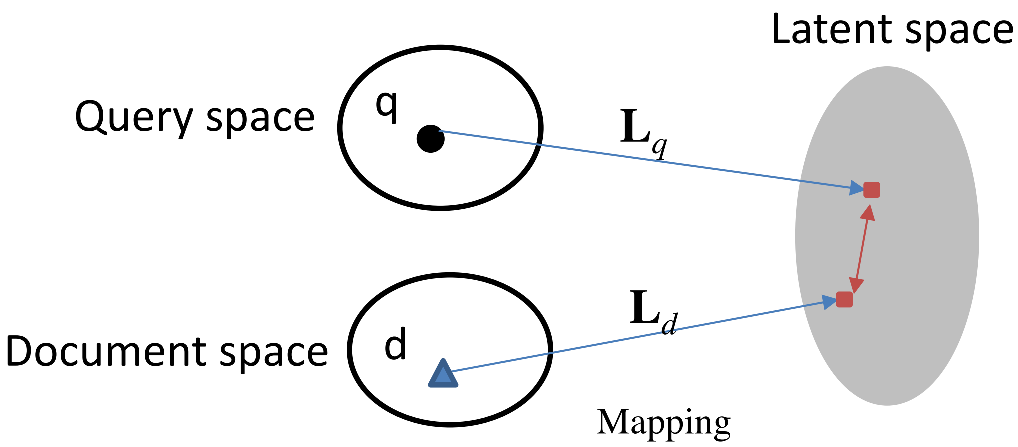
Traditional Matching Models for Search

- **Matching in latent space**: mapping query and document into a latent space



- **Matching with machine translation**: mapping document to query space





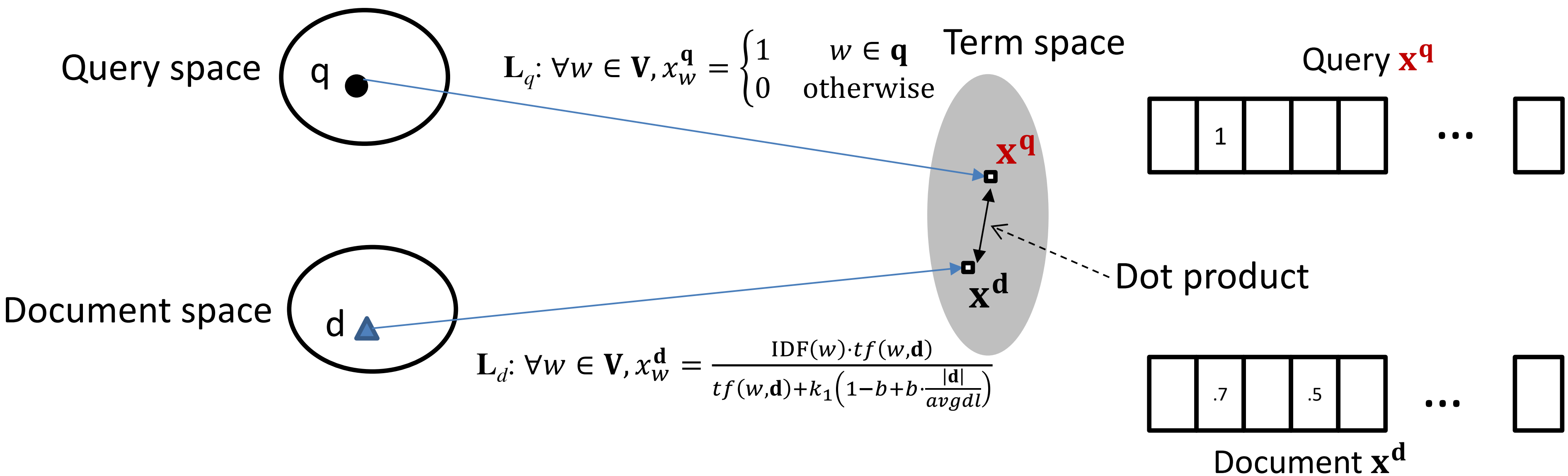
MATCHING IN LATENT SPACE

BM25: Matching in Term Space

$$f_{BM25}(\mathbf{q},\mathbf{d}) = \sum_{q_i \in \mathbf{q}} \mathrm{IDF}(q_i) \frac{tf(q_i,\mathbf{d})}{tf(q_i,\mathbf{d}) + k_1\left(1 - b + b \cdot \frac{|\mathbf{d}|}{avgdl}\right)}$$

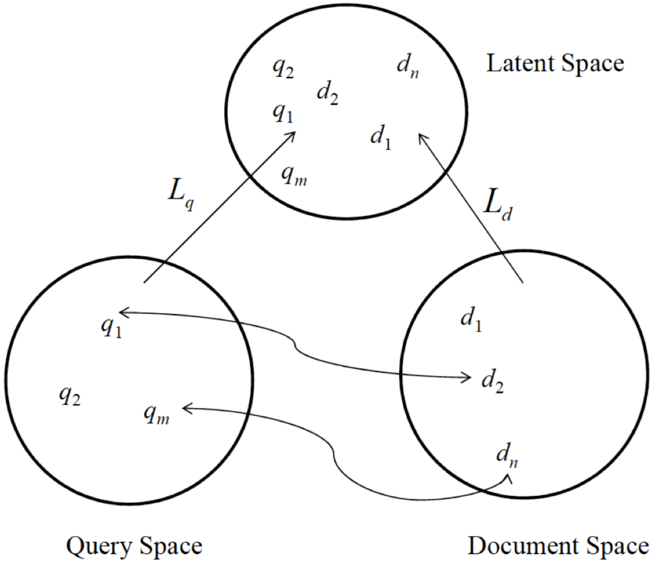
$$= \sum_{w \in \mathbf{V}} \color{red}{1_{w \in \mathbf{q}}} \times \frac{\mathrm{IDF}(w) \cdot tf(w,\mathbf{d})}{tf(w,\mathbf{d}) + k_1\left(1 - b + b \cdot \frac{|\mathbf{d}|}{avgdl}\right)}$$

$$= \langle \color{red}{\mathbf{x}^{\mathbf{q}}}, \mathbf{x}^{\mathbf{d}} \rangle$$



Matching in Latent Space

- Assumption
  - Queries/documents have similarities
  - Click-through data represent "similarity" relations between queries and documents
- Approach
  - Project queries and documents to latent space
  - With some regularization or constraints



Partial Least Square (PLS)

- Two spaces: $\mathcal{X} \subset \mathbb{R}^m$ and $\mathcal{Y} \subset \mathbb{R}^n$
- Training data: $\{(x_i, y_i, r_i)\}_{i=1}^N$, $r_i \in \{+1, -1\}$ or $r_i \in \mathbb{R}$
- Model
  - Dot product as similarity: $f(x, y) = \langle L_X^T x, L_Y^T y \rangle = x^T L_X L_Y^T y$
  - L_X and L_Y are two linear (and orthonormal) transformations
- Objective function

$$\operatorname{argmax}_{L_X, L_Y} \sum_{r_i=+1} x_i^T L_X L_Y^T y_i - \sum_{r_i=-1} x_i^T L_X L_Y^T y_i$$

$$\text{s.t. } L_X^T L_X = I_{K\times K}, L_Y^T L_Y = I_{K\times K}$$

Regularized Mapping to Latent Space (RMLS)

- Two spaces: $\mathcal{X} \subset \mathbb{R}^m$ and $\mathcal{Y} \subset \mathbb{R}^n$
- Training data: $\{(x_i, y_i, r_i)\}_{i=1}^N$, $r_i \in \{+1, -1\}$ or $r_i \in \mathbb{R}$
- Model
  - Dot product as similarity: $f(x, y) = \langle L_X^T x, L_Y^T y \rangle = x^T L_X L_Y^T y$
  - L_X and L_Y are two linear transformations with ℓ_1 and ℓ_2 regularizations (sparse transformations)
- Objective function

$$\operatorname{argmax}_{L_X, L_Y} \sum_{r_i=+1} x_i^T L_X L_Y^T y_i - \sum_{r_i=-1} x_i^T L_X L_Y^T y_i$$

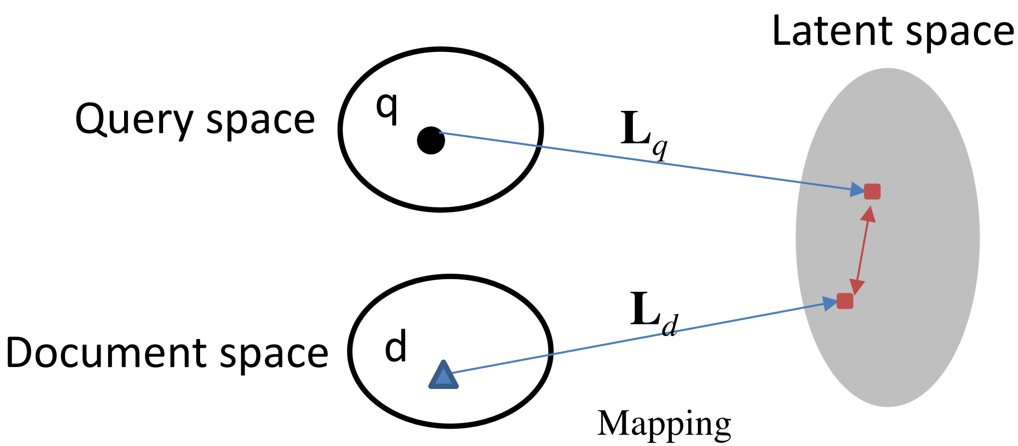
$$\text{s.t.} \quad |L_X| \leq \lambda_X, |L_Y| \leq \lambda_Y, \|L_X\| \leq \vartheta_X, \|L_X\| \leq \vartheta_Y$$

PLS v.s. RMLS

	PLS	RMLS
Transformation Assumption	orthonormal	L1 and L2 regularization
Optimization Method	singular value decomposition	coordinate ascent
Optimality	global optimum	local optimum
Efficiency	low	high
Scalability	low	high

Bridging the Semantic Gap

- Latent space models bridge semantic gap between words through
  - Reducing the dimensionality (from term level matching to semantic matching)
  - Correlating semantically similar terms (matrices are not diagonal)
- Automatically learning mapping functions from data

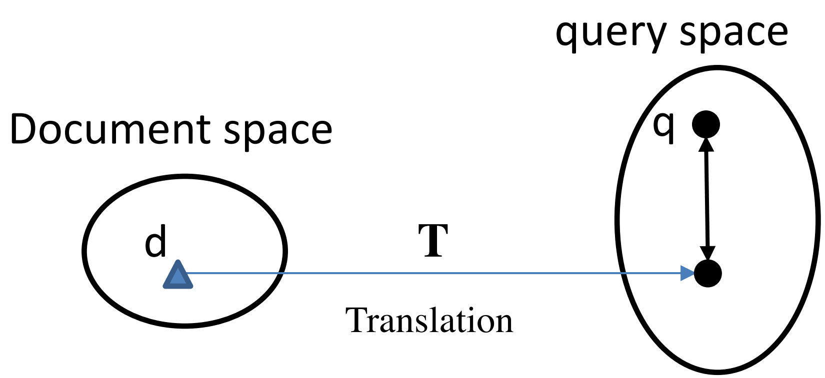


Experimental Results

	NDCG@1	NDCG@3	NDCG@5
BM25	0.637	0.690	0.690
SSI	0.538	0.621	0.629
BLTM	0.657	0.702	0.701
PLS	0.676	0.728	**0.736**
RMLS	**0.686**	**0.732**	0.729

Based on a web search data set containing 94,022 queries and 111,631 documents.
Click through associated with the queries and documents at a search engine is used.

- Latent space models work better than baseline (BM25)
- RMLS works equally well as PLS, with higher learning efficiency and scalability



MATCHING WITH TRANSLATION MODEL

Statistical Machine Translation (SMT)

- Given a sentence C in source language, translates it into sentence E in target language

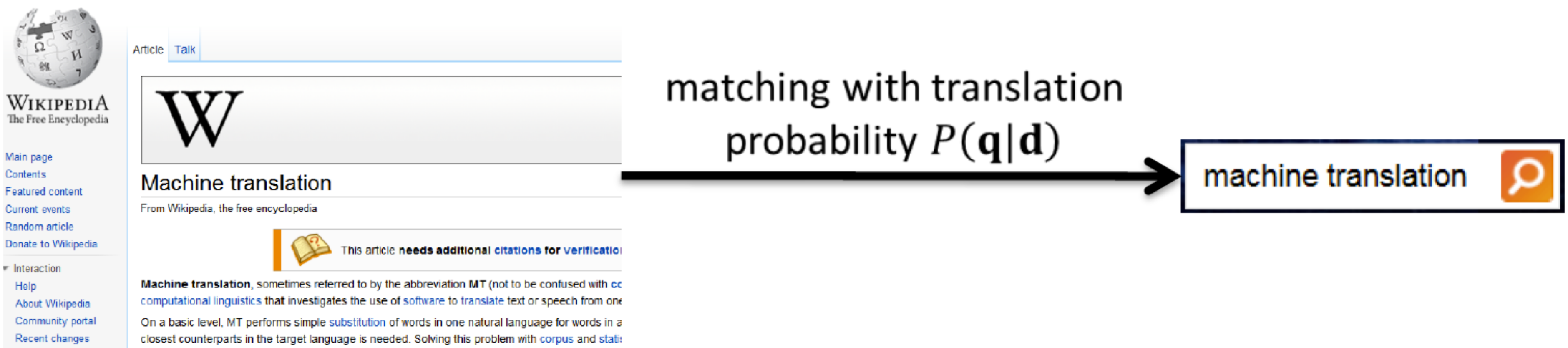
$$E^* = \text{argmax}_E P(E|C)$$

- Linear combination of features

$$P(E|C) = \frac{1}{Z(C,E)} \exp \sum_i \lambda_i h(C,E)$$

$$E^* = \text{argmax}_E \sum_i \lambda_i h(C,E)$$

Statistical Machine Translation for Query-Document Matching



- Translating document **d** to query **q**
- Matching degree: translation probability $P(\mathbf{q}|\mathbf{d})$
- Key difference from conventional translation model
  - Translation within the same language (need to handle self-translation)

Matching with Word-based Translation Models

- Basic model

$$P(\mathbf{q}|\mathbf{d}) = \prod_{q \in \mathbf{q}} P(q|\mathbf{d}) = \prod_{q \in \mathbf{q}} \sum_{w \in \mathbf{d}} \underbrace{P(q|w)}_{\text{translation probability}} \underbrace{P(w|\mathbf{d})}_{\text{Document language model}}$$

- Smoothing to avoid zero translation probability (Berger & Lafferty '99)

$$P(\mathbf{q}|\mathbf{d}) = \prod_{q \in \mathbf{q}} \left(\alpha \underbrace{P(q|C)}_{\text{background language model}} + (1-\alpha) \left(\sum_{w \in \mathbf{d}} P(q|w) P(w|\mathbf{d}) \right) \right)$$

- Self-translation (Gao et al., '10)

$$P(\mathbf{q}|\mathbf{d}) = \prod_{q \in \mathbf{q}} \left(\alpha P(q|C) + (1-\alpha) \left(\beta \underbrace{P(q|\mathbf{d})}_{\text{unsmoothed document language model}} + (1-\beta) \left(\sum_{w \in \mathbf{d}} P(q|w) P(w|\mathbf{d}) \right) \right) \right)$$

Bridging Semantic Gap between Words

- Translation matrix can bridge semantic gap between query words and document words

q	$t(q \mid w)$
solzhenitsyn	0.319
citizenship	0.049
exile	0.044
archipelago	0.030
alexander	0.025
soviet	0.023
union	0.018
komsomolskaya	0.017
treason	0.015
vishnevskaya	0.015

$w =$ solzhenitsyn

q	$t(q \mid w)$
carcinogen	0.667
cancer	0.032
scientific	0.024
science	0.014
environment	0.013
chemical	0.012
exposure	0.012
pesticide	0.010
agent	0.009
protect	0.008

$w =$ carcinogen

q	$t(q \mid w)$
zubin_mehta	0.248
zubin	0.139
mehta	0.134
philharmonic	0.103
orchestra	0.046
music	0.036
bernstein	0.029
york	0.026
end	0.018
sir	0.016

$w =$ zubin

q	$t(q \mid w)$
pontiff	0.502
pope	0.169
paul	0.065
john	0.035
vatican	0.033
ii	0.028
visit	0.017
papal	0.010
church	0.005
flight	0.004

$w =$ pontiff

q	$t(q \mid w)$
everest	0.439
climb	0.057
climber	0.045
whittaker	0.039
expedition	0.036
float	0.024
mountain	0.024
summit	0.021
highest	0.018
reach	0.015

$w =$ everest

q	$t(q \mid w)$
wildlife	0.705
fish	0.038
acre	0.012
species	0.010
forest	0.010
environment	0.009
habitat	0.008
endangered	0.007
protected	0.007
bird	0.007

$w =$ wildlife

From Berger & Lafferty, '99

Experimental Results

Models	NDCG@1	NDCG@3	NDCG@10
BM25 (baseline)	0.3181	0.3413	0.4045
WTM (without self-translation)	0.3210	0.3512	0.4211
WTM (with self-translation)	0.3310	0.3566	0.4232

Based on a large scale real world data set containing 12,071 English queries sampled from one-year query log files of a commercial search engine (Gao et al., 2010)

- Word-based translation model (WTM) outperformed the baseline
  - Translation probabilities bridge the semantic gap between query words and document words
  - Self-translation is effective

References

- Adam Berger, Rich Caruana, David Cohn, Dayne Freitag, and Vibhu Mittal. Bridging the Lexical Chasm: Statistical Approaches to Answer-Finding. In SIGIR 2000.
- Jianfeng Gao, Xiaodong He, and JianYun Nie. Click-through-based Translation Models for Web Search: from Word Models to Phrase Models. In CIKM 2010.
- Jianfeng Gao, Kristina Toutanova, and Wen-tau Yih. Clickthrough-based latent semantic models for web search. In SIGIR 2011.
- Jianfeng Gao : Statistical Translation and Web Search Ranking. http://research.microsoft.com/en-us/um/people/jfgao/paper/SMT4IR.res.pptx
- Dustin Hillard, Stefan Schroedl, and Eren Manavoglu, Hema Raghavan, and Chris Leggetter. Imrpoved Ad Relevance in Sponsored Search. In WSDM 2010.
- Jian Huang, Jianfeng Gao, Jiangbo Miao, Xiaolong Li, Kuansan Wang, Fritz Behr, and C. Lee Giles. Exploring web scale language models for search query processing. In WWW 2010.
- Ea-Ee Jan, Shih-Hsiang Lin, and Berlin Chen. Translation Retrieval Model for Cross Lingual Information Retrieval. In AIRS 2010.
- Rong Jin, Alex G. Hauptmann, and Chengxiang Zhai. Title Language Model for Information Retrieval. In SIGIR 2002.
- Maryan Karimzadehgan and Chengxiang Zhai. Estimation of Statistical Translation Models based on Mutual Information for Ad Hoc Information Retrieval. In SIGIR 2010.
- David Mimno , Hanna M. Wallach , Jason Naradowsky , David A. Smith, Andrew McCallum. Polylingual topic models. In EMNLP 2009.

References

- Adam Berger and John Lafferty. Information Retrieval as Statistical Translation. In SIGIR 1999.
- Jae-Hyun Park, W. Bruce Croft, and David A. Smith. Qusi-Synchronous Dependence Model for Information Retrieval. In CIKM 2011.
- Stefan Riezler and Yi Liu. Query Rewritting Using Monolingual Statistical Machine Translation. In ACL 2010.
- Dolf Trieschnigg, Djoerd Hiemstra, Franciska de Jong, and Wessel Kraaij. A cross-lingual Framework for Monolingual Biomedical Information Retrieval. In CIKM 2010.
- Elisabeth Wolf, Delphine Bernhard, and Iryan Gurevych. Combining Probabilistic and Translation-based Models for Information Retrieval based on Word Sense Annotations. In CLEF Workshop 2009.
- D.R. Hardoon, S. Szedmak, and J. Shawe-Taylor. Canonical correlation analysis: An overview with application to learning methods. Neural Computation, 2004.
- Jianfeng Gao, Kristina Toutanova and Wen-tau Yih. Clickthrough-based latent semantic models for web search. In Proc. of SIGIR, 2011.
- R. Rosipal and N. Krämer. Overview and recent advances in partial least squares. Subspace, Latent Structure and Feature Selection, 2006.
- Wei Wu, Hang Li, and Jun Xu. Learning Query and Document Similarities from Click-through Bipartite Graph with Metadata. Microsoft Research Technical Report, 2011.
- Jun Xu, Hang Li, Chaoliang Zhong, Relevance Ranking Using Kernels, In Proceedings of the 6th Asian Information Retrieval Societies Symposium (AIRS'10), 1-12, 2010.
- Hector Garcia-Molina, Georgia Koutrika, Aditya Parameswaran, Information Seeking: Convergence of Search, Recommendations, and Advertising Communications of the ACM, Vol. 54 No. 11, Pages 121-130.
- Brian H. Ross. Psychology of Learning and Motivation: Advances in Research and Theory. Elsevier. 2002.

Outline of Tutorial

- Unified View of Matching in Search and Recommendation
- Part 1: Traditional Approaches to Matching
  - Traditional matching models for search
  - Traditional matching models for recommendation
    - Collaborative Filtering Models
    - Generic Feature-based Models
- Part 2: Deep Learning Approaches to Matching
- Summary

Slides: http://comp.nus.edu.sg/~xiangnan/sigir18-deep.pdf

Collaborative Filtering

- Collaborative Filtering (CF) is the most well-known technique for recommendation.
  - Homophily assumption: a user preference can be predicted from his/her similar users.
- Math formulation: matrix completion problem

User	Movie	Rating
Alice	Titanic	5
Alice	Notting Hill	3
Alice	Star Wars	1
Bob	Star Wars	4
Bob	Star Trek	5
Charlie	Titanic	1
Charlie	Star Wars	5
. . .	. . .	. . .

Input Tabular data

Movie

User	TI	NH	SW	ST	...
A	5	3	1	?	...
B	?	?	4	5	...
C	1	?	5	?	...
...	...	...	...	...	...

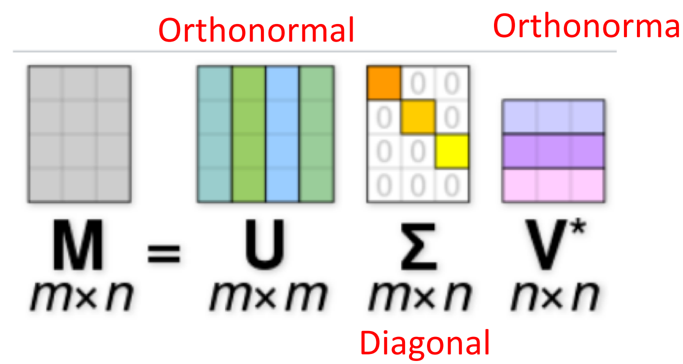
Rating Matrix
(Interaction Matrix)

Solving Matrix Completion

- Singular Value Decomposition (SVD) is the most well-known technique for matrix completion

User	Movie: TI	NH	SW	ST	...
A	5	3	1	?	...
B	?	?	4	5	...
C	1	?	5	?	...
...	...	...	...	...	...

Rating Matrix

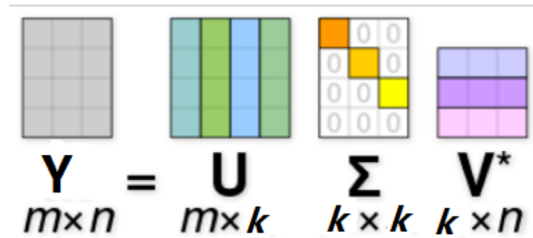


Steps to use SVD for CF:

1. Impute missing data to 0 in **Y**
2. Solving the SVD problem
3. Using only *K* dimensions in **U** and **V** to obtain a low rank model to estimate **Y**

Figure adopted from: https://en.wikipedia.org/wiki/Singular-value_decomposition

SVD is Suboptimal for CF



- In essence, SVD is solving the problem:

$$\arg\min_{\mathbf{U},\Sigma,\mathbf{V}} (\mathbf{Y} - \mathbf{U}\Sigma\mathbf{V}^T)^2$$

$$= \arg\min_{\mathbf{U},\Sigma,\mathbf{V}} \underbrace{\sum_{i=1}^{m}\sum_{j=1}^{n}}_{\text{Training instance}} \Big(\overbrace{\boxed{y_{ij}}}^{\text{Label}} - \underbrace{\boxed{(\mathbf{U}\Sigma\mathbf{V}^T)_{ij}}}_{\text{Model Prediction}}\Big)^2$$

- Several Implications (weaknesses):
  1. Missing data has the same weight as observed data (>99% sparsity)
  2. No regularization is enforced – easy to overfit

Adjust SVD for CF

- The "SVD" method the context of recommendation:
  - Model:

$$\hat{y}_{ui} = \underbrace{\mathbf{v}_u^T}_{\text{User latent vector}} \underbrace{\mathbf{v}_i}_{\text{Item latent vector}}$$

  - Regularized Loss function:

$$L = \underbrace{\sum_u \sum_i w_{ui}(y_{ui} - \hat{y}_{ui})^2}_{\text{Prediction error}} + \lambda \underbrace{(\sum_u ||\mathbf{v}_u||^2 + \sum_i ||\mathbf{v}_i||^2)}_{\text{L2 regularizer}}$$

- This method is also called *Matrix Factorization* (MF) in RecSys:
  - It represents a user and an item as a latent vector (ID embedding).
  - The interaction between user and item is modeled using inner product

(measure how much user latent "preferences" match with item "properties"

  - Besides L2 loss, other loss can also be used, e.g., cross-entropy, margin-based pairwise loss, etc.

Factored Item Similarity Model (Kabbur et al., KDD'14)

- MF encodes a user with an ID, and projects it to embedding.
- Another more information-rich encoding is to use rated items of the user.
  - Also called as item-based CF (i.e., find similar items for recom)

$$\hat{y}_{ui} = \overbrace{(\sum_{\boxed{j \in \mathcal{R}_u}} \mathbf{q}_j)}^{\Leftrightarrow \text{ user representation}}{}^T \mathbf{v}_i$$

Can be interpreted as the **similarity** between item i and j

Items rated by u

SVD++: Fusing User-based and Item-based CF (Koren, KDD'08)

- MF (user-based CF) represents a user as her ID.
  - Directly projecting the ID into latent space
- FISM (item-based CF) represents a user as her interacted items.
  - Projecting interacted items into latent space
- SVD++ fuses the two types of models in the latent space:

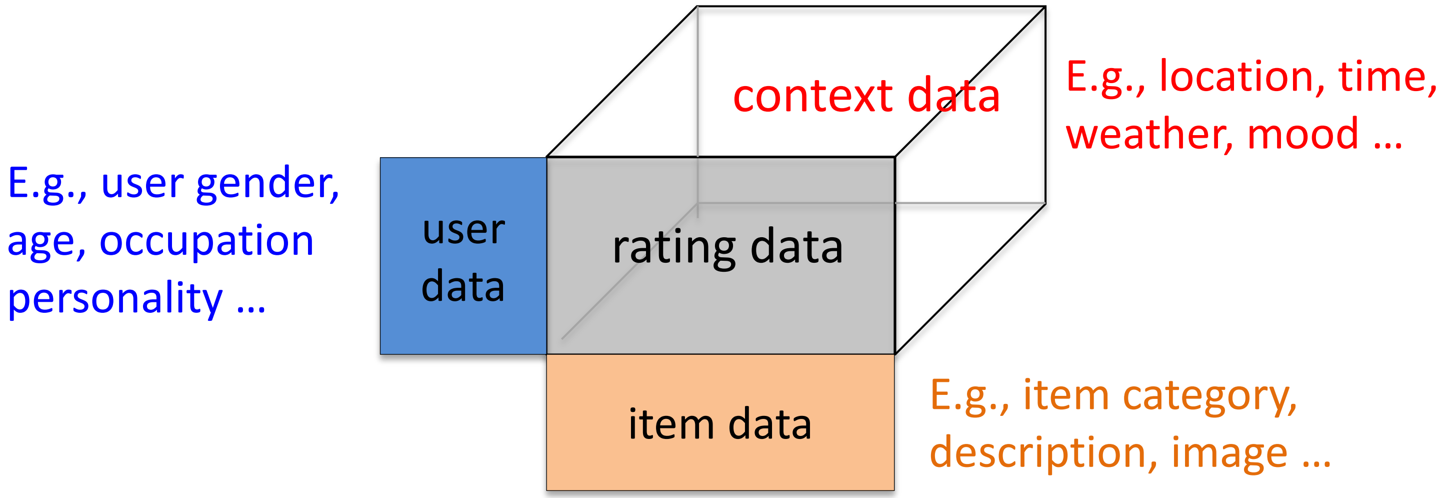
$$\hat{y}_{ui} = \underbrace{(\mathbf{v}_u + \sum_{j \in \mathcal{R}_u} \mathbf{q}_j)}_{\text{User representation in latent space}}{}^T \mathbf{v}_i$$

  - This is the best single model for rating prediction in the Netflix challenge (3 years, 1 million prize).

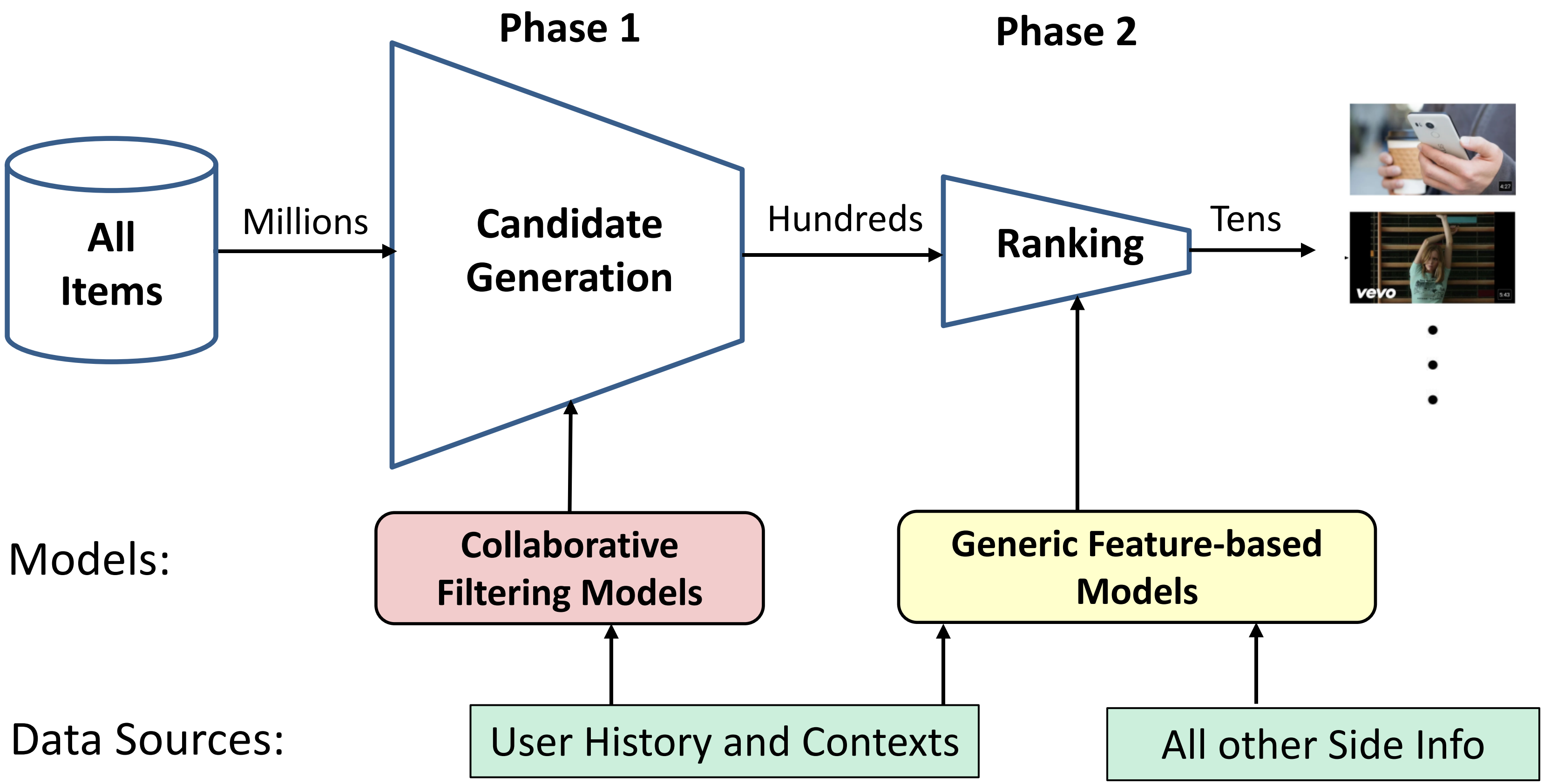
Note: the normalization terms are discarded for clarity.

How about Side Info?

- CF utilizes only the interaction matrix only to build the predictive model.
- How about other information like user/item attributes and contexts?
- Example data used for building a RecSys:



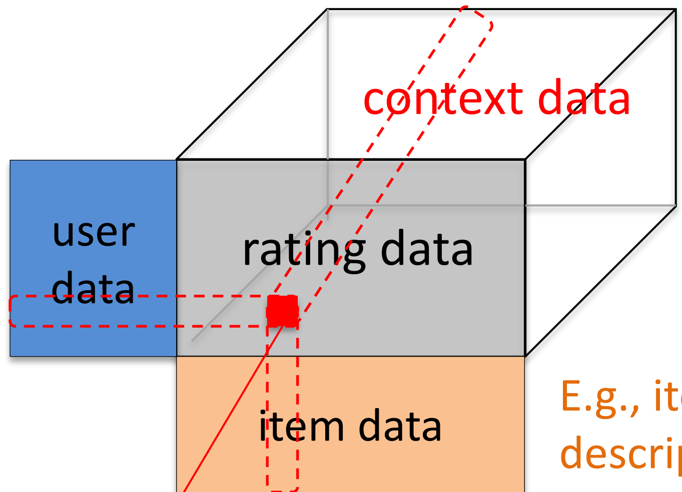
Modern RecSys Architecture



(Covington et al, Recsys'16)

Input to Feature-based Models

E.g., user gender, age, occupation personality …



E.g., recent history, location, time, weather, mood …

E.g., item category, description, image …

One-hot encoding

Each row encodes all info for a rating

Feature vector **x**

	User				Movie					Other Movies rated					Target y	
	A	B	C	...	TI	NH	SW	ST	...	TI	NH	SW	ST	...		
$\mathbf{x}^{(1)}$	1	0	0	...	1	0	0	0	...	0.3	0.3	0.3	0	...	5	$y^{(1)}$
$\mathbf{x}^{(2)}$	1	0	0	...	0	1	0	0	...	0.3	0.3	0.3	0	...	3	$y^{(2)}$
$\mathbf{x}^{(3)}$	1	0	0	...	0	0	1	0	...	0.3	0.3	0.3	0	...	1	$y^{(3)}$
$\mathbf{x}^{(4)}$	0	1	0	...	0	0	1	0	...	0	0	0.5	0.5	...	4	$y^{(4)}$
$\mathbf{x}^{(5)}$	0	1	0	...	0	0	0	1	...	0	0	0.5	0.5	...	5	$y^{(5)}$
$\mathbf{x}^{(6)}$	0	0	1	...	1	0	0	0	...	0.5	0	0.5	0	...	1	$y^{(6)}$
$\mathbf{x}^{(7)}$	0	0	1	...	0	0	1	0	...	0.5	0	0.5	0	...	5	$y^{(7)}$

Sparse Predictive Model

FM: Factorization Machines (Rendle, ICDM'10)

- It represents each feature an embedding vector, and models the second-order feature interactions:

Only nonzero features are considered

$$\hat{y}(\mathbf{x}) = \underbrace{w_0 + \sum_{i=1}^{p} w_i x_i}_{\text{First-order: Linear Regression}} + \underbrace{\sum_{i=1}^{p} \sum_{j>i}^{p} < \mathbf{v}_i, \mathbf{v}_j > \boxed{x_i x_j}}_{\text{Second-order: pair-wise interactions between features}}$$

  - Note: self-interaction is not included: $\cancel{< \mathbf{v}_i, \mathbf{v}_i >}$.
- FM allows easy feature engineering for recommendation, and can mimic many existing models (that are designed for a specific task) by inputting different features.
  - E.g., MF, SVD++, timeSVD (Koren, KDD'09), PITF (Rendle, WSDM'10) etc.

Matrix Factorization with FM

- Input: 2 variables <user (ID), item (ID)>.

Movie

User	TI	NH	SW	ST	...
A	5	3	1	?	...
B	?	?	4	5	...
C	1	?	5	?	...
...	...	...	...	...	...

Feature vector **x**

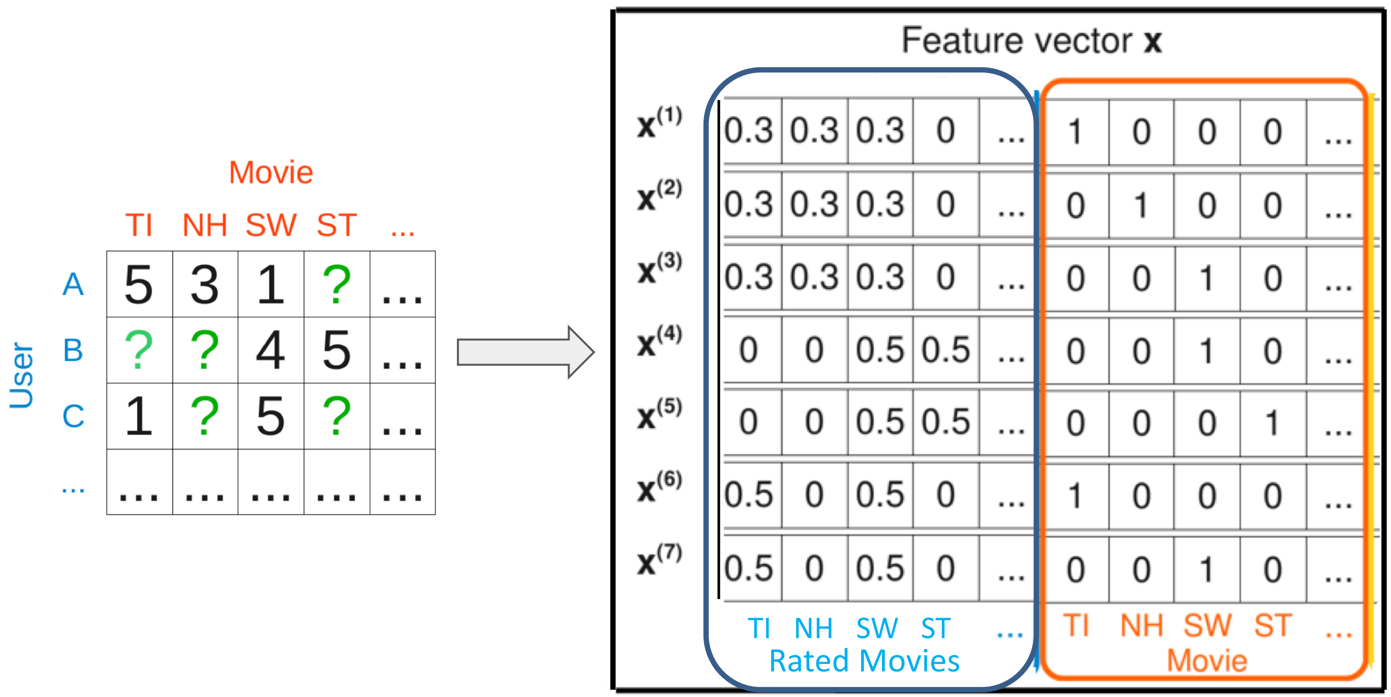
	User: A	User: B	User: C	User: ...	Movie: TI	Movie: NH	Movie: SW	Movie: ST	Movie: ...
$\mathbf{x}^{(1)}$	1	0	0	...	1	0	0	0	...
$\mathbf{x}^{(2)}$	1	0	0	...	0	1	0	0	...
$\mathbf{x}^{(3)}$	1	0	0	...	0	0	1	0	...
$\mathbf{x}^{(4)}$	0	1	0	...	0	0	1	0	...
$\mathbf{x}^{(5)}$	0	1	0	...	0	0	0	1	...
$\mathbf{x}^{(6)}$	0	0	1	...	1	0	0	0	...
$\mathbf{x}^{(7)}$	0	0	1	...	0	0	1	0	...

With this input, FM is identical to MF with bias:

$$\hat{y}(\mathbf{x}) = w_0 + w_u + w_i + \underbrace{< \mathbf{v}_u, \mathbf{v}_i >}_{\text{MF}}$$

Factored Item Similarity Model with FM

- Input: 2 variables <user (historical items ID), item (ID)>.



Further input user ID into FM will resume SVD++

With this input, FM subsumes FISM with additional terms:

$$\hat{y}(\mathbf{x}) = bias + \underbrace{\sum_{j \in \mathcal{R}_u} < \mathbf{v}_j, \mathbf{v}_i >}_{\text{FISM}} + \sum_{j \in \mathcal{R}_u, j' > j} < \mathbf{v}_j, \mathbf{v}_{j'} >$$

Next: Learning Recommender Models

Rating Prediction is Suboptimal

- Old work on recommendation optimize L2 loss:

$$L = \sum_u \sum_i w_{ui}(y_{ui} - \hat{y}_{ui})^2 + \lambda(\sum_u ||\mathbf{v}_u||^2 + \sum_i ||\mathbf{v}_i||^2)$$

  – But many empirical evidence show that:

  A lower error rate does not lead to a good ranking performance…

  – Possible Reasons:

    1) Discrepancy between error measure (e.g., RMSE) and ranking measure.
    2) Observation bias – users tend to rate on the items they like.

Towards Top-N Recommendation

- Recommendation is a personalized ranking task by nature, rather than rating prediction (regression).
  - Evaluated by Precision/Reall/AUC etc, rather than RMSE!
- Optimizing the relative ranking of a user on two items are more advantageous:
  - Higher rating > Lower rating (explicit feedback)
  - Observed interaction > Unobserved interaction (implicit feedback)

sigmoid　　Positive prediction　　Negative prediction

$$L_{BPR} = \arg\max_{\Theta} \underbrace{\sum_{(u,i,j) \in \mathcal{R}_B}}_{\text{Pairwise training examples: } u \text{ prefers } i \text{ over } j} \ln \sigma\left(\boxed{\hat{y}_{ui}} - \boxed{\hat{y}_{uj}}\right) - \lambda ||\Theta||^2$$

  - Known as the Bayesian Personalized Ranking objective (BPR, Rendle et al, UAI’09)

References

- https://en.wikipedia.org/wiki/Collaborative_filtering
- Xiangnan He, Hanwang Zhang, Min-Yen Kan, and Tat-Seng Chua. Fast matrix factorization for online recommendation with implicit feedback. In SIGIR 2016.
- Yehuda Koren, and Robert Bell. Advances in collaborative filtering. Recommender systems handbook. Springer, Boston, MA, 2015. 77-118.
- Santosh Kabbur, Xia Ning, and George Karypis. Fism: factored item similarity models for top-n recommender systems. In KDD 2013.
- Yehuda Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In KDD 2018.
- Paul Covington, Jay Adams, and Emre Sargin. Deep neural networks for youtube recommendations. In Recsys 2016.
- Steffen Rendle. Factorization machines. In ICDM 2010.
- Yehuda Koren. Collaborative filtering with temporal dynamics. Communications of the ACM 53, no. 4 (2010): 89-97.
- Steffen Rendle, and Lars Schmidt-Thieme. Pairwise interaction tensor factorization for personalized tag recommendation. In WSDM 2010.
- Immanuel Bayer, Xiangnan He, Bhargav Kanagal, and Steffen Rendle. A generic coordinate descent framework for learning from implicit feedback. In WWW 2017.
- Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. Neural collaborative filtering. In WWW 2017.
- Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. BPR: Bayesian personalized ranking from implicit feedback. In UAI 2009.

Outline of Tutorial

- Unified View of Matching in Search and Recommendation
- Part 1: Traditional Approaches to Matching
- Part 2: Deep Learning Approaches to Matching
  - Overview
  - Deep matching models for search
  - Deep matching models for recommendation
- Summary

Slides: http://comp.nus.edu.sg/~xiangnan/sigir18-deep.pdf

Growing Interests in "Deep Matching"

- Success of deep learning in other fields
  - Speech recognition, computer vision, and natural language processing
- Growing presence of deep learning in IR research
  - SIGIR keynote, Tutorial, and Neu-IR workshop
- Adopted by industry
  - ACM News: *Google Turning its Lucrative Web Search Over to AI Machines* (Oct. 26, 2015)
  - WIRED: *AI is Transforming Google Search. The Rest of the Web is Next* (April 2, 2016)
- Chris Manning (Stanford)'s SIGIR 2016 keynote:
  "I'm certain that *deep learning will come to dominate SIGIR over the next couple of years* … just like speech, vision, and NLP before it."

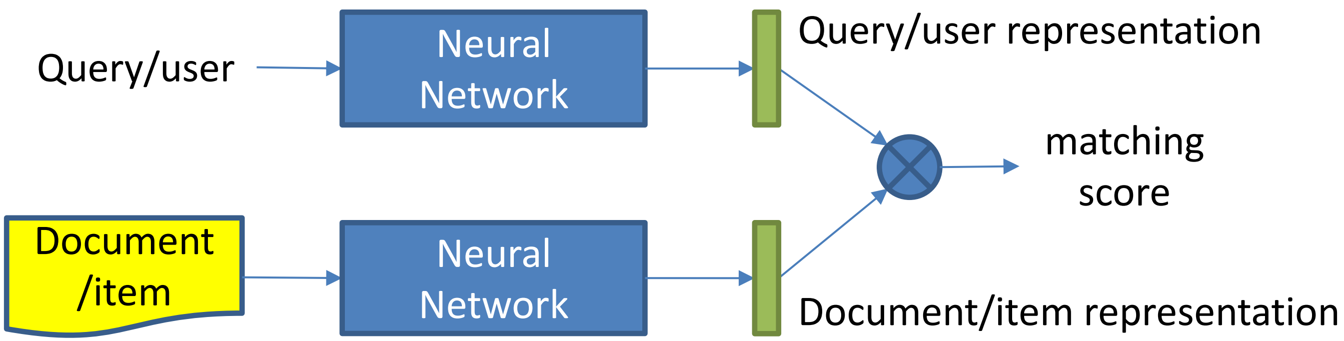
“Deep” Semantic Matching

- Representation
  - Word: one hot —> distributed
  - Sentence: bag-of-words —> distributed representation
  - Better representation ability, better generalization ability

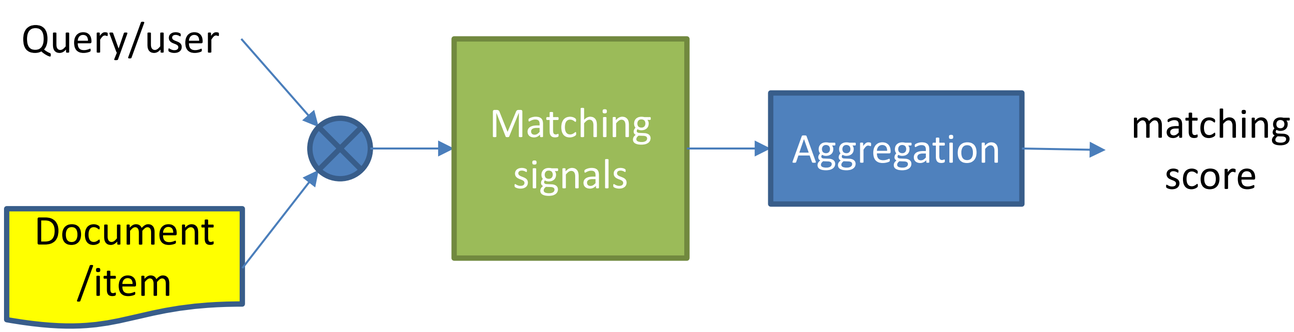
- Matching function
  - Inputs (features): handcrafted —> automatically learned
  - Function: simple functions (e.g., cosine, dot product) —> neural networks (e.g., MLP, neural tensor networks)
  - Involving richer matching signals
  - Considering soft matching patterns

Deep Learning Paradigms for Matching

- Methods of representation learning

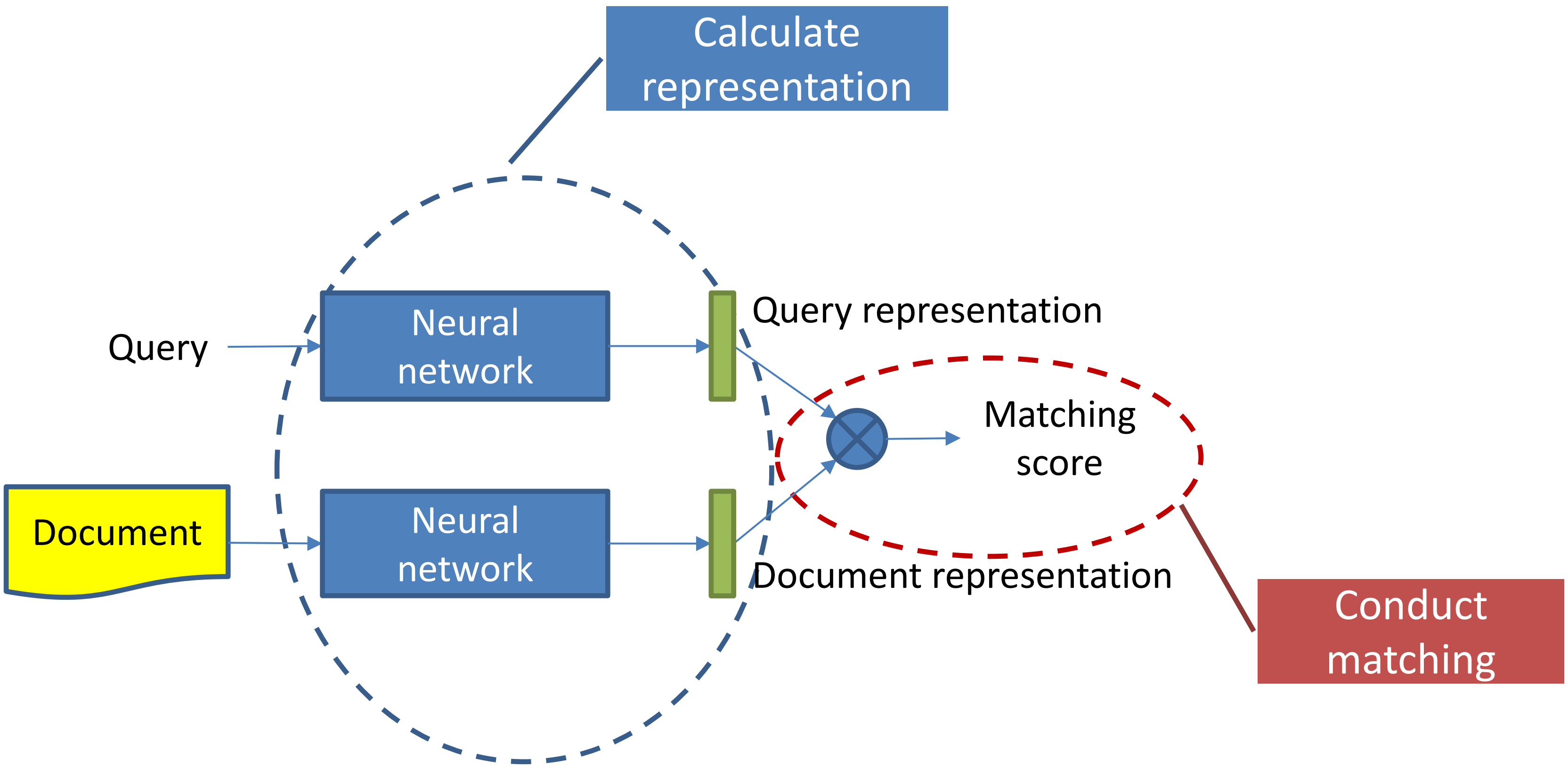


- Methods of matching function learning



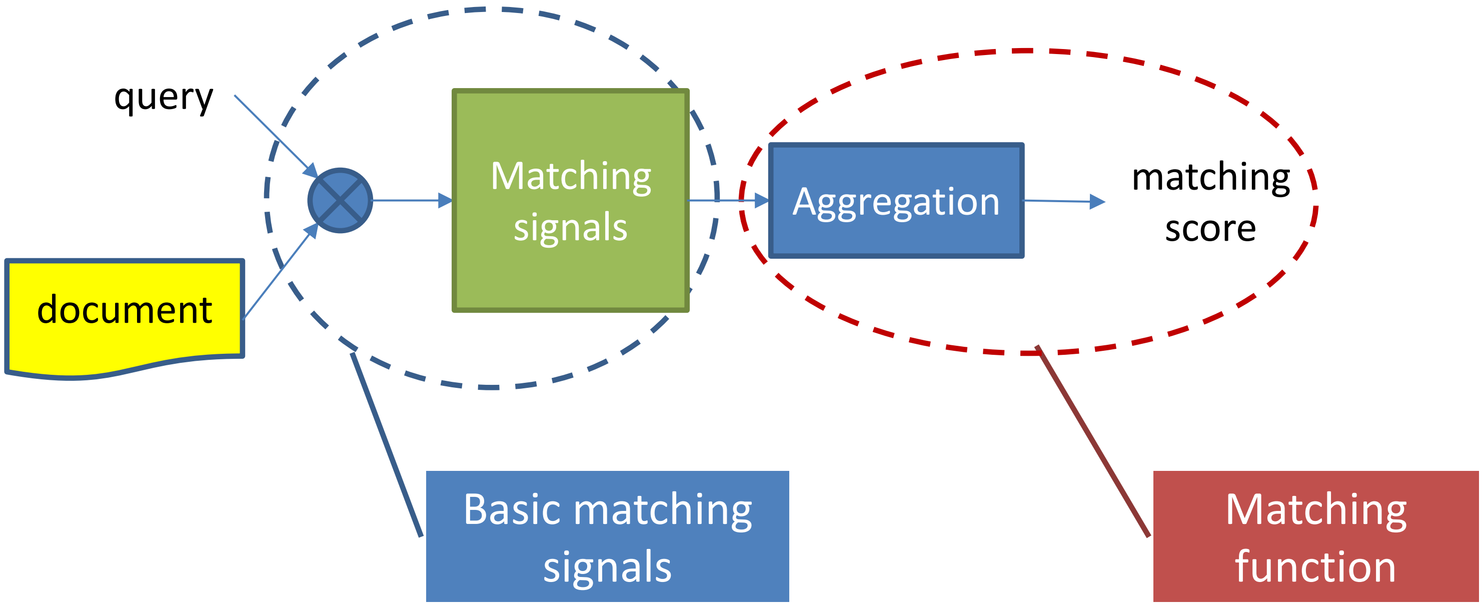
Methods of Representation Learning

- Step 1: calculate representation $\phi(x)$
- Step 2: conduct matching $F\big(\phi(x), \phi(y)\big)$



Methods of Matching Function Learning

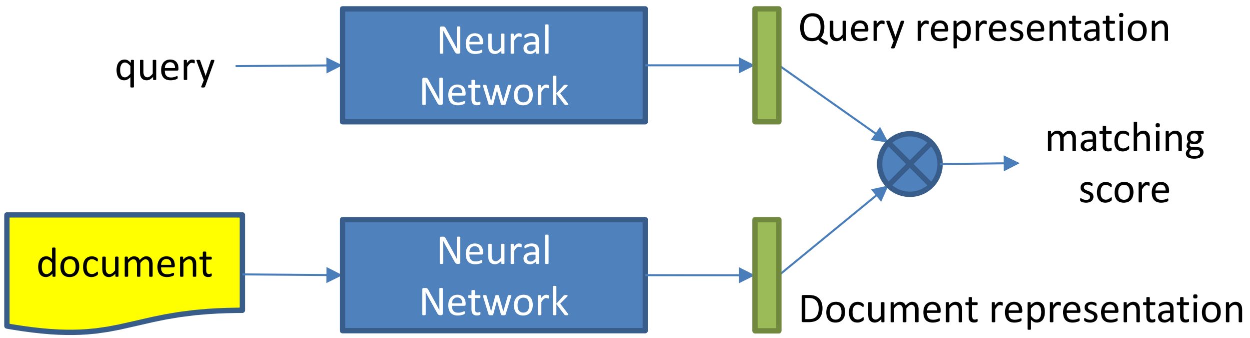
- Step 1: construct basic low-level matching signals
- Step 2: aggregate matching patterns



Outline of Tutorial

- Unified View of Matching in Search and Recommendation
- Part 1: Traditional Approaches to Matching
- Part 2: Deep Learning Approaches to Matching
  - Overview
  - Deep matching models for search
  - Deep matching models for recommendation
- Summary

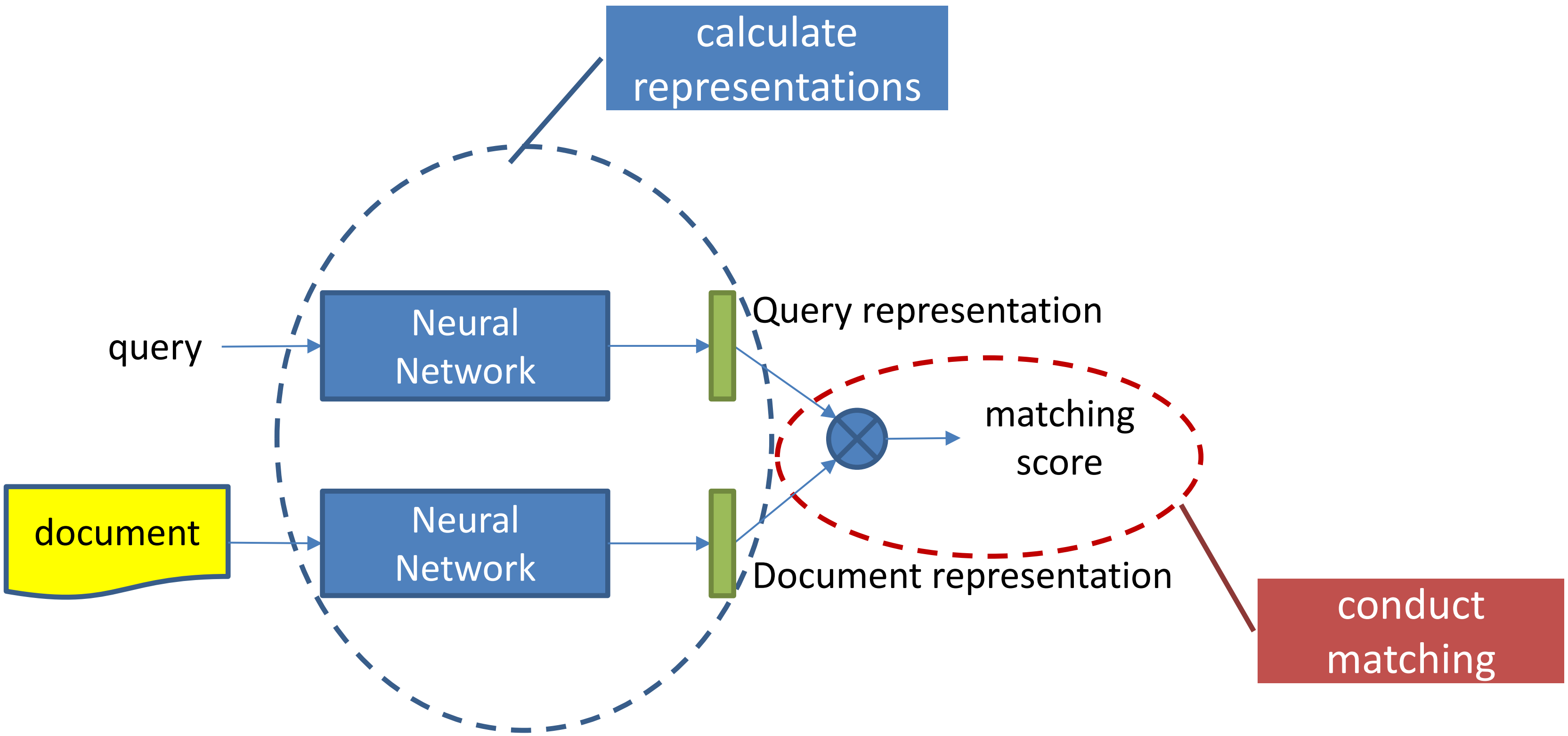
Slides: http://comp.nus.edu.sg/~xiangnan/sigir18-deep.pdf



METHODS OF REPRESENTATION LEARNING

Representation Learning for Query-Document Matching

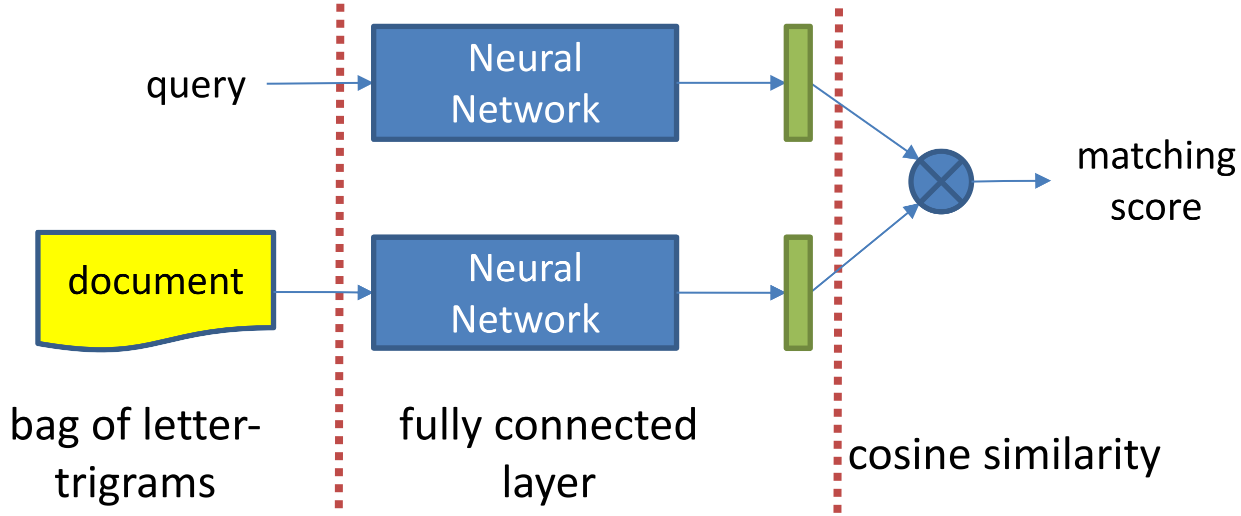
- Step 1: calculate query and document representation
- Step 2: conduct query-document matching



Typical Methods of Representation Learning for Matching

- Based on DNN
  - **DSSM**: Learning Deep Structured Semantic Models for Web Search using Click-through Data (Huang et al., CIKM ’13)
- Based on CNN
  - **CDSSM**: A latent semantic model with convolutional-pooling structure for information retrieval (Shen et al. CIKM ’14)
  - **ARC I**: Convolutional Neural Network Architectures for Matching Natural Language Sentences (Hu et al., NIPS ’14)
  - **CNTN**: Convolutional Neural Tensor Network Architecture for Community-Based Question Answering (Qiu and Huang, IJCAI ’15)
- Based on RNN
  - **LSTM-RNN**: Deep Sentence Embedding Using the Long Short Term Memory Network: Analysis and Application to Information Retrieval (Palangi et al., TASLP ’16)

Deep Structured Semantic Model (DSSM)



- Bag-of-words representation
  - "candy store": [0, 0, 1, 0, …, 1, 0, 0]
- Bag of letter-trigrams representation
  - "#candy# #store#" --> #ca can and ndy dy# #st sto tor ore re#
  - Representation: [0, 1, 0, 0, 1, 1, 0, …, 1]
- Advantages of using bag of letter-trigrams
  - Reduce vocabulary: #words 500K → # letter-trigram: 30K
  - Generalize to unseen words
  - Robust to misspelling, inflection etc.

DSSM Query/Doc Representation: DNN

- Model: DNN (auto-encoder) to capture the compositional sentence representations

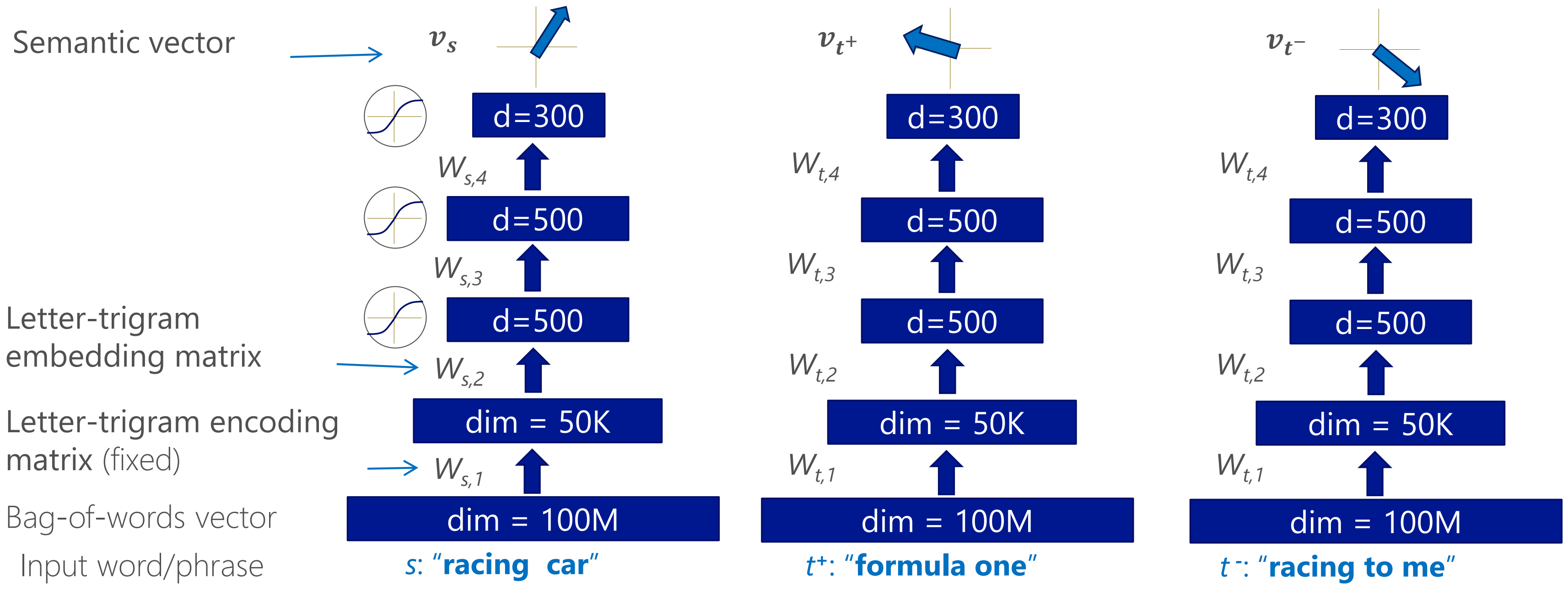


Figure courtesy of He et al., CIKM '14 tutorial

DSSM Matching Function

- Cosine similarity between semantic vectors

$$S = \frac{x^T \cdot y}{|x| \cdot |y|}$$

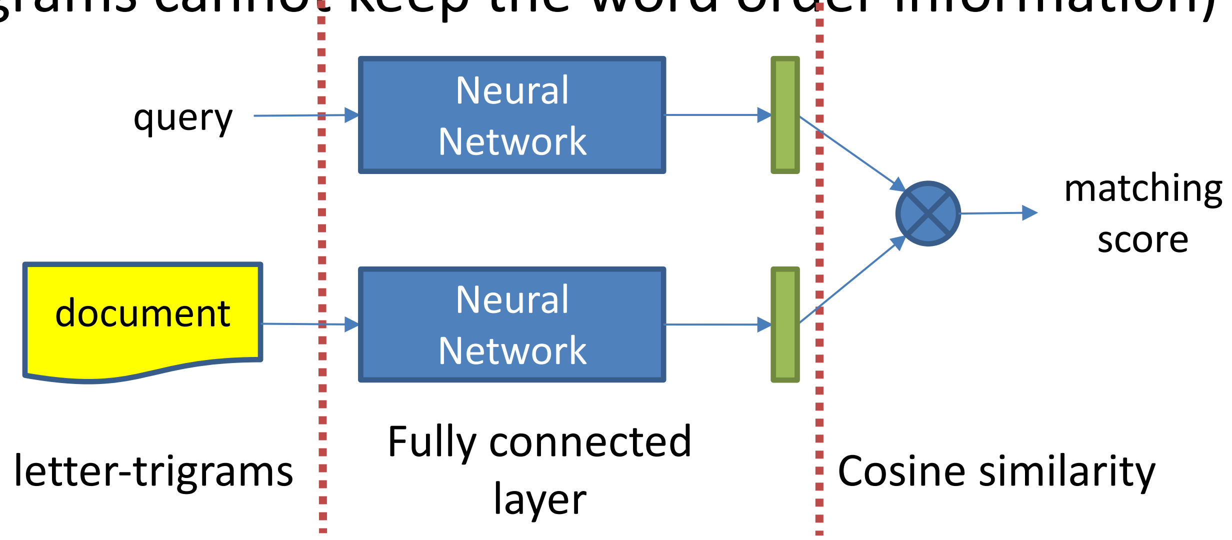
- Training
  - A query q and a list of docs $D = \{d^+, d_1^-, \cdots, d_k^-\}$
  - d^+ positive doc, $d_1^-, \cdots, d_k^-$ negative docs to query
  - Objective:

$$P(d^+|q) = \frac{\exp(\gamma \cos(q, d^+))}{\sum_{d \in D} \exp(\gamma \cos(q, d))}$$

DSSM: Brief Summary

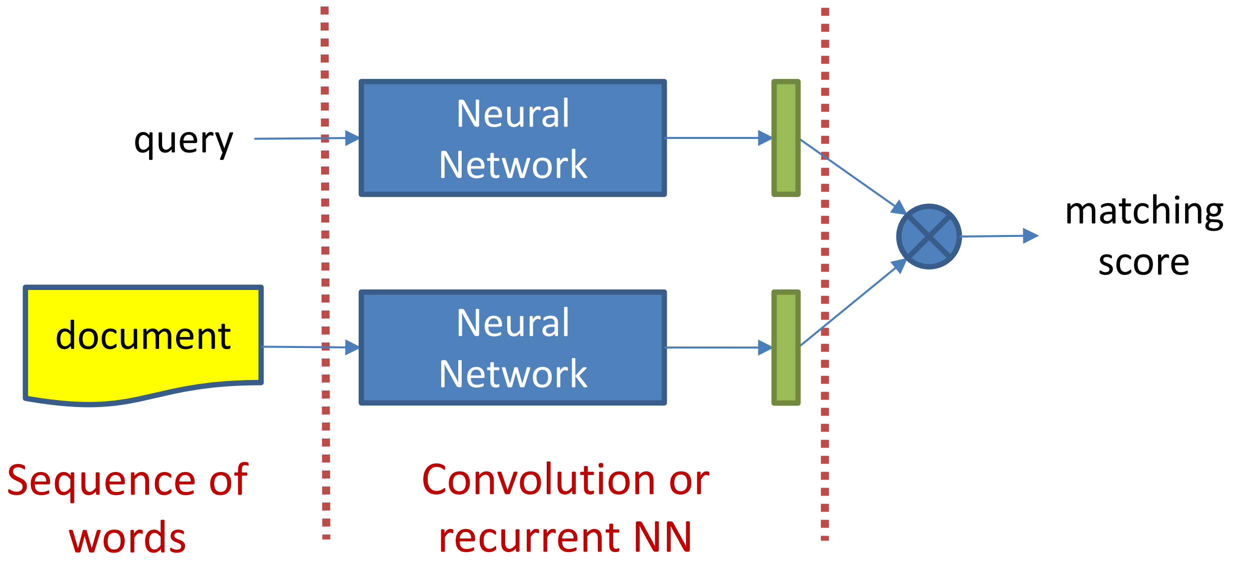
- **Inputs**: Bag of letter-trigrams as input for improving the scalability and generalizability
- **Representations**: mapping sentences to vectors with DNN: semantically similar sentences are close to each other
- **Matching**: cosine similarity as the matching function

- **Problem**: *the order information of words is missing* (bag of letter-trigrams cannot keep the word order information)



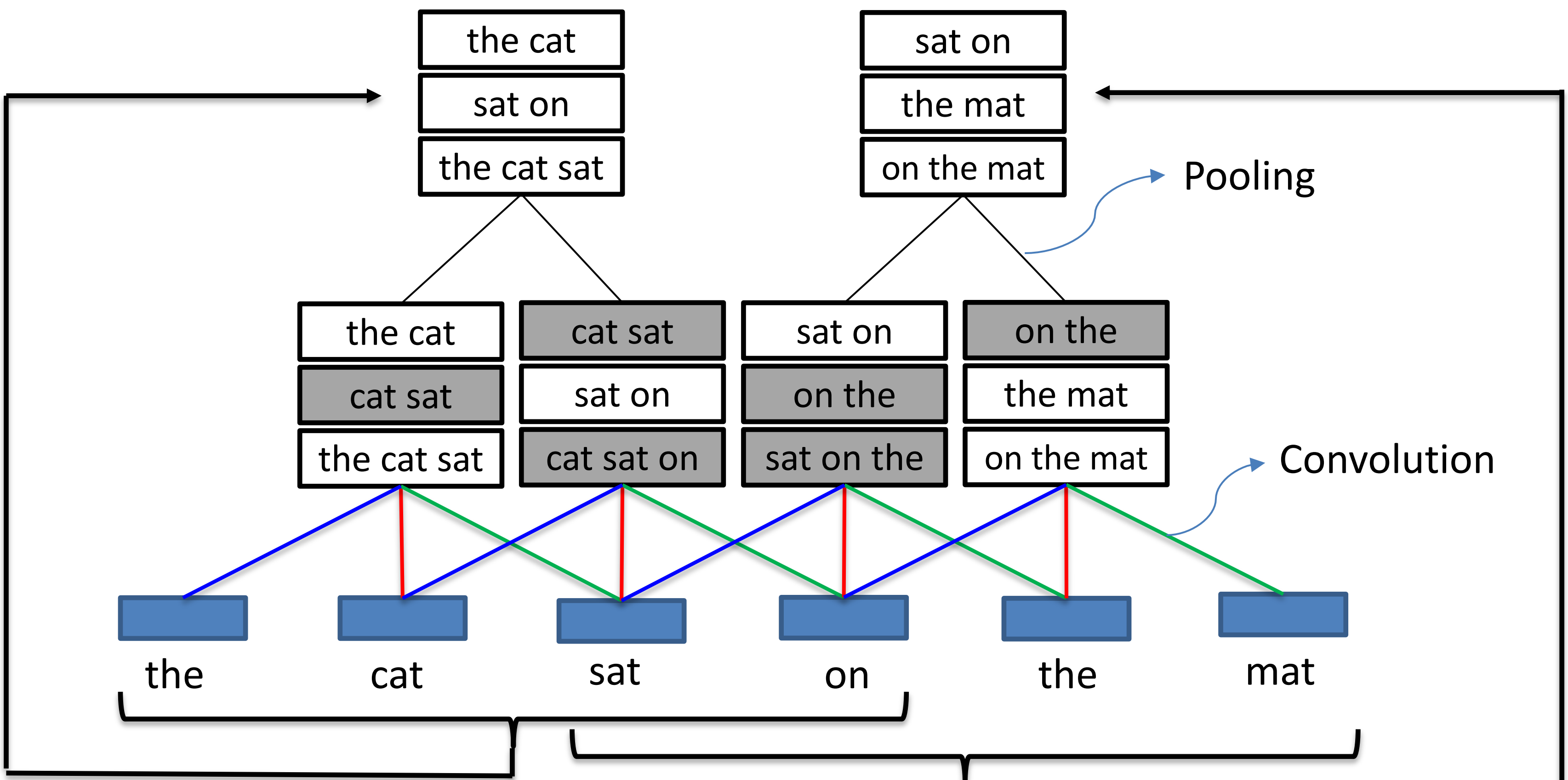
How to Capture Order Information?

- Input: **word sequence** instead of bag of letter-trigrams
- Model
  - **Convolution** based methods can keep locally order
  - **Recurrent** based methods can keep long dependence relations



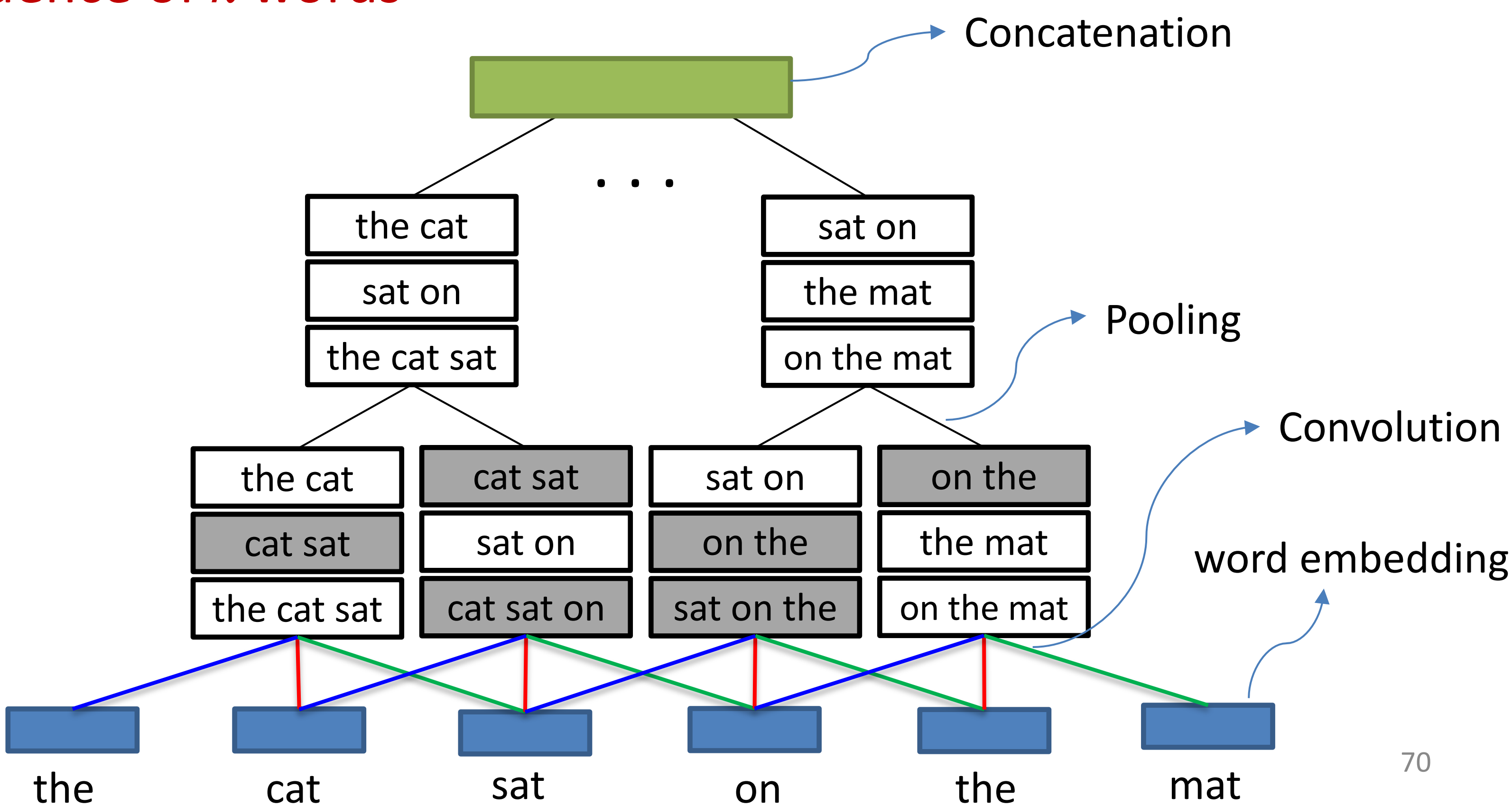
CNN Can Keep Order Information

1-D convolution and pooling operations can keep the local word order information



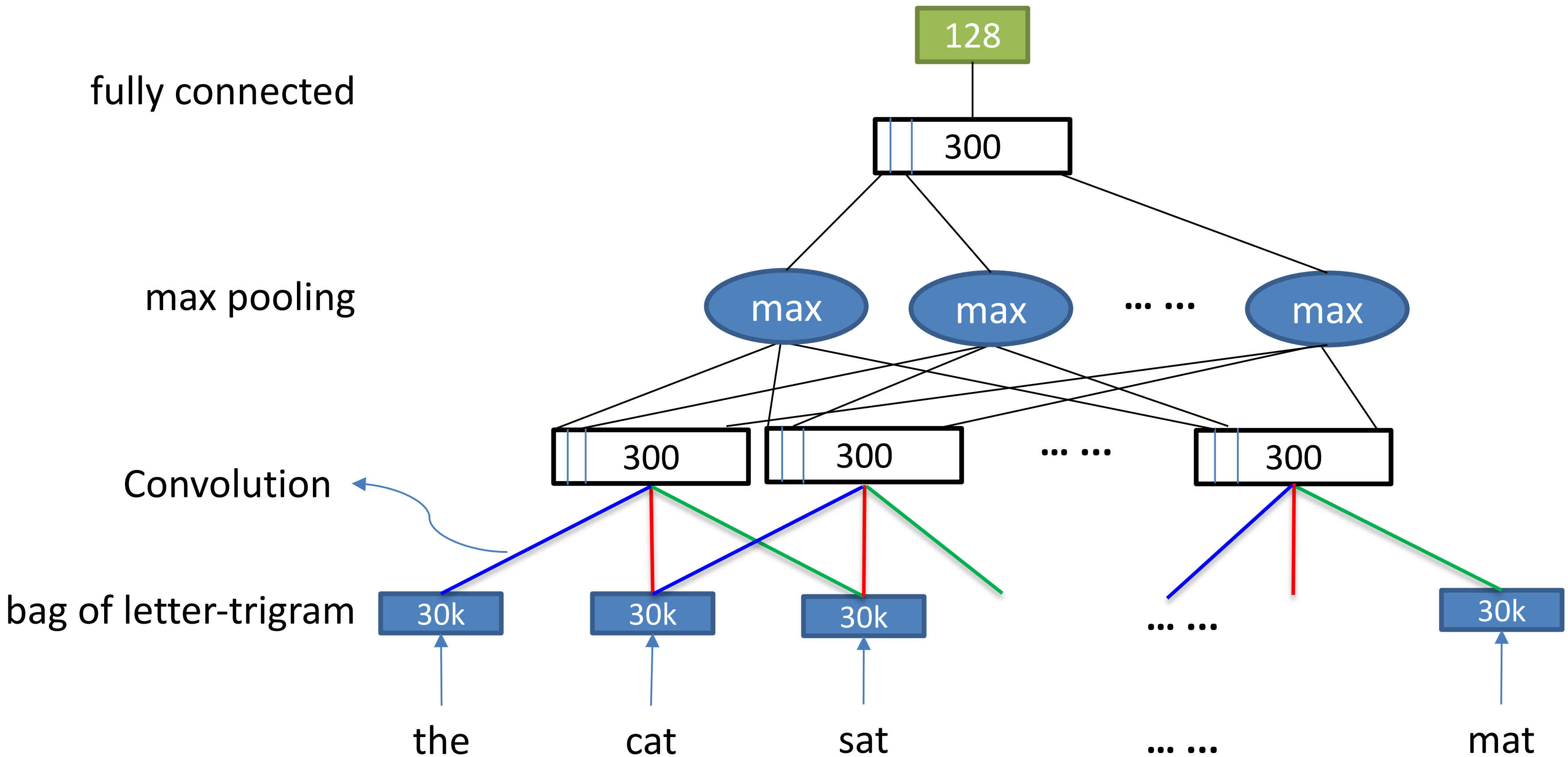
Using CNN: ARC-I (Hu et al., 2014) and CNTN (Qiu et al., 2015)

- Input: sequence of word embeddings trained on a large dataset
- Model: the convolutional operation in CNN compacts each sequence of k words

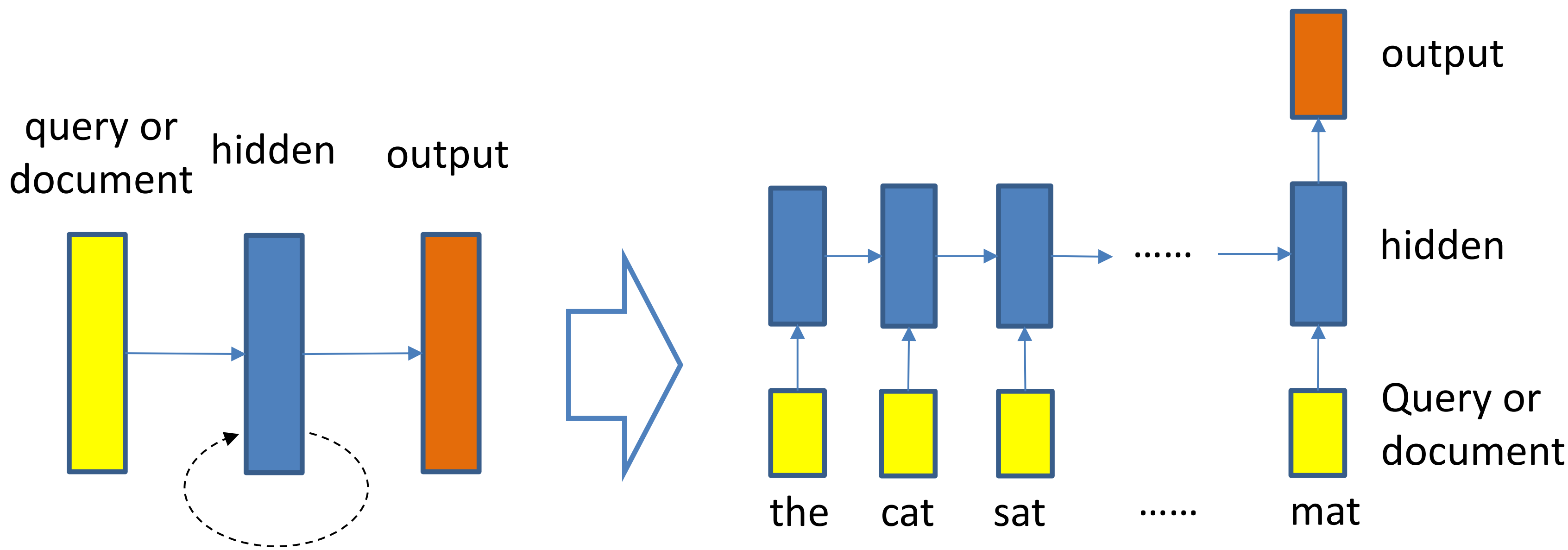


Using CNN: CDSSM (Shen et al., ’14)

The convolutional operation in CNN compacts each sequence of k words



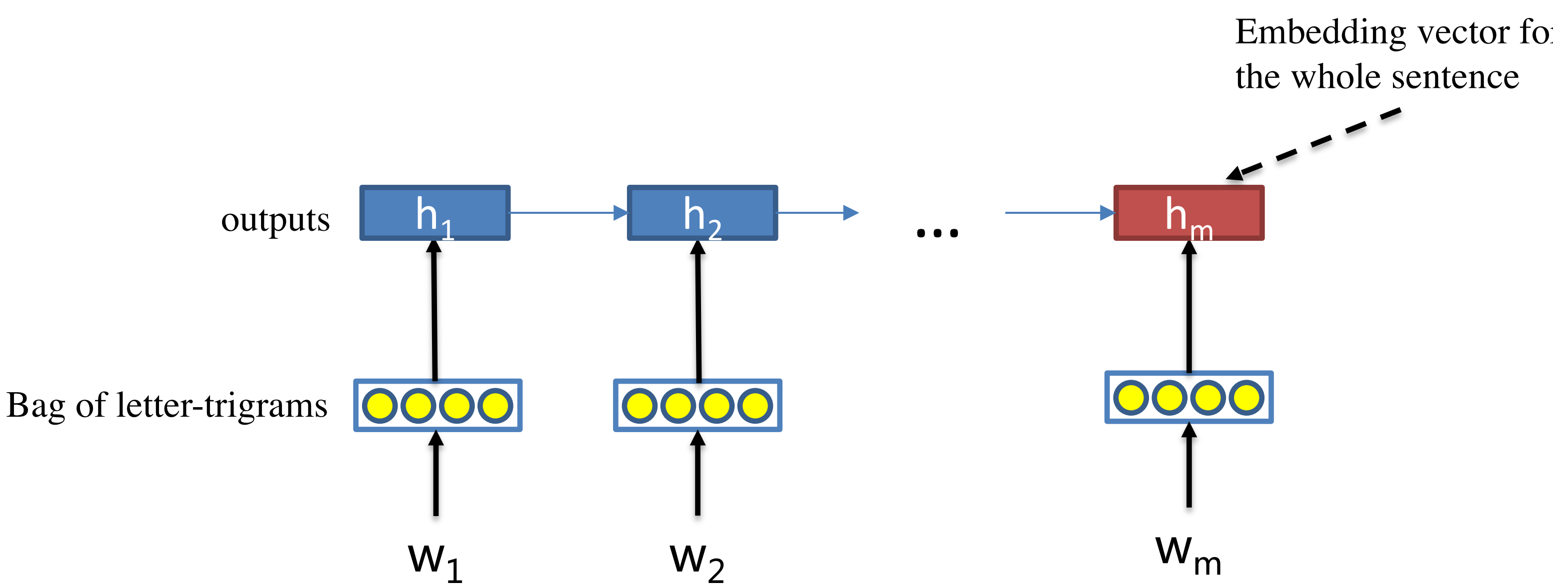
RNN can Keep the Order Information



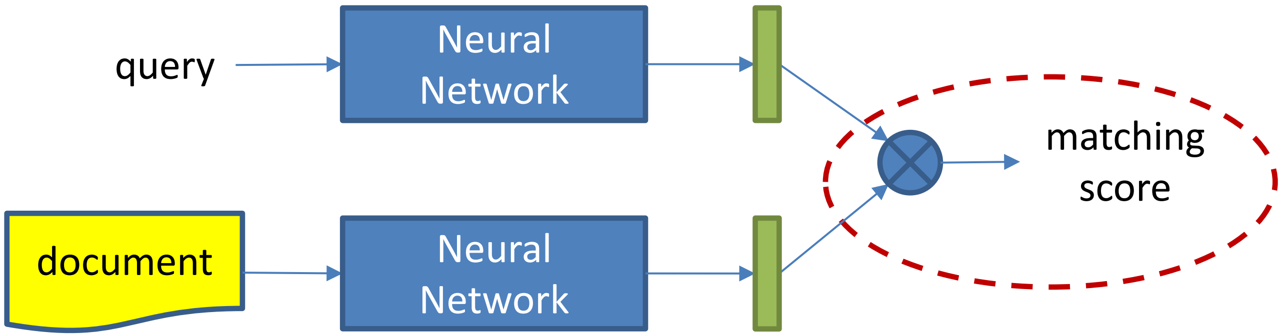
- Two popular variations: long-short term memory (LSTM) and gated recurrent unit (GRU)

Using RNN: LSTM-RNN (Palangi et al., ’16)

- Input: sequence letter trigrams
- Model: long-short term memory (LSTM)
  - The last output as the sentence representation



Matching Function



- **Heuristic**: cosine, dot product
- **Learning**: MLP, Neural tensor networks

Matching Functions (cont')

- Given representations of query and document : q and d
- Similarity between these two representations:
  - Cosine Similarity (DSSM, CDSSM, RNN-LSTM)

$$s = \frac{q^T \cdot d}{|q| \cdot |d|}$$

  - Dot Product

$$s = q^T \cdot d$$

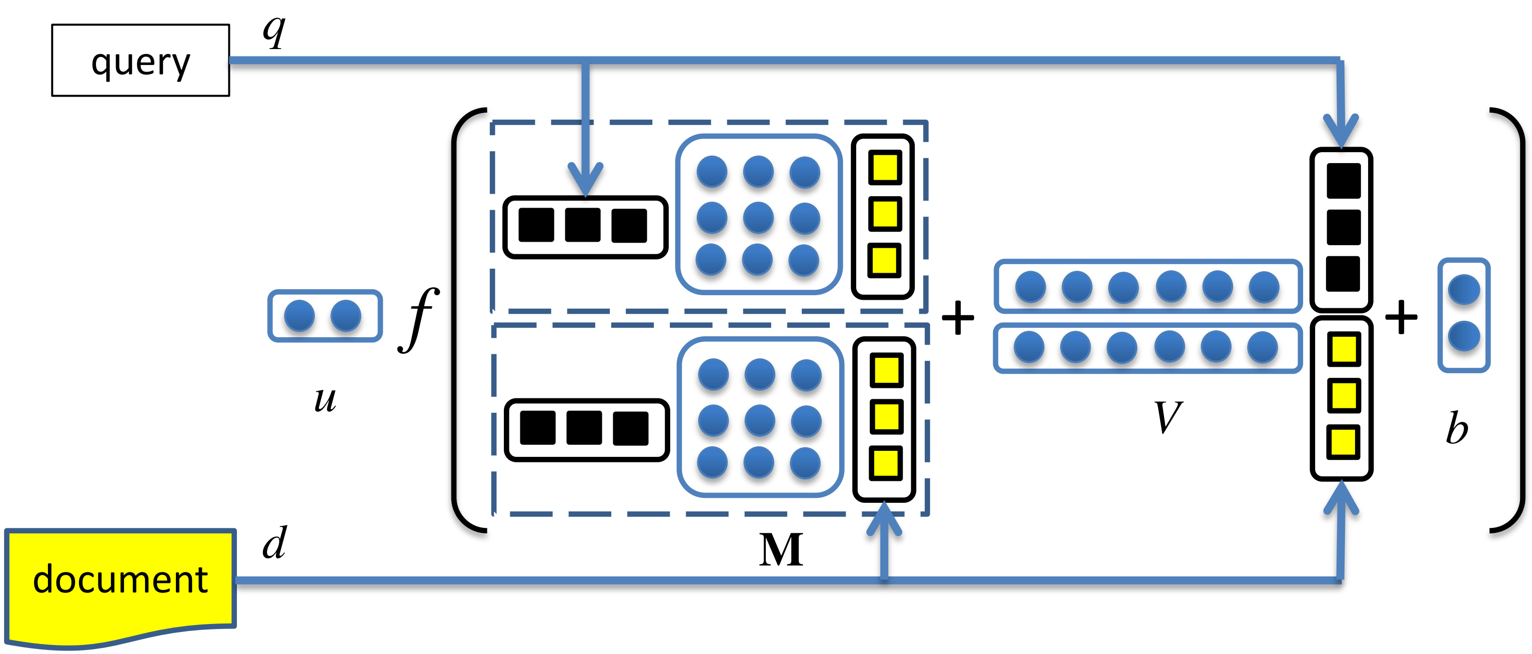
  - Multi-Layer Perception (ARC-I)

$$s = W_2 \cdot \sigma \left(W_1 \cdot \left[\begin{array}{c} q \\ d \end{array} \right] + b_1 \right) + b_2$$

Matching Functions (cont')

- Neural Tensor Networks (CNTN) (Qiu et al., IJCAI '15)

$$s = u^T f \left(q^T \mathbf{M}^{[1:r]} d + V \begin{bmatrix} q \\ d \end{bmatrix} + b \right)$$



Experimental Results

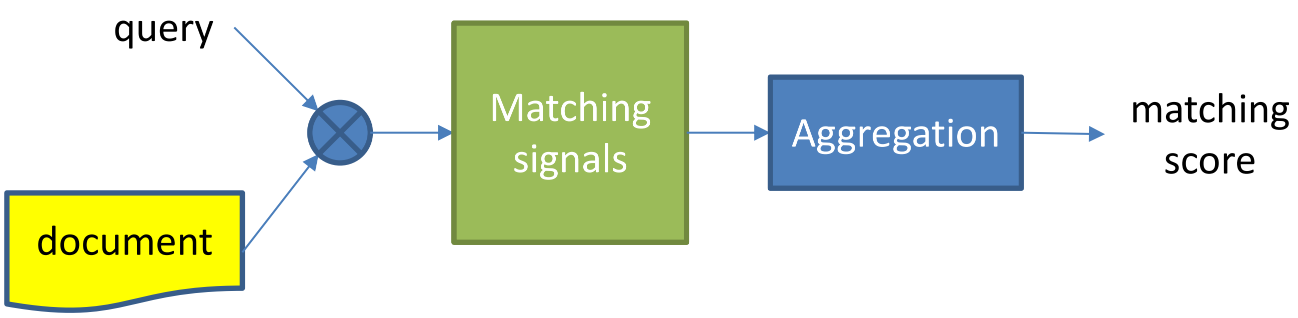
	Model	P@1	MRR
Traditional methods	BM25	0.579	0.726
Representation learning for matching	ARC-I	0.581	0.756
	CNTN	0.626	0.781
	LSTM-RNN	0.690	0.822

Based on Yahoo! Answers dataset (60,564 question-answer pairs)

- Representation learning methods outperformed baselines
  - Semantic representation is important
- LSTM-RNN performed better than ARC-I and CNTN
  - Modeling the order information does help

Short Summary

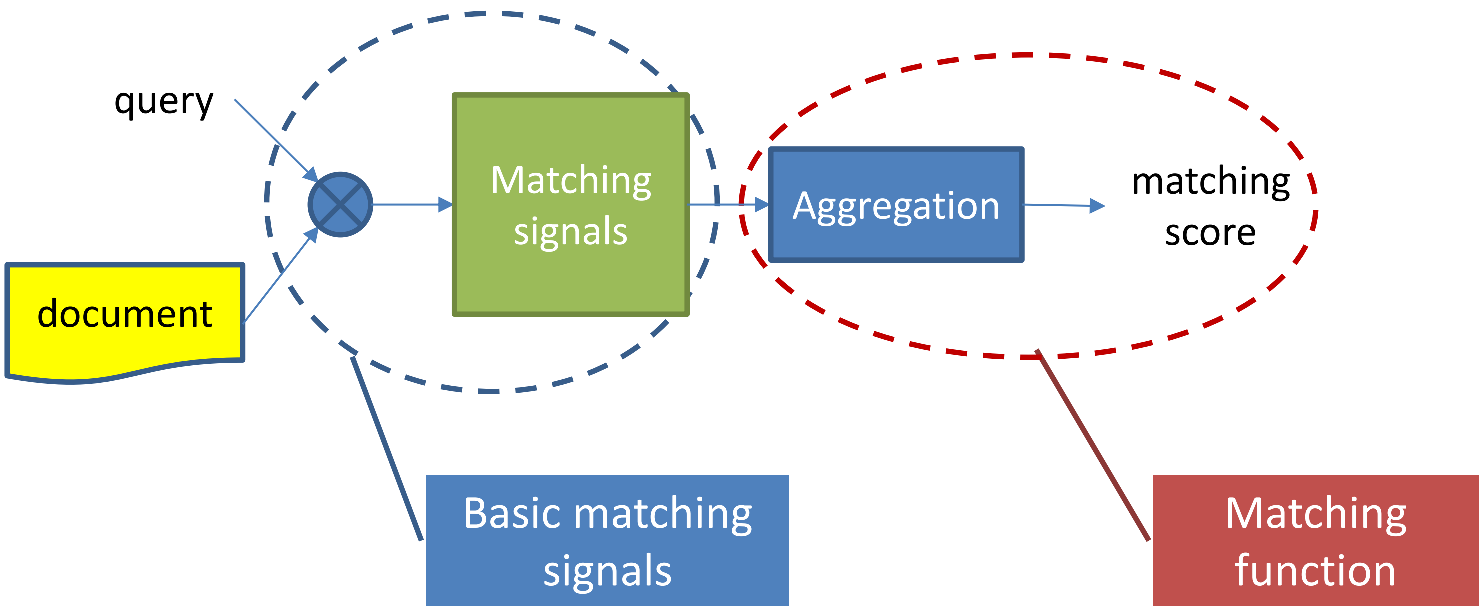
- Two steps
  - 1. Calculate representations for query and document
  - 2. Conduct matching
- Representations for query and document
  - Using DNN
  - Using CNN and RNN to capture order information
- Matching function
  - Dot product (cosine similarity)
  - Multi-layer Perceptron
  - Neural tensor networks



METHODS OF MATCHING FUNCTION LEARNING

Matching Function Learning

- Step 1: construct basic low-level matching signals
- Step 2: aggregate matching patterns

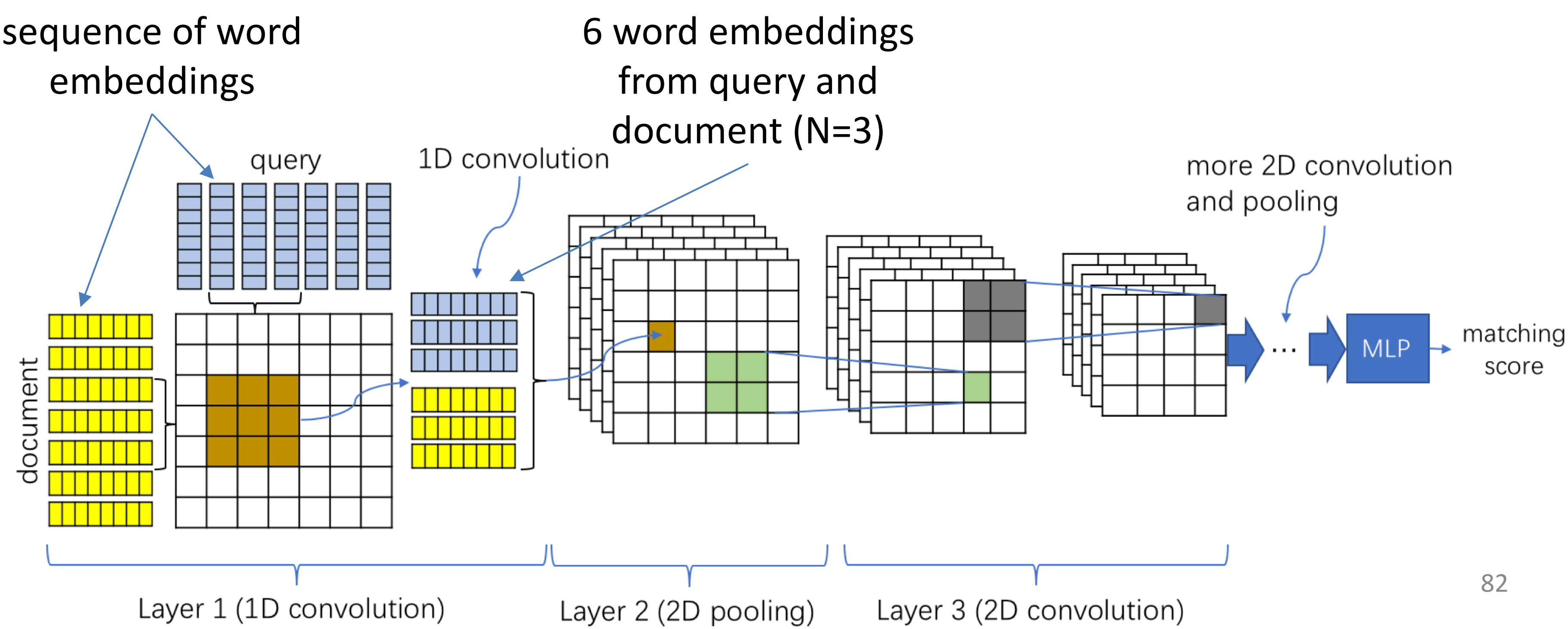


Typical Matching Function Learning Methods

- Matching with word-level similarity matrix
  - ARC II (Hu et al., NIPS ’14)
  - MatchPyramid (Pang et al., AAAI ’16)
  - Match-SRNN (Wan et al., IJCAI ’16)
- Matching with attention model
  - (Parikh et al., EMNLP ‘16)
- Combining matching function learning and representation learning
  - Duet (Mitra et al., WWW ’17)

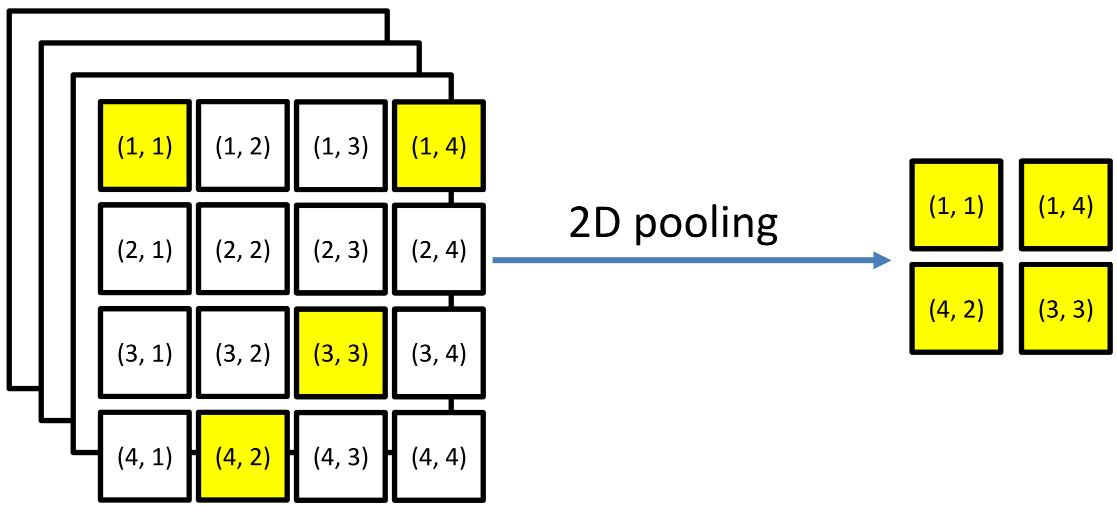
ARC-II (Hu et al., NIPS '14)

- Let two sentences meet before their own high-level representations mature
- Basic matching signals: phrase sum interaction matrix
- Interaction: CNN to capture the local interaction structure
- Aggregation function: MLP



ARC-II (cont')

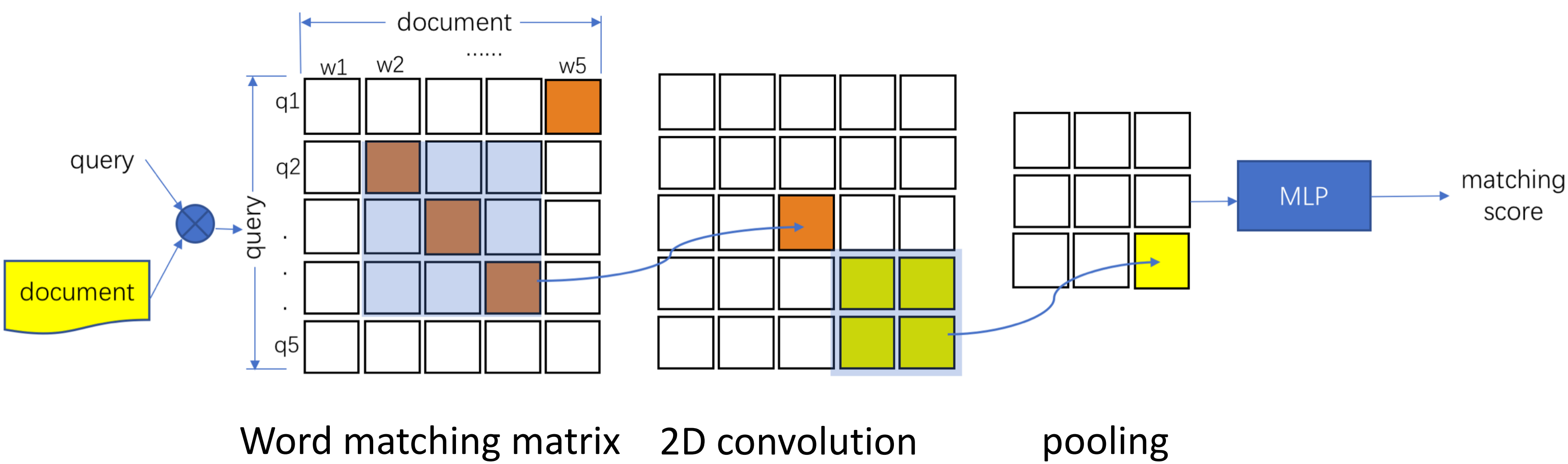
- Keeping word order information
  - Both the convolution and pooling are order preserving



- However, word level exact matching signals are lost
  - 2-D matching matrix is constructed based on the embedding of the words in two N-grams

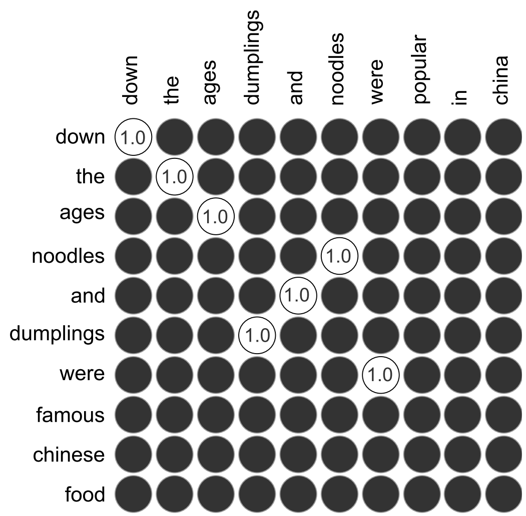
MatchPyramid (Pang et al., AAAI '16)

- Inspired by image recognition
- Basic matching signals: word-level matching matrix
- Matching function: 2D convolution + MDP

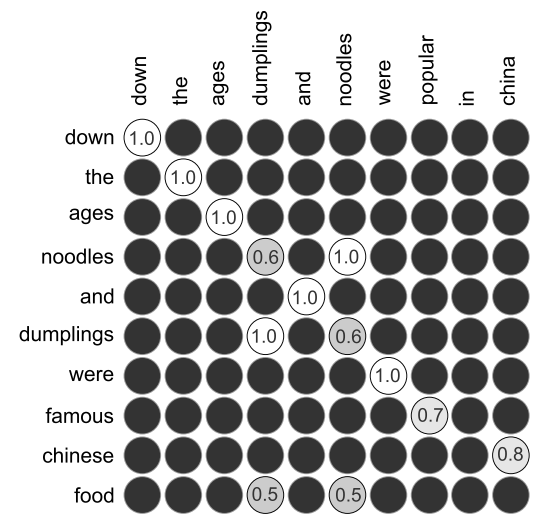


Matching Matrix: Basic Matching Signals

- Each entry calculated based on
  - Word-level exact matching (0 or 1)
  - Semantic similarity based on embeddings of words



Exact match

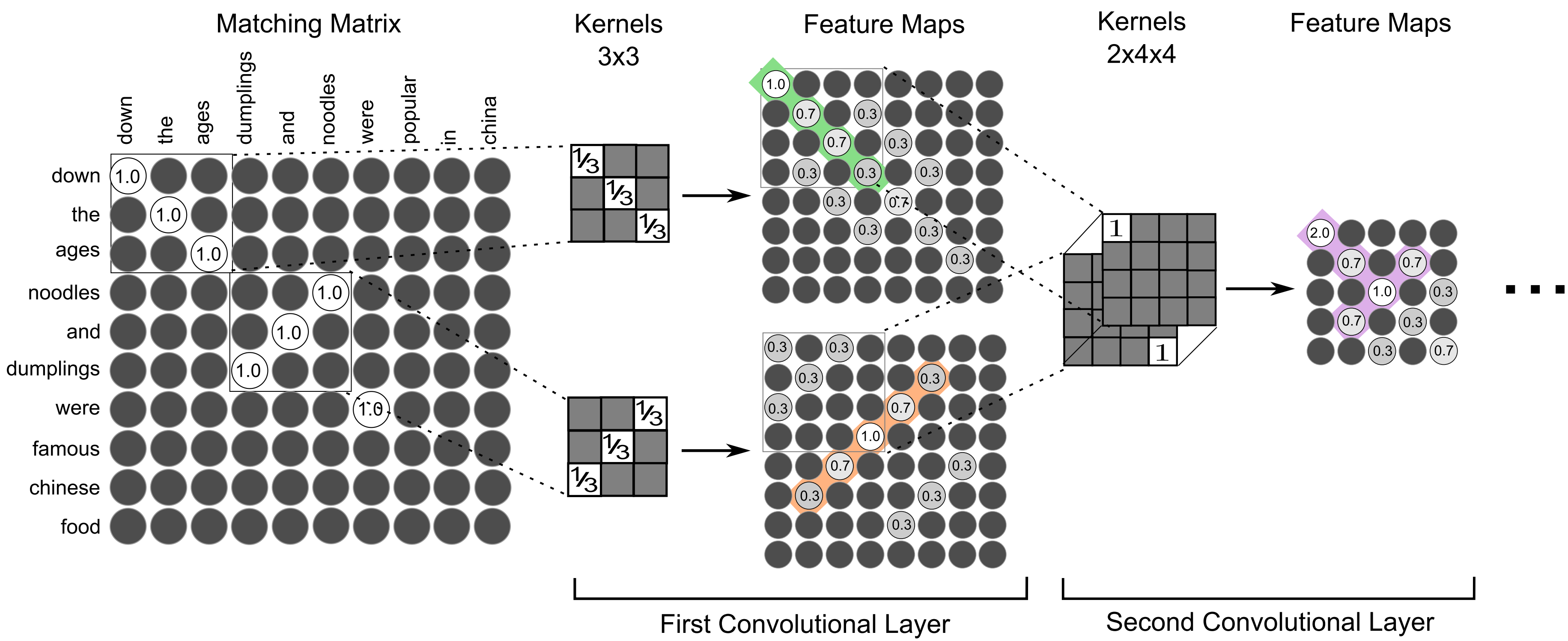


Cosine similarity

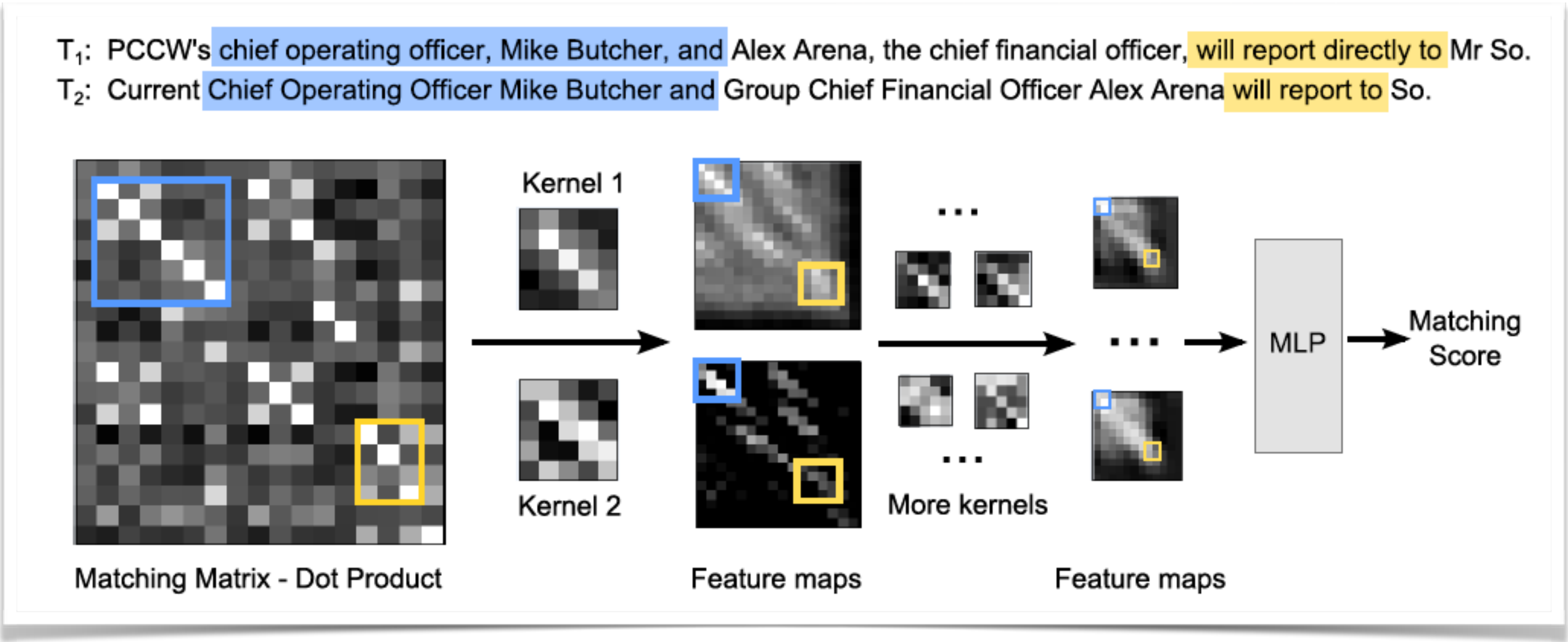
- Positions information of words is kept

Matching Function: 2D Convolution

- Discovering the matching patterns with CNN, stored in the kernels

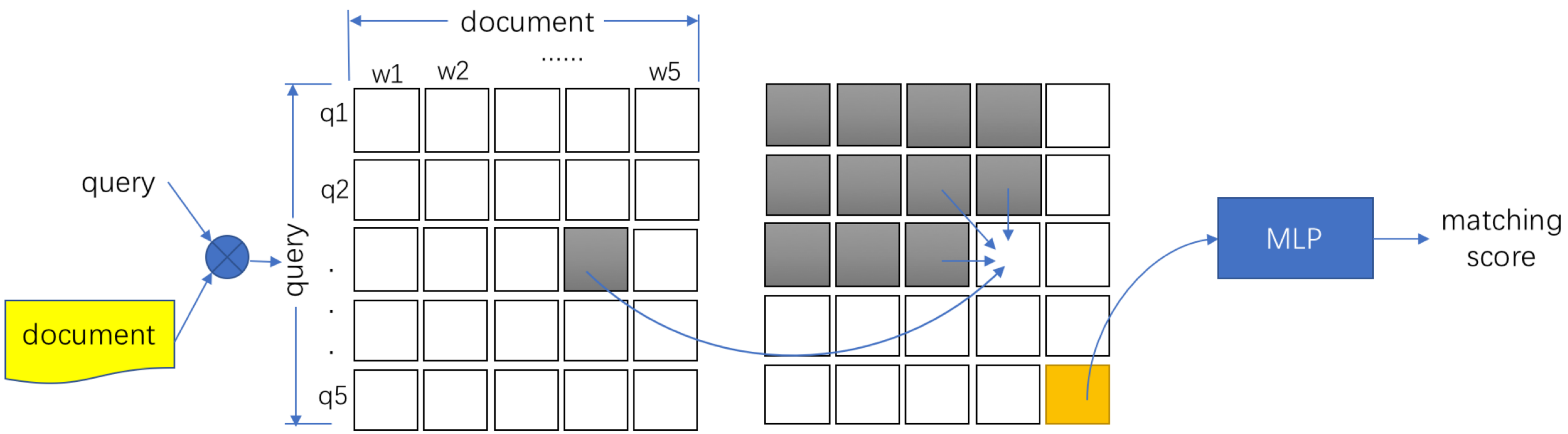


Discovered Matching Patterns



Match-SRNN (Wan et al., IJCAI '16)

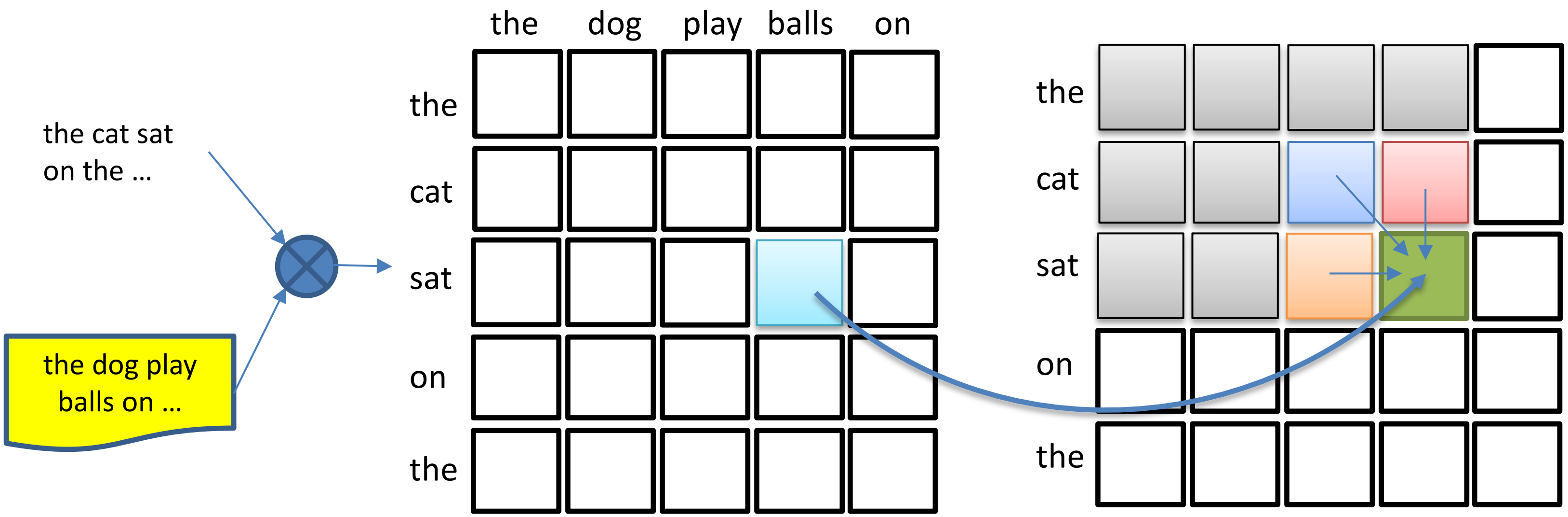
- Based on spatial recurrent neural network (SRNN)
- Basic matching signals: word-level matching matrix
- Matching function: Spatial RNN + MLP

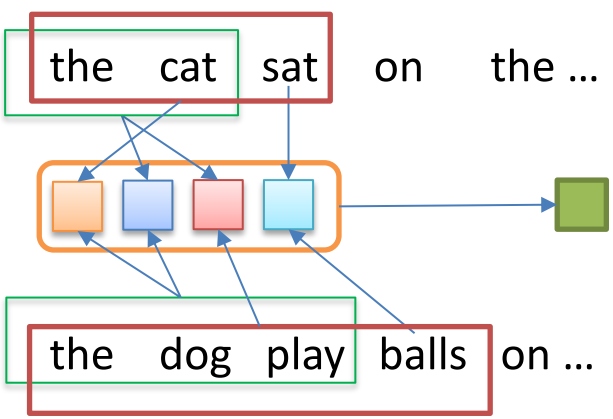


Matching matrix

Spatial RNN

Match-SRNN: Recursive Matching Structure

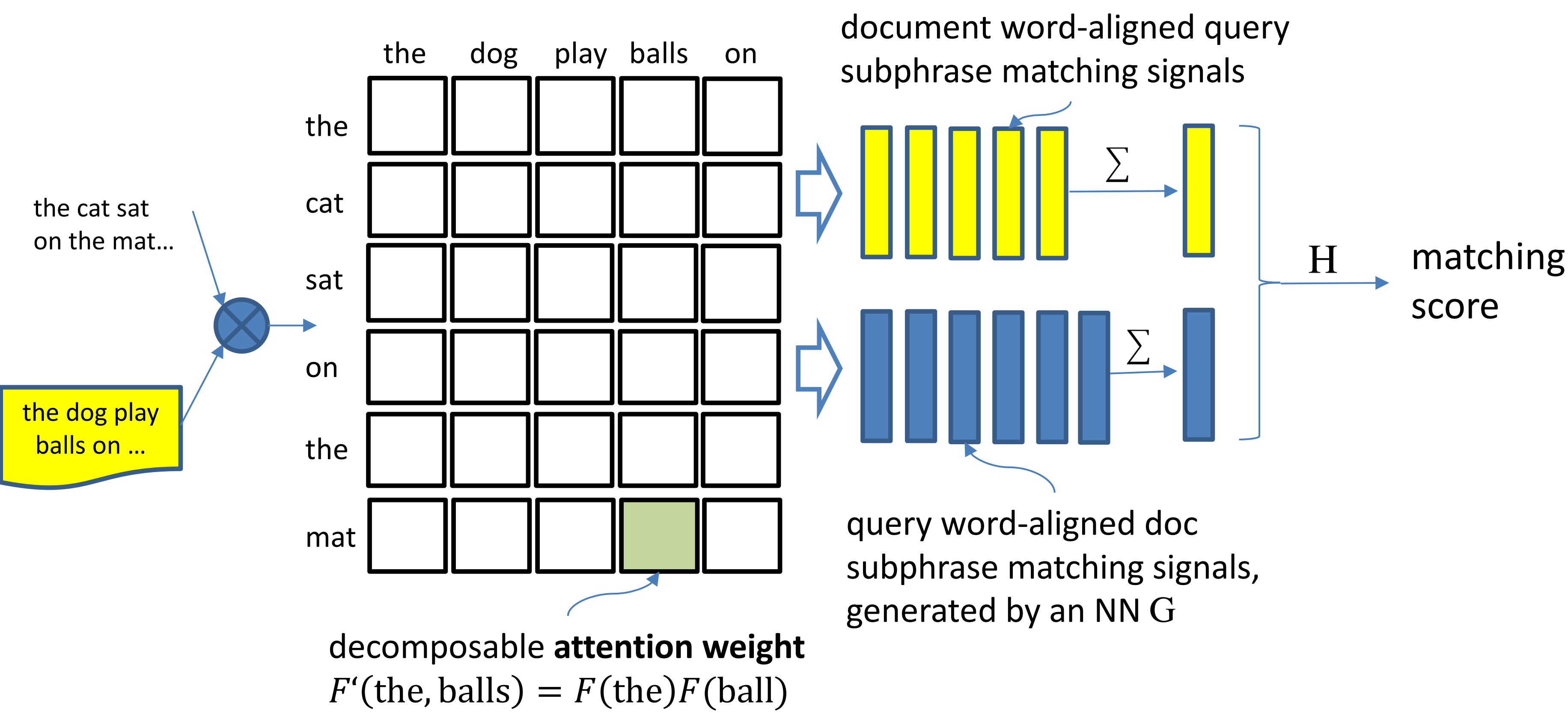




- Calculated recursively (from top left to bottom right)
- All matching signals between the prefixes been utilized
  - **Current position**: sat <—> balls
  - **Substrings**:
    - the cat <—> the dog play
    - the cat <—> the dog play balls
    - the cat sat <—> the dog play

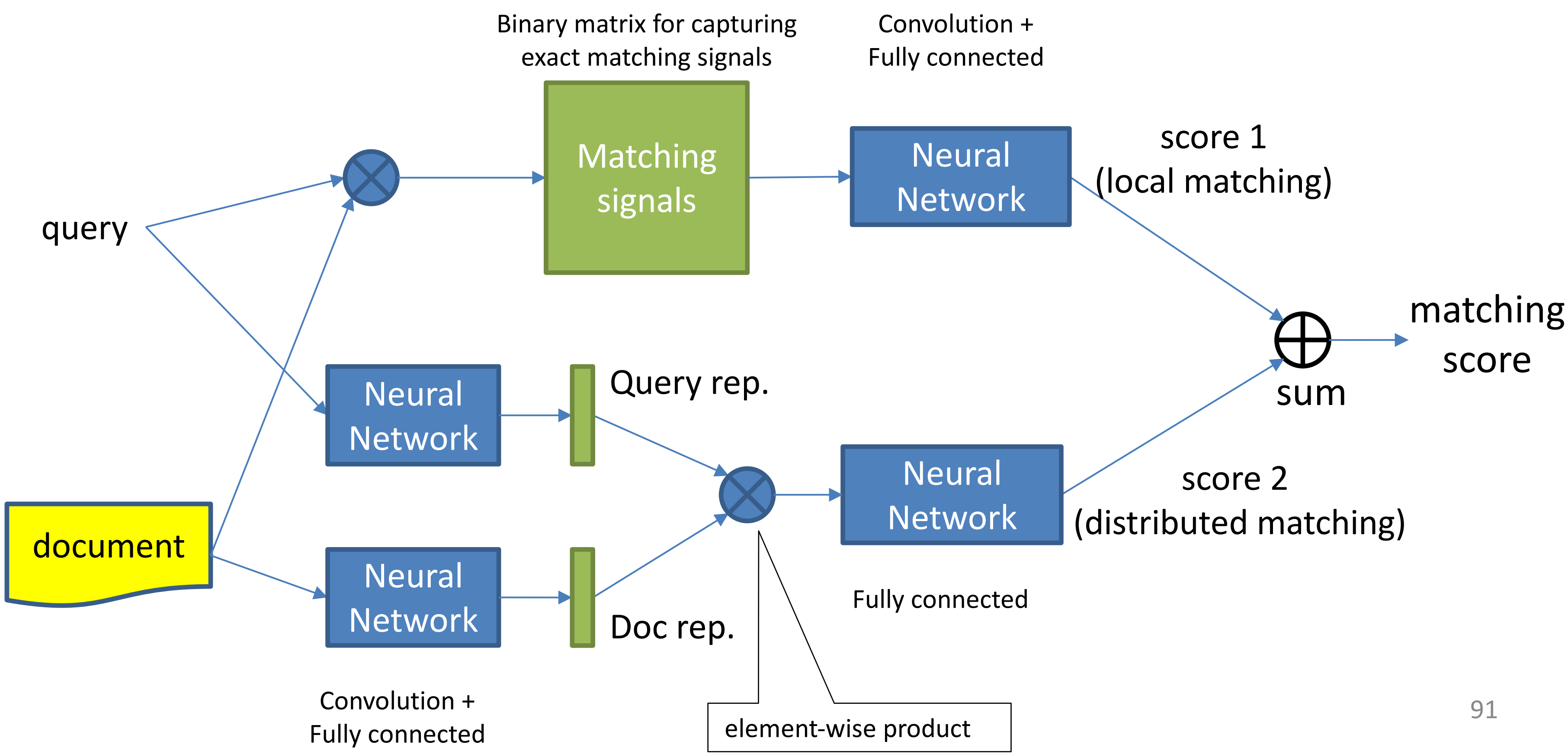
Decomposable Attention Model for Matching (Parikh et al., EMNLP '16)

- Based on decomposable attention model
- Three steps: attend -> compare -> aggregate
  - **Attend**: soft-align words of query and document
  - **Compare**: separately compare word-aligned subphrase, get matching signals
  - **Aggregate**: aggregate the matching signals for produce final matching score



Representation Learning + Matching Function Learning (Duet, Mitra et al., WWW '17)

- Hypothesis: matching with distributed representations complements matching with local representations
  - Local matching: matching function learning
  - Distributed matching: representation learning



Experimental Evaluation

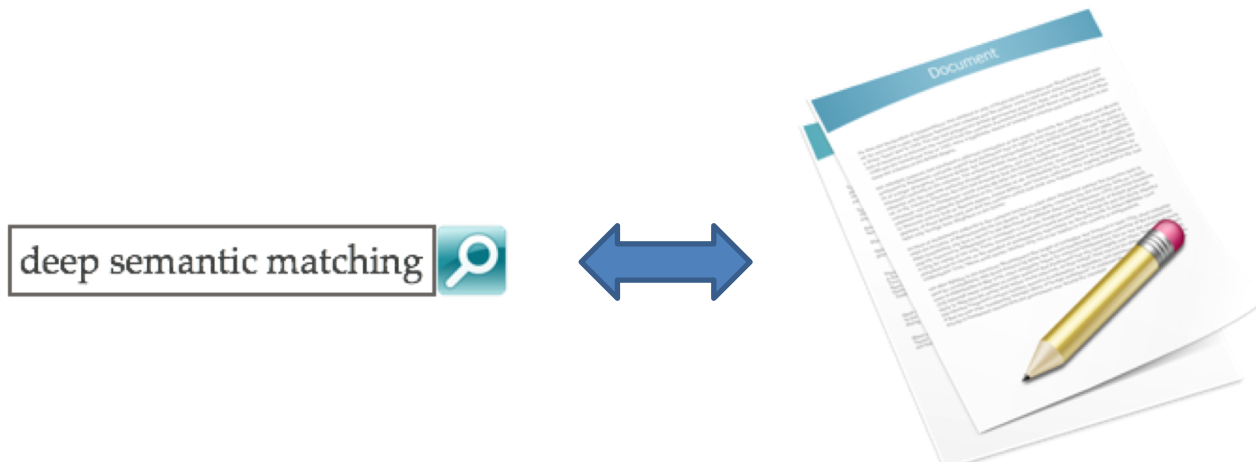
	Method	P@1	MRR
Traditional IR	BM25	0.579	0.457
Representation Learning methods	ARC-I	0.581	0.756
	CNTN	0.626	0.781
	LSTM-RNN	0.690	0.822
Matching Function Learning	ARC-II	0.591	0.765
	MatchPyramid	0.764	0.867
	Match-SRNN	0.790	0.882

Based on Yahoo! Answers dataset (60,564 question-answer pairs)

- Matching function learning based methods outperformed the representation learning ones

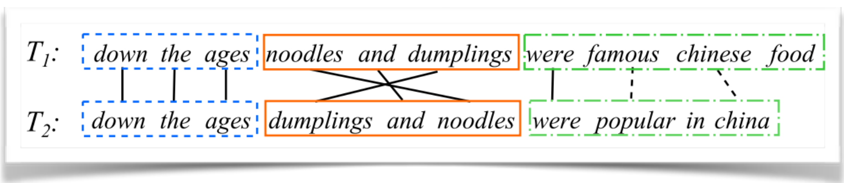
Short Summary

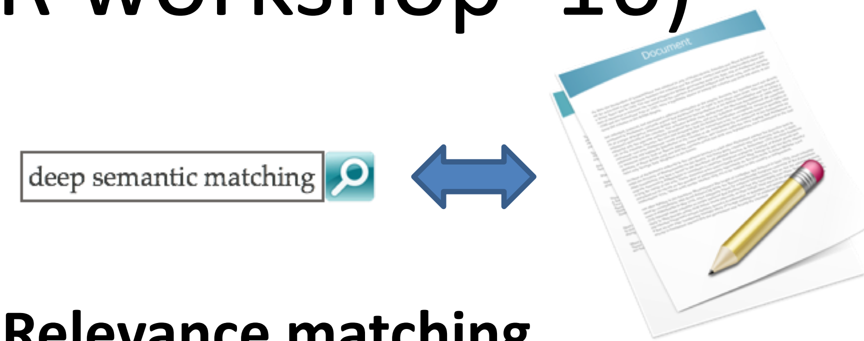
- Two steps
  - 1. Construct basic matching signals
  - 2. Aggregate matching patterns
- Basic matching signals
  - Similarity matrix (exact match, dot product, cosine similarity)
  - Attention weights
- Aggregate matching patterns
  - CNN
  - Spatial RNN
  - MLP
- Combining representation learning (inexact match) and matching function learning (exact match)



QUERY-DOCUMENT RELEVANCE MATCHING

Similarity ≠ Relevance
(Pang et al., Neu-IR workshop '16)





Similarity matching

- Whether two sentences are semantically similar
- Homogeneous texts with comparable lengths
- Matches at all positions of both sentences
- Symmetric matching function
- Representative task: Paraphrase Identification

Relevance matching

- Whether a document is relevant to a query
- Heterogeneous texts (keywords query, document) and very different in lengths
- Matches in different parts of documents
- Asymmetric matching function
- Representative task: ad-hoc retrieval

Typical Query-Document Relevance Matching Methods

- Based on global distribution of matching strengths
  - DRMM (Guo et al., CIKM ’16)
  - aNMM (Yang et al., CIKM ‘16)
  - K-NRM (Xiong et al., SIGIR ‘17)
  - Conv-KNRM (Dai et al., WSDM ‘18)
- Based on local context of matched terms
  - DeepRank (Pang et al., CIKM ‘17)
  - PACRR (Hui et al., EMNLP ’17)

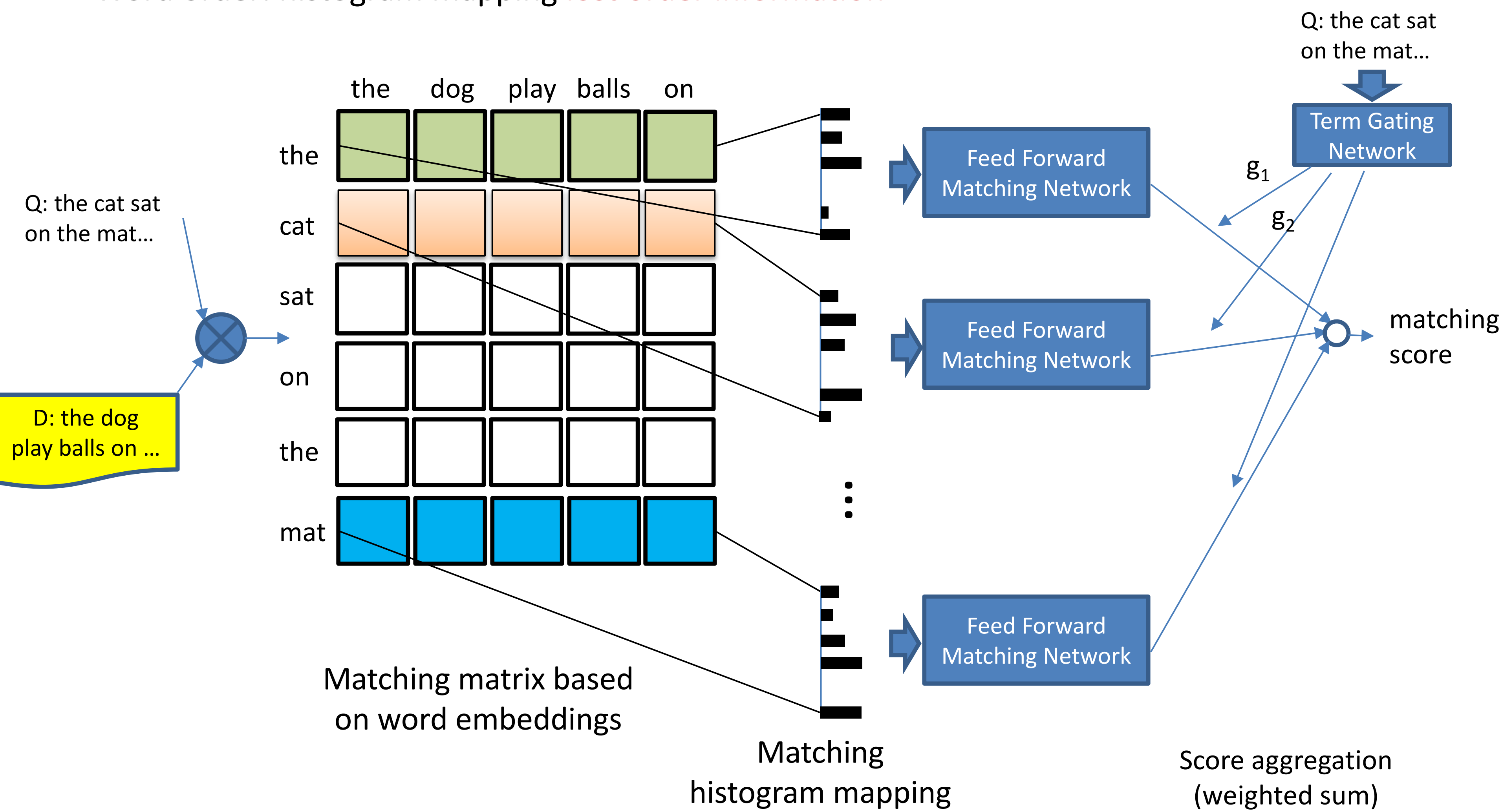
Relevance Matching based on Global Distribution of Matching Strengths

- Step 1: for each query term
  - Calculate its matching signals among the document
  - Calculate the global matching strength distributions
- Step 2: aggregate the distributions

- Advantages
  - Conducting matching between **short query** and **long document**
  - Matching strength distributions are robust, compared with the raw matching signals
- Disadvantage
  - **Lost term order** information when calculating the distributions

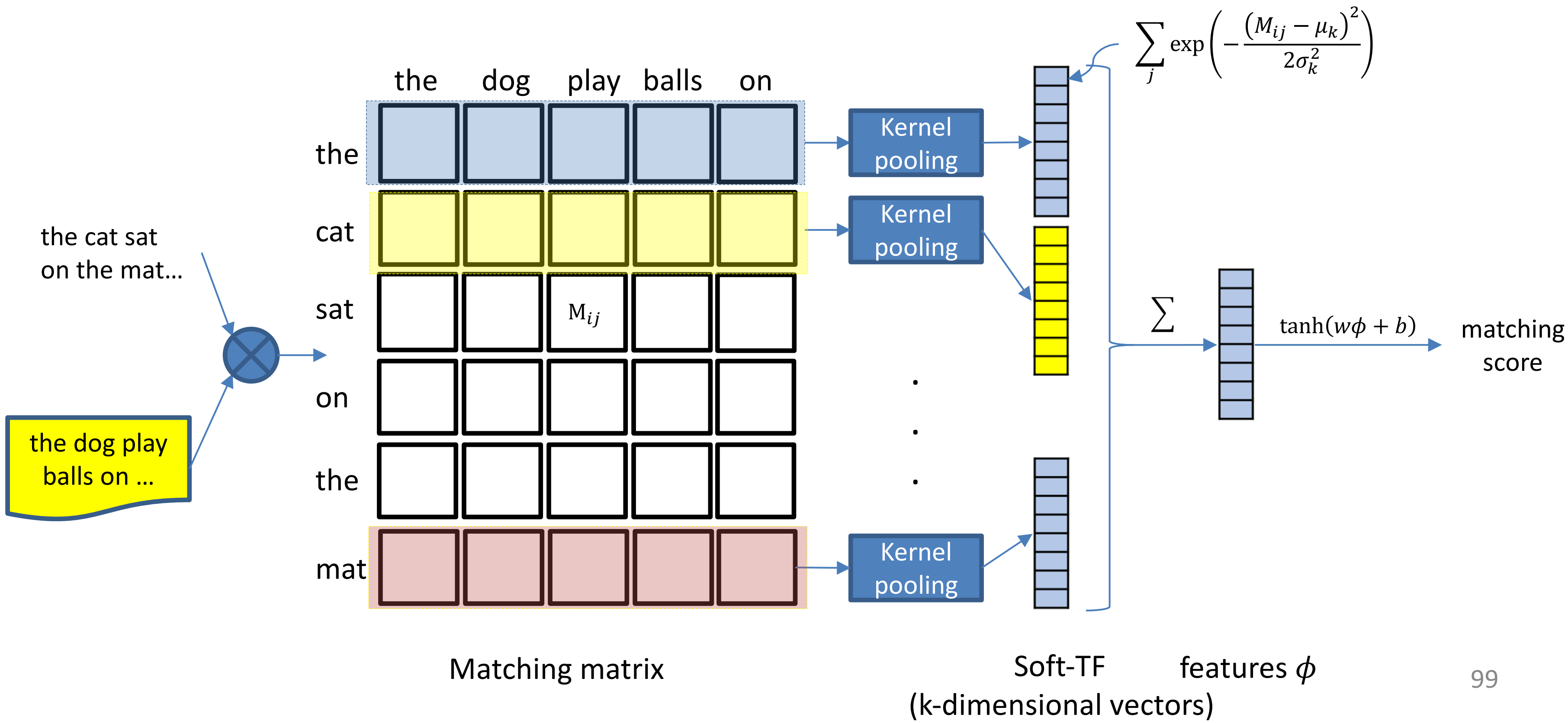
Deep Relevance Matching Model (Guo et al., CIKM ’16)

- Basic matching signals: cosine similarity of word embeddings → matching strength histogram
- Ranking function: Neural Network + Term Gating Network
- Semantic gap: embeddings bridge the semantic gap
- Word order: histogram mapping lost order information



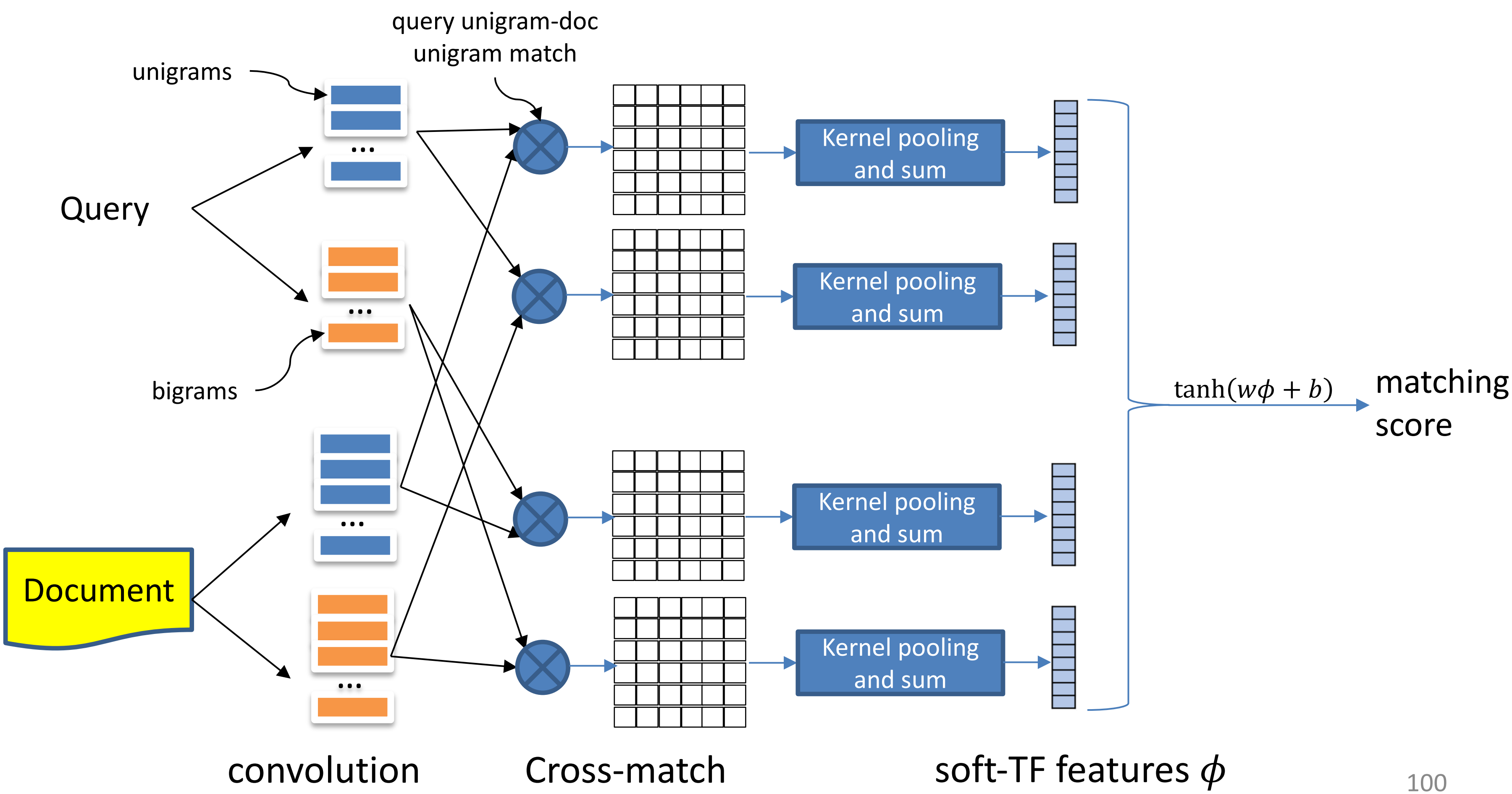
K-NRM: Kernel Pooling as Matching Function (Xiong et al., SIGIR '17)

- Basic matching signals: cosine similarity of word embeddings
- Ranking function: kernel pooling + nonlinear feature combination
- Semantic gap: embedding bridge the semantic gap
- Word order: kernel pooling and sum operations lost order information



Conv-KNRM (Dai et al., WSDM '18)

- Based on KNRM
- N-gram cross-matching to capture local word order information



Experimental Evaluation

	Method	NDCG@1	NDCG@10
Traditional IR / Learning to rank	BM25	0.142	0.287
	RankSVM	0.146	0.309
Representation Learning Methods	CDSSM	0.144	0.333
Matching Function Learning	MatchPyramid	0.218	0.379
Query-Document Relevance Matching	DRMM	0.137	0.315
	K-NRM	0.264	0.428
	Conv-KNRM	**0.336**	**0.481**

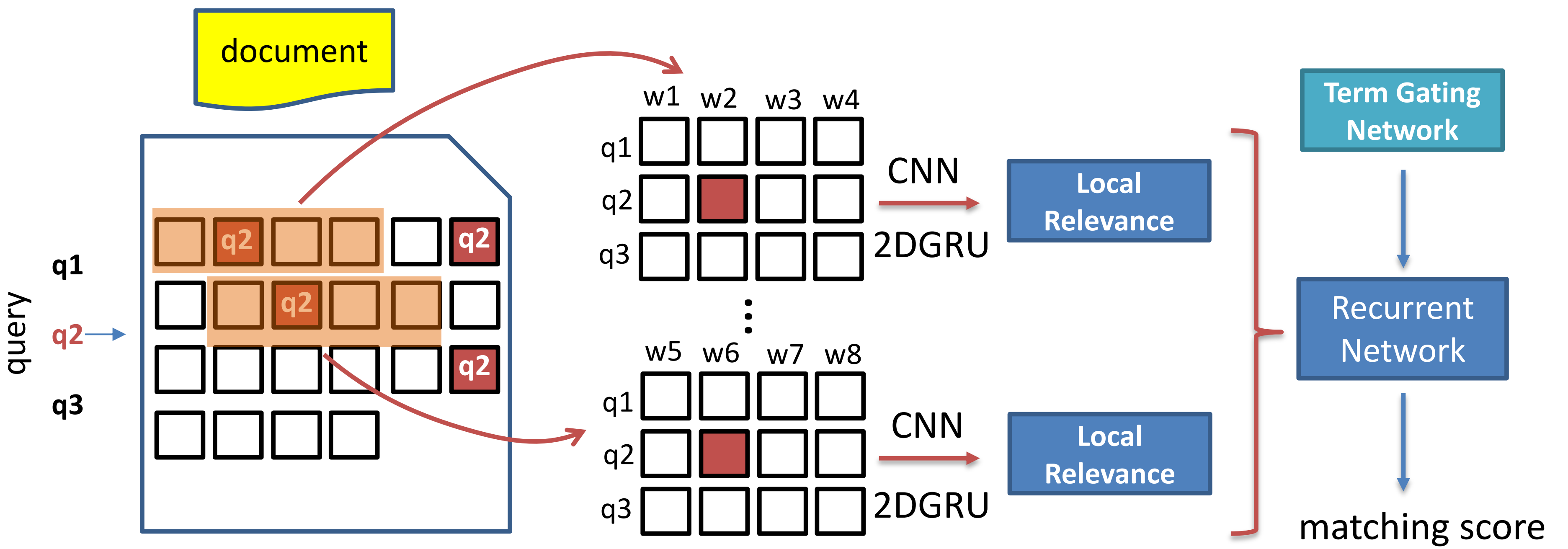
Results reported in (Dai et al., WSDM '18), based on Sogou Log dataset (95,229 queries)

Relevance Matching based on Local Context of Matched Terms

- Step 1: for each query term
  - Identify its local contexts (area around exact matching positions) among the document
  - Conduct matching between query and local contexts
- Step 2: aggregate the local matching signals
- Advantages:
  - Matching between short query and long document text
  - Robust: filter out irrelevant document contents
  - Keep order information within each local context

DeepRank (Pang et al., CIKM '17)

- Calculate relevance by mimicking the human relevance judgement process



1. Detecting Relevance locations: focusing on locations of query terms when scanning the whole document

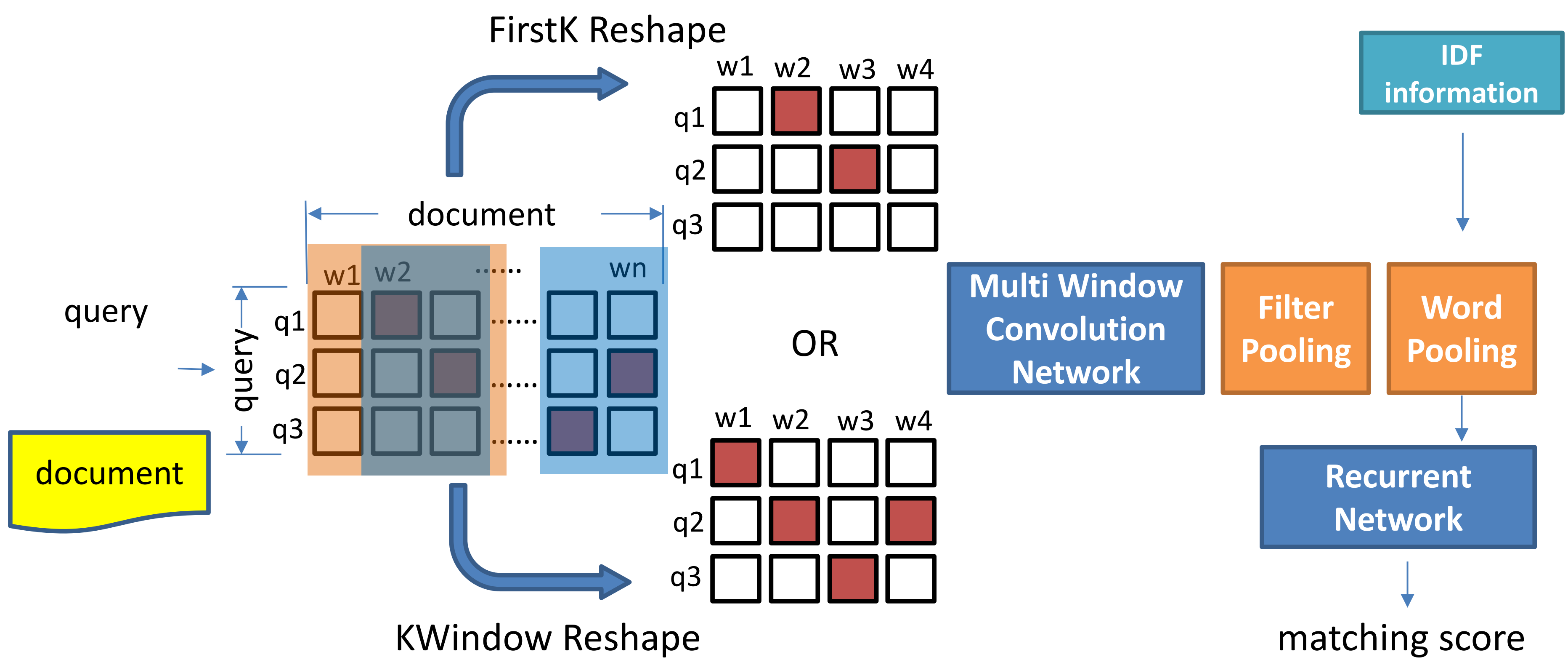
2. Determining local relevance: relevance between query and each location context, using MatchPyramid / MatchSRNN etc.

3. Matching signals aggregation:

$$F(\mathbf{q}, \mathbf{d}) = \sum_{w \in \mathbf{q}} (E_w \mathbb{I})^T \cdot \mathcal{T}(w)$$

Position-Aware Neural IR Model (PACRR, Hui et al., EMNLP '17)

- Hypothesis: relevance matching is determined by some positions in documents
  - First k words in document (FirstK), or
  - Most similar context positions in document (Kwindow)



Experimental Evaluation

	Method	NDCG@20	ERR@20
Traditional IR	QL	0.231	0.131
Matching Function Learning	MatchPyramid	0.278	0.176
Global Distribution of Matching Signals	DRMM	0.300	0.193
	K-NRM	0.324	0.201
	DUET	0.267	0.179
Local Context of Matched Terms	PACRR-firstk	0.339	0.221

Results reported in (Hui et al., EMNLP ‘17), based on Web Track 14 dataset.

Short Summary

- Methods based on global distributions of matching strengths
  - 1. calculating term matching strength distributions
  - 2. aggregating the distributions to a matching score
- Methods based on local context of matched terms
  - 1. Identifying the relevance locations / contexts
  - 2. Matching the whole query with the local contexts
  - 3. Aggregating the local matching signals

References

- Clark J. Google turning its lucrative web search over to ai machines[J]. Bloomberg Technology. Publicado em, 2015, 26.
- Metz C. AI is transforming Google search[J]. The rest of the web is next. WIRED Magazine, 2016.
- Huang P S, He X, Gao J, et al. Learning deep structured semantic models for web search using clickthrough data[C]//Proceedings of the 22nd ACM international conference on Conference on information & knowledge management. ACM, 2013: 2333-2338.
- Hu B, LuShen Y, He X, Gao J, et al. A latent semantic model with convolutional-pooling structure for information retrieval[C]//Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management. ACM, 2014: 101-110.
- Z, Li H, et al. Convolutional neural network architectures for matching natural language sentences[C]//Advances in neural information processing systems. 2014: 2042-2050.
- Qiu X, Huang X. Convolutional Neural Tensor Network Architecture for Community-Based Question Answering[C]//IJCAI. 2015: 1305-1311.
- Palangi H, Deng L, Shen Y, et al. Deep sentence embedding using long short-term memory networks: Analysis and application to information retrieval[J]. IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP), 2016, 24(4): 694-707.
- Yin W, Schütze H. Multigrancnn: An architecture for general matching of text chunks on multiple levels of granularity[C]//Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers). 2015, 1: 63-73.
- Socher R, Huang E H, Pennin J, et al. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection[C]//Advances in neural information processing systems. 2011: 801-809.
- Wan S, Lan Y, Guo J, et al. A Deep Architecture for Semantic Matching with Multiple Positional Sentence Representations[C]//AAAI. 2016, 16: 2835-2841.

References

- Pang L, Lan Y, Guo J, et al. Text Matching as Image Recognition[C]//AAAI. 2016: 2793-2799.
- Shengxian Wan, Yanyan Lan, Jun Xu, Jiafeng Guo, Liang Pang, and Xueqi Cheng. 2016. Match-SRNN: modeling the recursive matching structure with spatial RNN. In Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence (IJCAI'16), 2922-2928.
- Ankur P. Parikh, Oscar Tackstrom, Dipanjan Das, and Jakob Uszkoreit. A Decomposable Attention Model for Natural Language Inference. In Proceedings of EMNLP, 2016.
- Zhuyun Dai, Chenyan Xiong, Jamie Callan, and Zhiyuan Liu. Convolutional Neural Networks for Soft-Matching N-Grams in Ad-hoc Search. In Proceedings of WSDM 2018.
- Chenyan Xiong, Zhuyun Dai, Jamie Callan, Zhiyuan Liu, Russell Power. End-to-End Neural Ad-hoc Ranking with Kernel Pooling. In Proceedings of SIGIR 2017.
- Bhaskar Mitra, Fernando Diaz, and Nick Craswell. Learning to match using local and distributed representations of text for web search. In Proceedings of WWW 2017.
- Jiafeng Guo, Yixing Fan, Qiqing Yao, W. Bruce Croft, A Deep Relevance Matching Model for Ad-hoc Retrieval. In Proceedings of CIKM 2016.
- Liu Yang, Qingyao Ai, Jiafeng Guo, W. Bruce Croft, aNMM: Ranking Short Answer Texts with Attention-Based Neural Matching Model. In Proceedings of CIKM 2016.
- Liang Pang, Yanyan Lan, Jiafeng Guo, Jun Xu and Xueqi Cheng. DeepRank: a New Deep Architecture for Relevance Ranking in Information Retrieval. In Proceedings of CIKM 2017.
- Qin Chen, Qinmin Hu, Jimmy Xiangji Huang, Liang He. CA-RNN: Using Context-Aligned Recurrent Neural Networks for Modeling Sentence Similarity. . In Proceedings of AAAI 2018.
- Liang Pang, Yanyan Lan, Jiafeng Guo, Jun Xu, Xueqi Cheng. A Study of MatchPyramid Models on Ad-hoc Retrieval. In Proceedings of SIGIR 2016 Neu-IR Workshop.
- Kai Hui, Andrew Yates, Klaus Berberich, and Gerard de Melo. Co-PACRR: A Context-Aware Neural IR Model for Ad-hoc Retrieval. WSDM '18, 279-287.

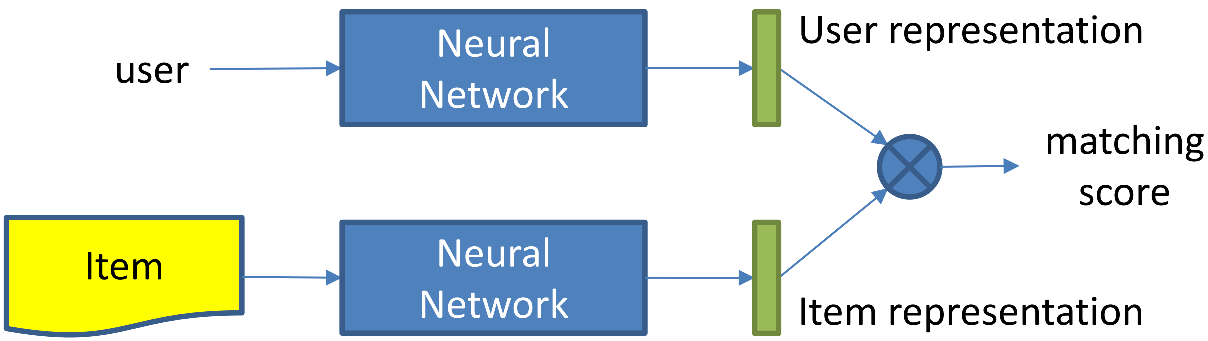
Outline of Tutorial

- Unified View of Matching in Search and Recommendation
- Part 1: Traditional Approaches to Matching
- Part 2: Deep Learning Approaches to Matching
  - Deep matching models for search
  - Deep matching models for recommendation
- Summary

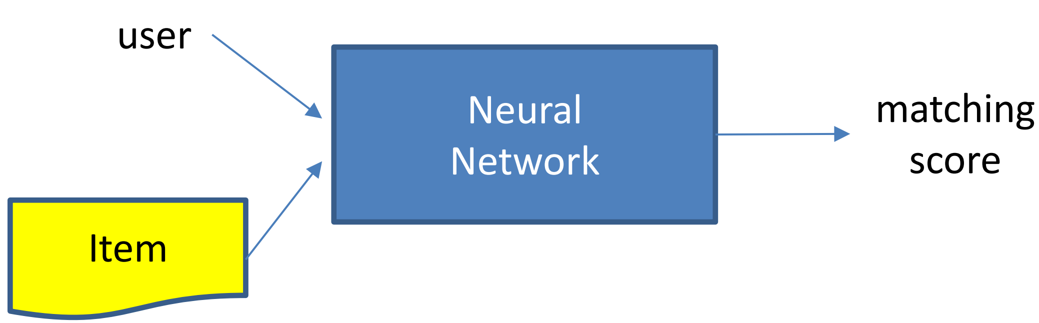
Slides: http://comp.nus.edu.sg/~xiangnan/sigir18-deep.pdf

Deep Matching Models for Recommendation

- Methods of representation learning



- Methods of matching function learning



Methods of Representation Learning

1. Collaborative Filtering:

   Models are built based on user-item interaction matrix only.

   - **DeepMF**: Deep Matrix Factorization (Xue et al, IJCAI'17)
   - **AutoRec**: Autoencoders Meeting CF (Sedhain et al, WWW'15)
   - **CDAE**: Collaborative Denoising Autoencoder (Wu et al, WSDM'16)

2. Collaborative Filtering + Side Info:

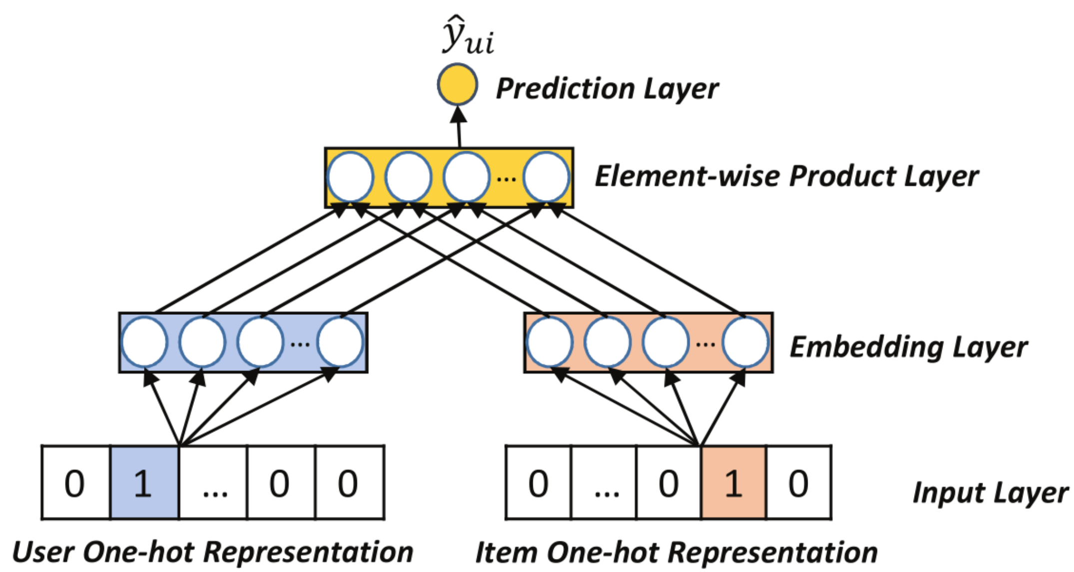
   Models are built based on user-item interaction + side info.

   - **DCF**: Deep Collaborative Filtering via Marginalized DAE (Li et al, CIKM'15)
   - **DUIF**: Deep User-Image Feature (Geng et al, ICCV'15)
   - **ACF**: Attentive Collaborative Filtering (Chen et al, SIGIR'17)
   - **CKB**: Collaborative Knowledge Base Embeddings (Zhang et al, KDD'16)

Recap MF as a Neural Network

- **Input**: user -> ID (one-hot), item -> ID (one-hot).
- **Representation Function**: linear embedding layer.
- **Matching Function**: inner product.

$$f_{MF}(u, i|\mathbf{p}_u, \mathbf{q}_i) = \mathbf{p}_u^{\top}\mathbf{q}_i = \sum_{k=1}^{K} p_{uk} q_{ik},$$



Deep Matrix Factorization (Xue et al, IJCAI'17)

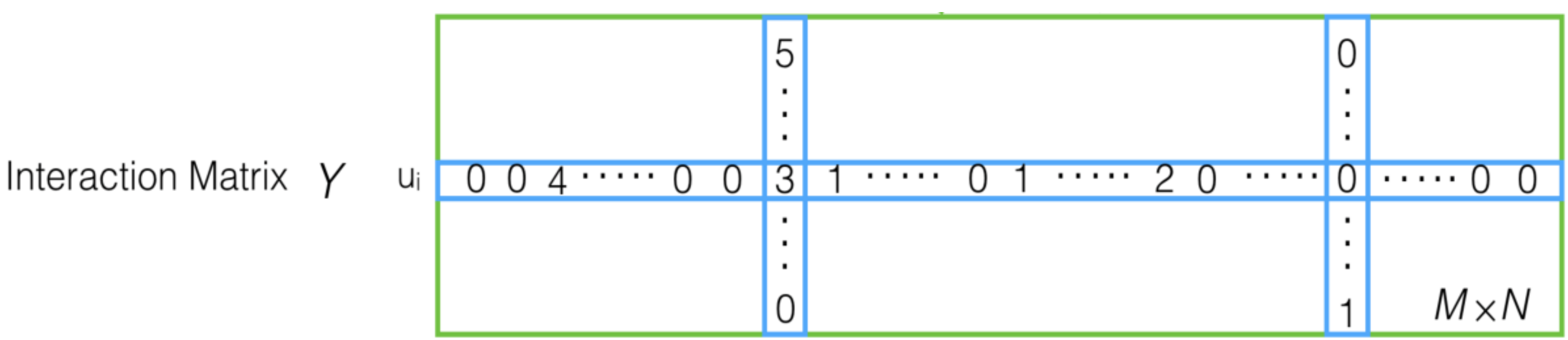
- **Input**:

  user -> items that she has rated (multi-hot), i.e., row vector of Y

  indicates the user's global preference

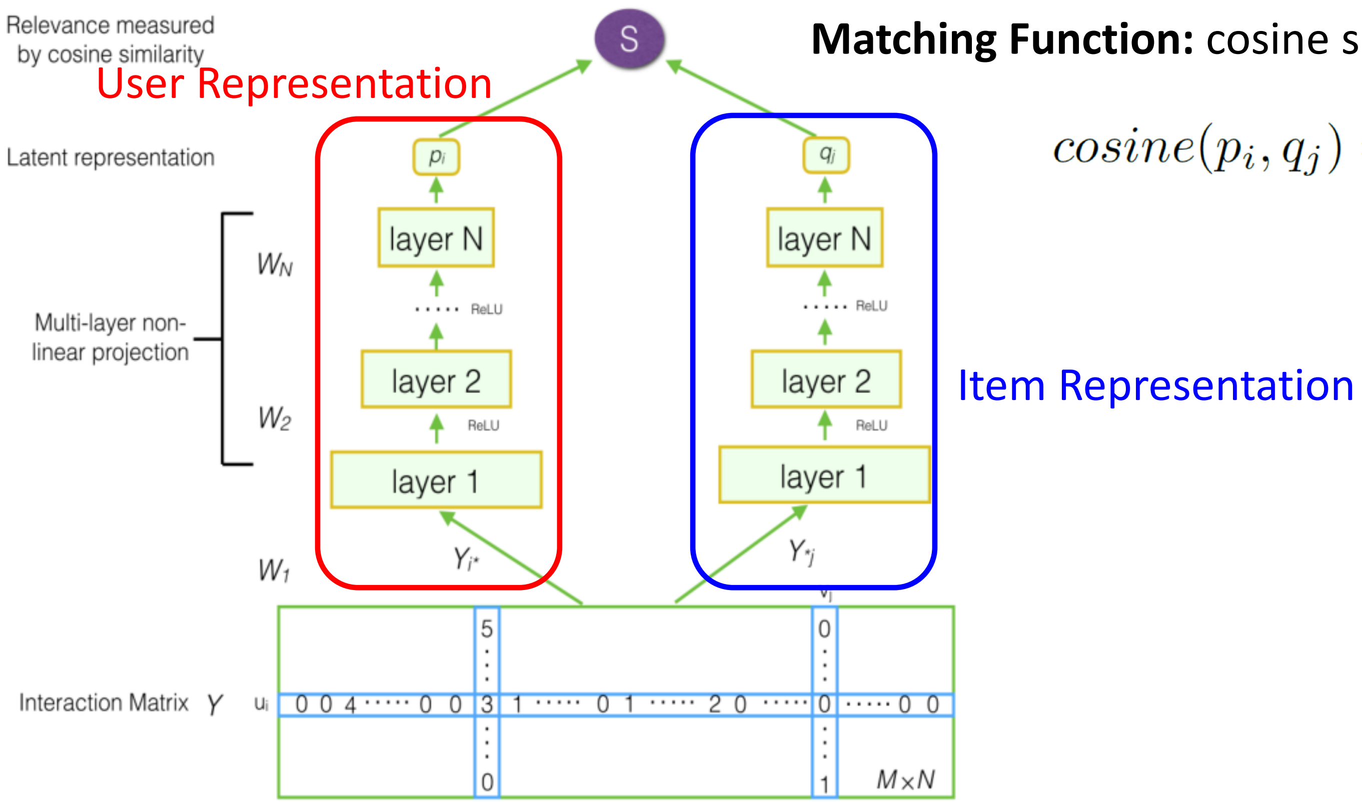
  item -> users who have rated it (multi-hot), i.e., column vector of Y

  indicates the item's rating profile.



Deep Matrix Factorization (Xue et al, IJCAI’17)

- **Representation Function**:
  - Multi-Layer Perceptron



Matching Function: cosine similarity

$$cosine(p_i, q_j) = \frac{p_i^T q_j}{||p_i||\ ||q_j||}$$

AutoRec (Sedhain et al, WWW'15)

- **Input: (similar to DeepMF)**

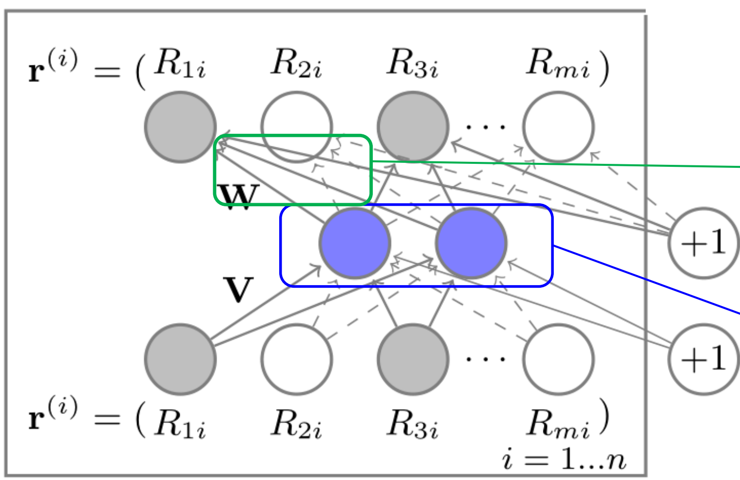
  user -> historically rated items (user-based autoencoder).

  item-> ID

- **Representation Function:** Multi-Layer Perceptron
- **Matching Function**: inner product

Input reconstruction: $h(\mathbf{r};\theta) = f\left(\mathbf{W}\cdot g(\mathbf{V}\mathbf{r}+\boldsymbol{\mu})+\mathbf{b}\right)$

$$\min_{\theta} \sum_{i=1}^{n} \boxed{||\mathbf{r}^{(i)} - h(\mathbf{r}^{(i)};\theta))||_{\mathcal{O}}^2} + \frac{\lambda}{2}\cdot(||\mathbf{W}||_F^2 + ||\mathbf{V}||_F^2),$$



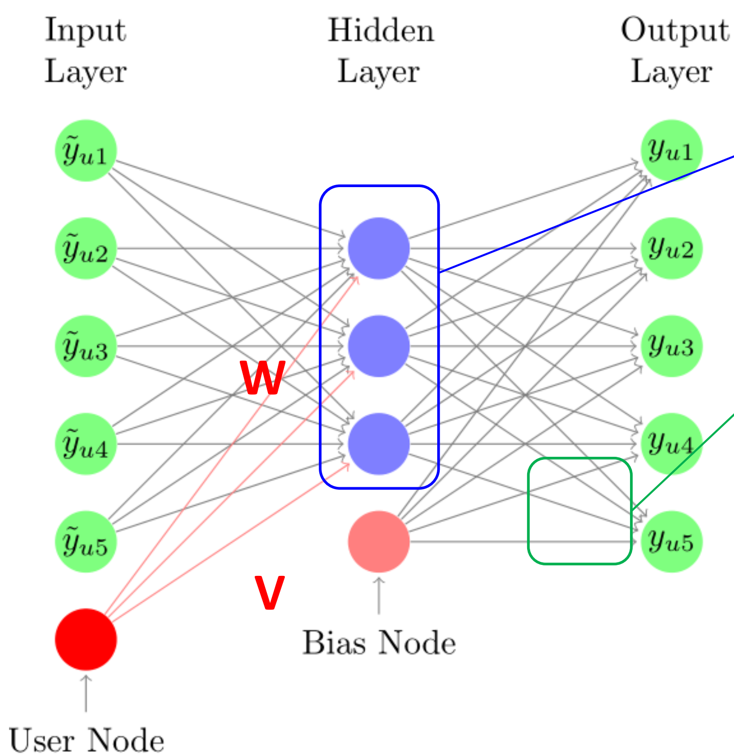
Output weights denote item representation

Hidden neurons denote user representation

user-based autoencoder

Collaborative Denoising Auto-Encoder (Wu et al, WSDM’16)

- **Input:**
  user -> ID & historically rated items (similar to SVD++)
  item -> ID
- **Representation Function**: Multi-Layer Perceptron



Hidden neurons are user representation:

$$\boldsymbol{z}_u = h\left(\boldsymbol{W}^\top \tilde{\boldsymbol{y}}_u + \boldsymbol{V}_u + \boldsymbol{b}\right),$$

Weights of output layer are item representation

$$\hat{y}_{ui} = f\left(\boldsymbol{W}_i'^\top \boldsymbol{z}_u + b_i'\right),$$

Short Summary

- Either ID or history is used as the profile of user/item
- Models with history as input are more expressive, but are also more expensive to train.

- The Auto-Encoder architecture is essentially identical to

$$\underbrace{\text{MLP (representation learning)}}_{\text{Nonlinear}} + \underbrace{\text{MF (matching function)}}_{\text{Linear}}.$$

Methods of Representation Learning

1. Collaborative Filtering:

   Models are built based on user-item interaction matrix only.

   - **DeepMF**: Deep Matrix Factorization (Xue et al, IJCAI'17)
   - **AutoRec**: Autoencoders Meeting CF (Sedhain et al, WWW'15)
   - **CDAE**: Collaborative Denoising Autoencoder (Wu et al, WSDM'16)

2. Collaborative Filtering + side info:

   Models are built based on user-item interaction matrix + side info.

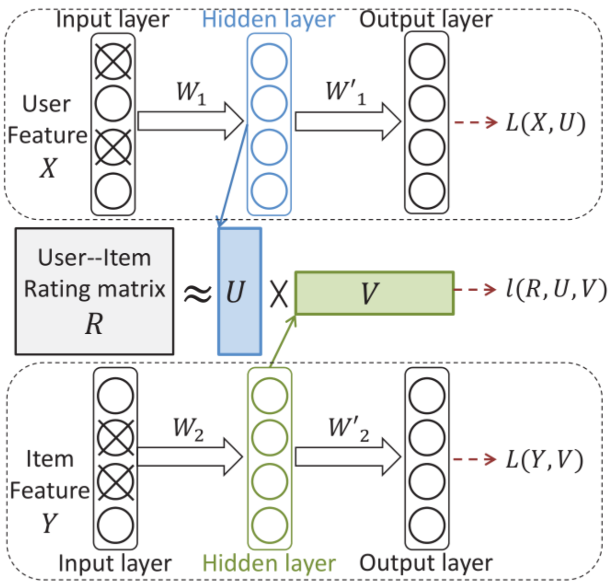
   - **DCF**: Deep Collaborative Filtering via Marginalized DAE (Li et al, CIKM'15)
   - **DUIF**: Deep User-Image Feature (Geng et al, ICCV'15)
   - **ACF**: Attentive Collaborative Filtering (Chen et al, SIGIR'17)
   - **CKB**: Collaborative Knowledge Base Embeddings (Zhang et al, KDD'16)

Deep Collaborative Filtering via Marginalized DAE (Li et al, CIKM'15)

- Denoising Auto-Encoder is used to learn features (hidden layers) of user and item from side information.
- The predictive model is MF.

User age, gender, city, occupation, locations …

Item genres, title, texts



User features reconstruction

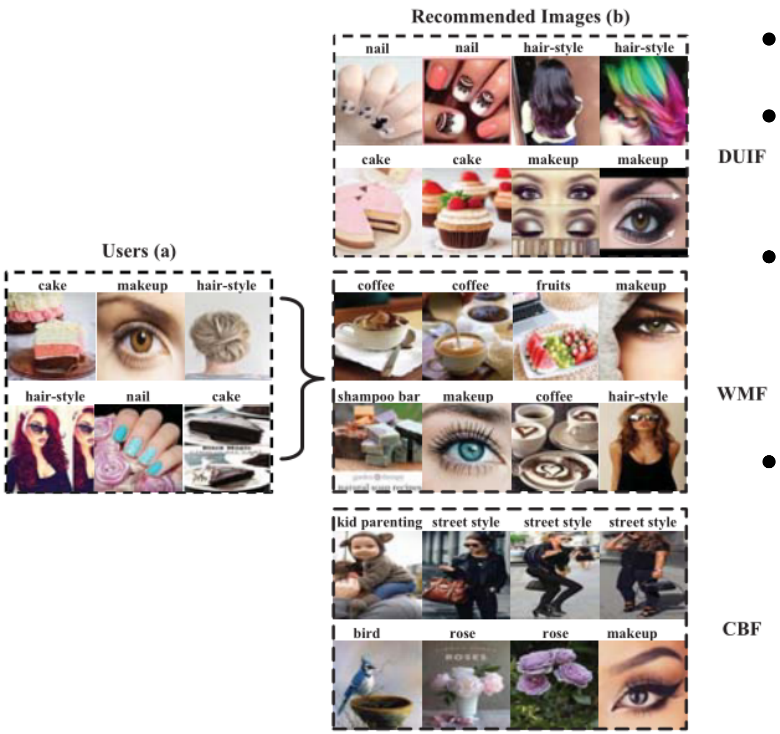
Item features reconstruction

$$\arg\min_{\substack{U,V,W_1,\\W_2,P_1,P_2}} \boxed{\mathcal{L}_U(W_1,P_1,U)} + \boxed{\mathcal{L}_V(W_2,P_2,V)} +$$

$$\underline{\alpha\|A\odot(R-UV^\top)\|_F^2} + \beta(\|U\|_F^2+\|V\|_F^2)$$

Matrix Factorization

DUIF: Deep User and Image Feature Learning (Geng et al, ICCV'15)



- Task: collaborative image recommendation
- Deep CNN (AlexNet) is used to extract features for images
- The deep image features (dim=4096) are projected to user latent space (dim=300) by using linear projection.
- The predictive model is MF:

$$\hat{y}_{ui} =< \mathbf{p}_u, \mathbf{W}^T \mathrm{CNN}(\mathbf{f}_i) >,$$

Linear Projection Image raw features

- The overall model (MF+W+CNN) is trained end-to-end.

ACF: Attentive Collaborative Filtering (Chen et al, SIGIR’17)

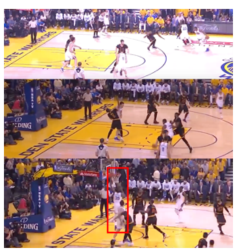
- **Input:**

  user -> ID & historical interacted items.

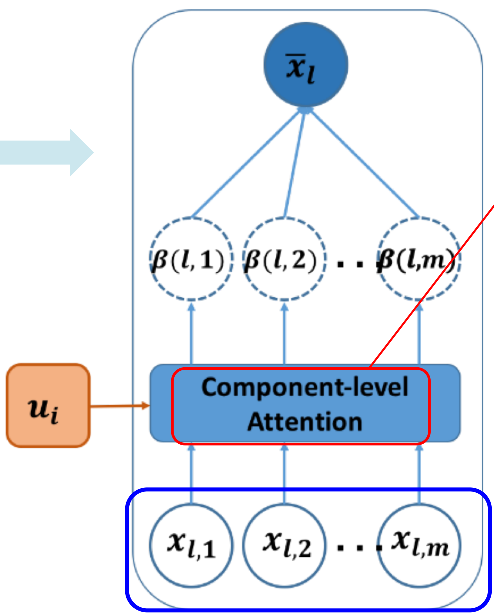
  Item -> ID & visual features.

- **Item Representation**:

  Component-level attention -> components contribute differently to an item’s representation



A user’s preference on different components of the item are not equal!



$$\overline{\mathbf{x}}_l = \sum_{m=1}^{|\{\mathbf{x}_{l*}\}|} \beta(i,l,m) \cdot \mathbf{x}_{lm}$$

Attention Net determines each region’s weight:
- Input: user embedding and region features

Features of each region

ACF: Attentive Collaborative Filtering (Chen et al, SIGIR'17)

- **Input:**

  user -> ID & historical interacted items.

  item -> ID & visual features.

- **User Presentation:**
  - Item-level attention -> historical items contribute differently to a

Attentive SVD++ model:

$$\hat{R}_{ij} = \left(\mathbf{u}_i + \sum_{l \in \mathcal{R}(i)} \alpha(i,l)\mathbf{p}_l\right)^T \mathbf{v}_j$$

A user's preference on different items of user history are not equal!

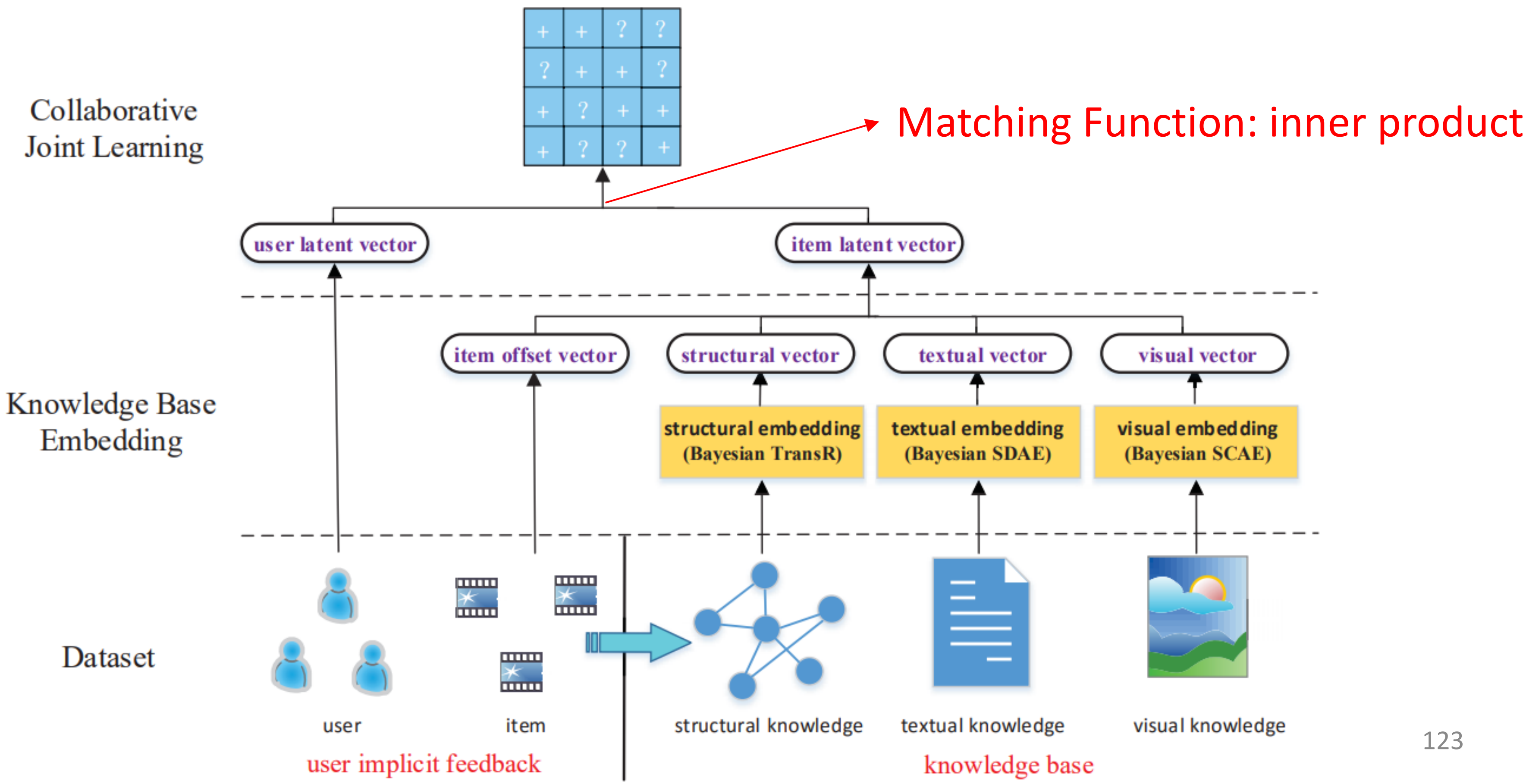
Attention weight determines each item's weight:
- Input: user embedding and item embedding

CKE: Collaborative Knowledge Base Embedding (Zhang et al, KDD'16)

- **Input:**

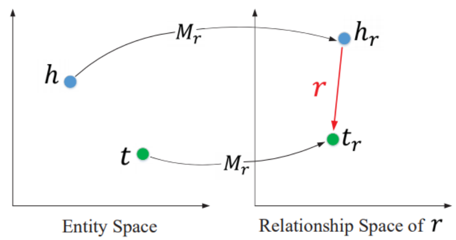
  user -> ID

  item -> ID + Information in KB (structural, textual, visual)



CKE: Collaborative Knowledge Base Embedding (Zhang et al, KDD'16)

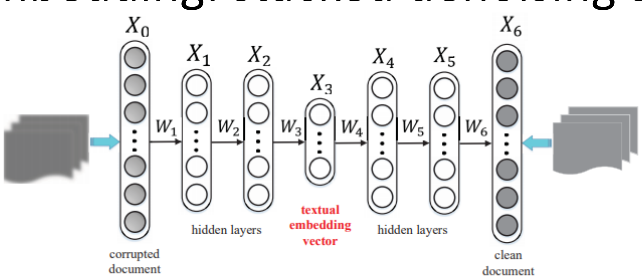
- **Representations:**
  - Structural embedding: TransR, TransE, …



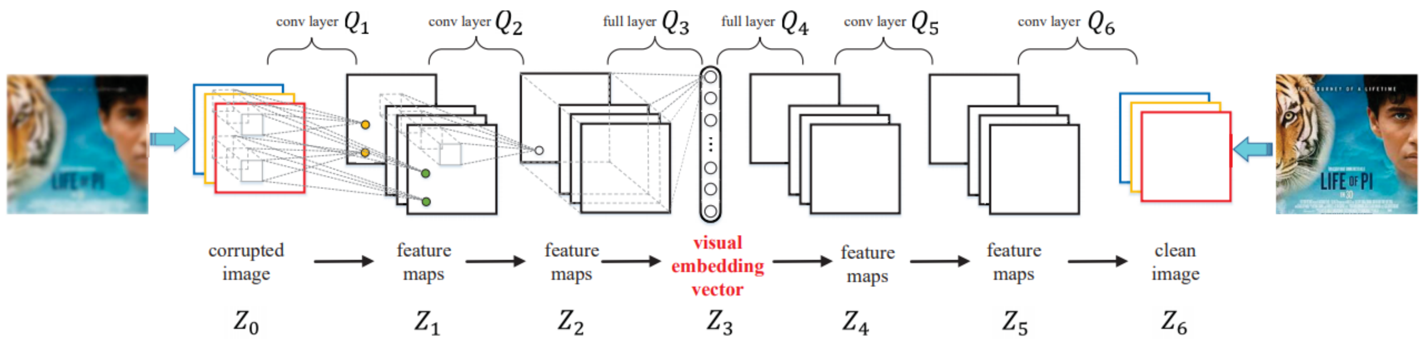
$$\mathbf{v}_h^r = \mathbf{v}_h \mathbf{M}_r, \qquad \mathbf{v}_t^r = \mathbf{v}_t \mathbf{M}_r.$$

$$f_r(v_h, v_t) = ||\mathbf{v}_h^r + \mathbf{r} - \mathbf{v}_t^r||_2^2.$$

  - Textual embedding: stacked denoising auto-encoders (S-DAE)

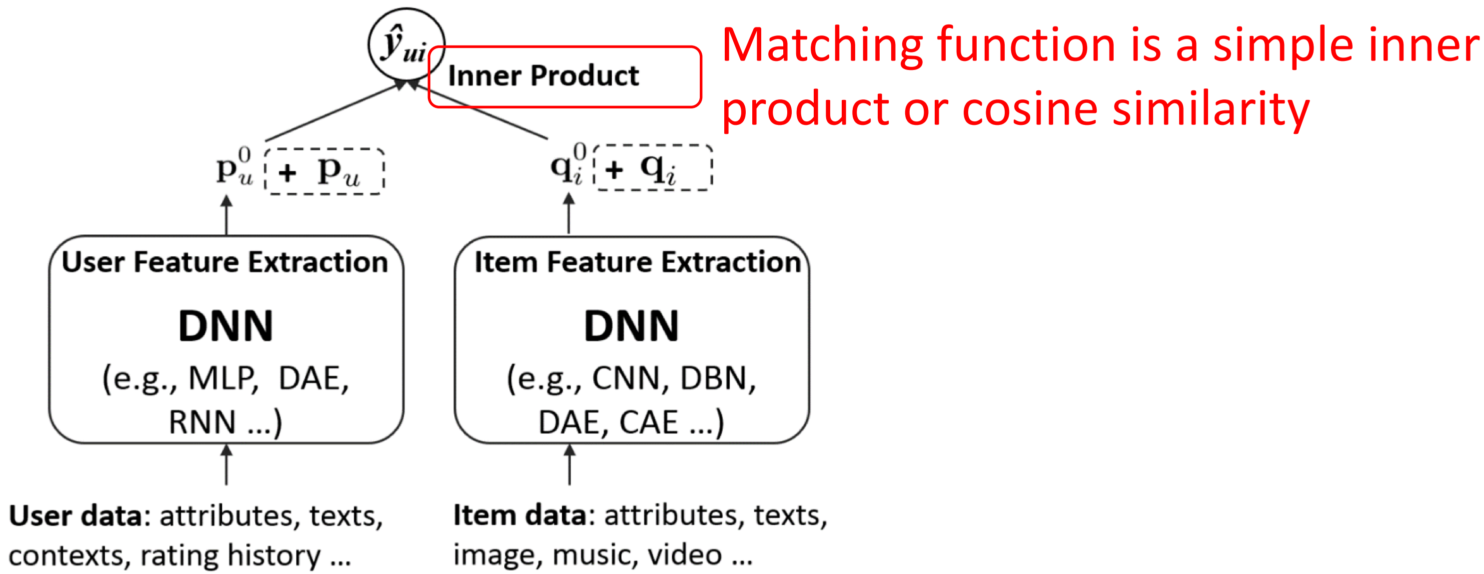


  - Visual embedding: stacked convolutional auto-encoders (SCAE)



Short Summary

- A General framework to summarize the above works:



- Depending on the available data to describe a user/item, we can choose appropriate DNN to learn representation.

  E.g., Textual Attributes -> AutoRec, Image -> CNN, Video -> RNN etc.

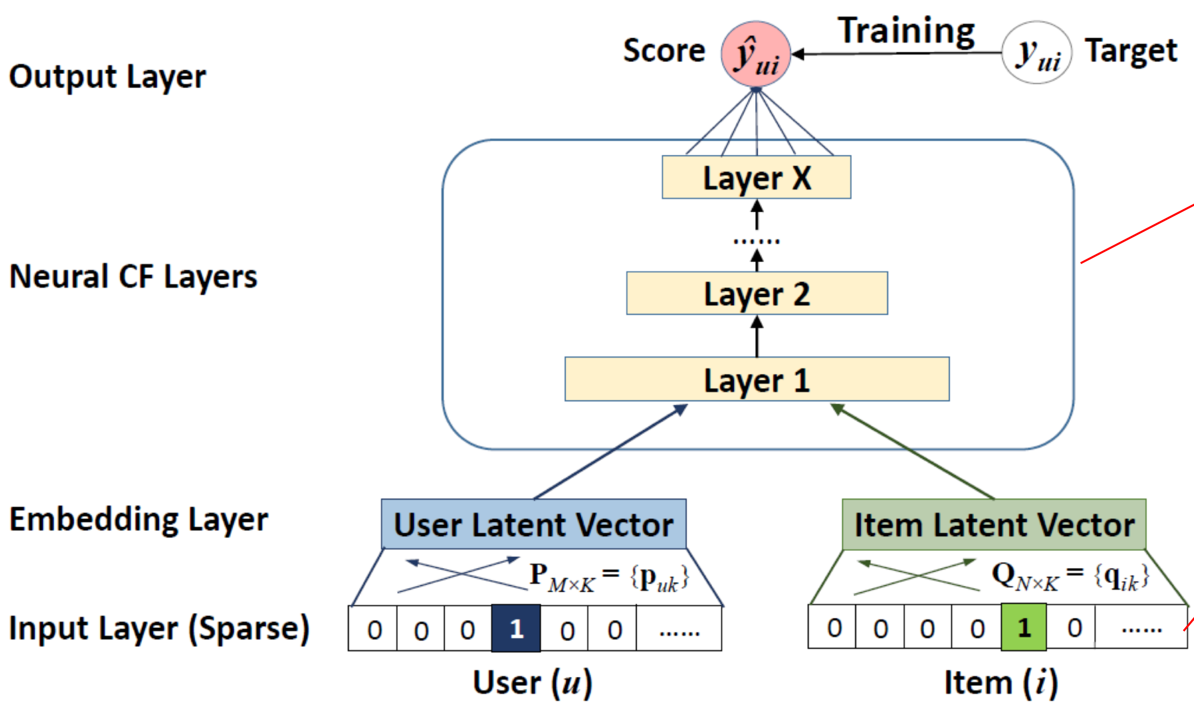
Next: Methods of Matching Function Learning

1. CF models:
   - Based on Neural Collaborative Filtering (NCF) framework:
     NeuMF: Neural Matrix Factorization (He et al, WWW'17)
     ConvNCF: Outer Product-based NCF (He et al, IJCAI'18)
   - Based on Translation framework:
     TransRec: Translation-based Recommendation (He et al, Recsys'17)
     LRML: Latent Relational Metric Learning (Tay et al, WWW'18)
2. Feature-based models:
   - Based on Multi-Layer Perceptron:
     Wide&Deep (Cheng et al, DLRS'16),
     Deep Crossing (Shan et al, KDD'16)
   - Based on Factorization Machines (FM):
     Neural FM (He and Chua, SIGIR'17),
     Attentional FM (Xiao et al, IJCAI'17),
     DeepFM (Guo et al, IJCAI'17)

Neural Collaborative Filtering Framework (He et al, WWW'17)

- **NCF is a general framework that replaces the inner product with a neural network to learn the matching function.** $\hat{y}_{ui} = f(\mathbf{p}_u, \mathbf{q}_i)$

Matching function based on NN



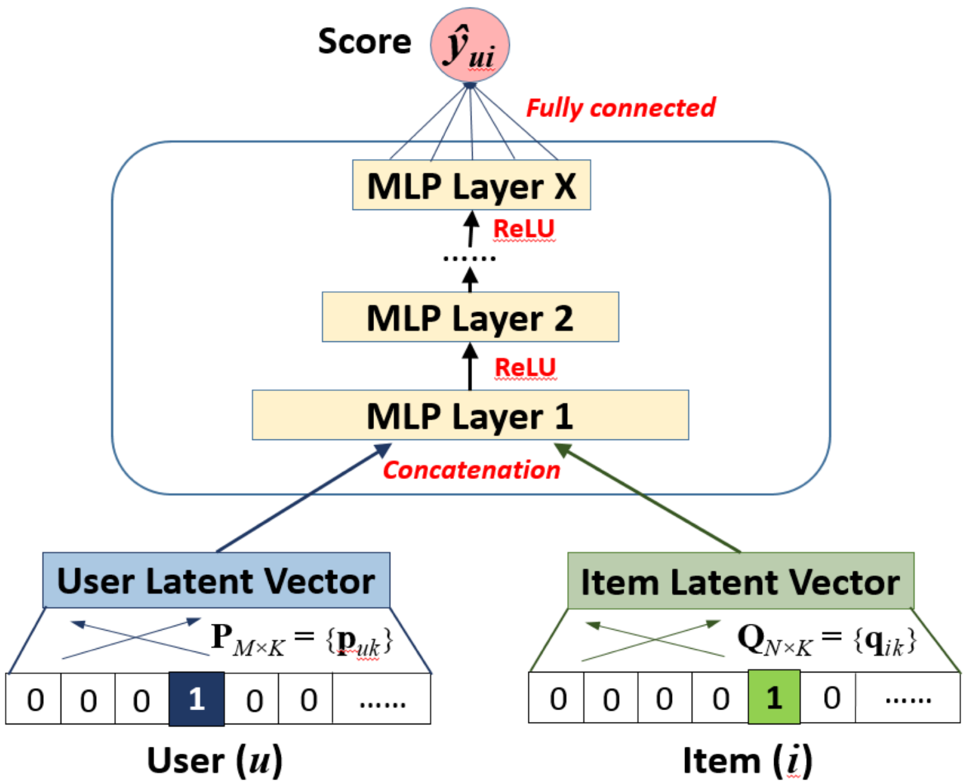
Matching function: design whatever layers as you like.

Input: design whatever features as you want.

Figure 2: Neural collaborative filtering framework

Multi-Layer Perceptron for CF

- The most intuitive idea is to use Multi-Layer Perceptron as the matching function.



Unfortunately, MLP doesn't perform well and underperforms MF.

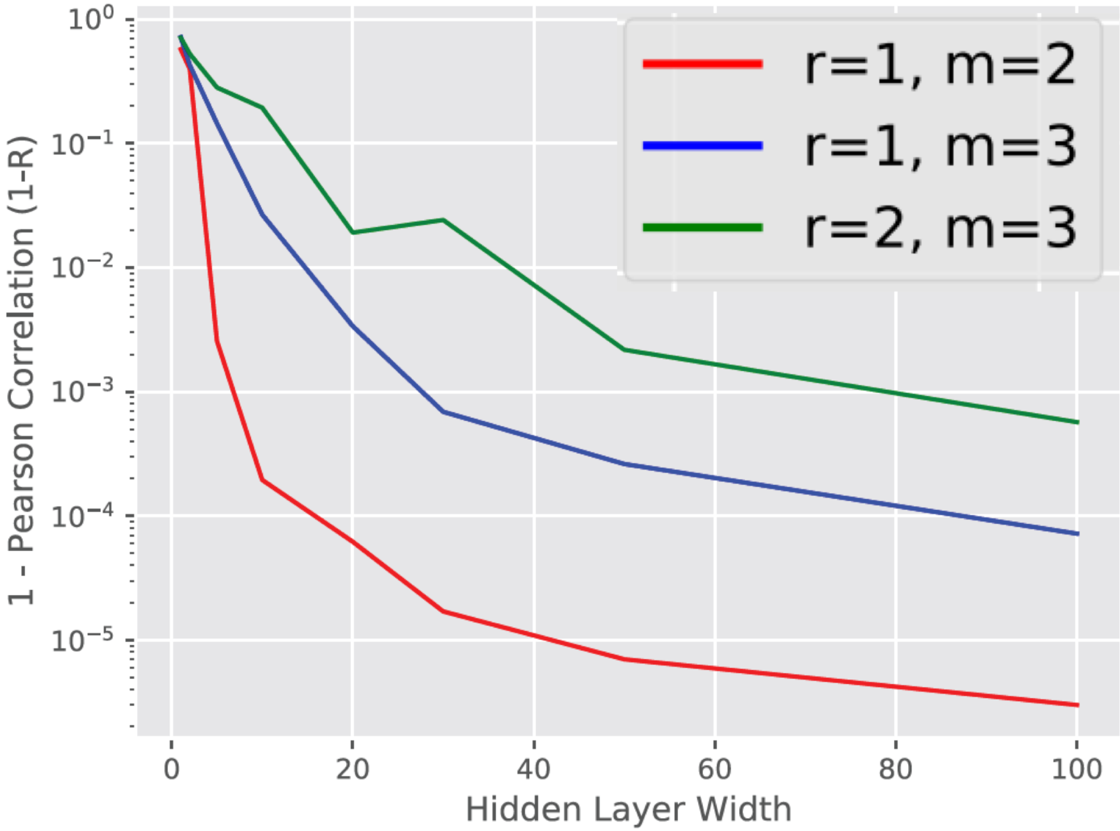
Why?

(He et al, WWW'17)

DNN is Weak in Capturing Multiplicative Relation

- Evidence from Google researchers (Beutel et al, WSDM'18)
  - ➢ Setting: generate low-rank data, and use one-layer MLP to fit it

r: rank size; m: data dimension (2 -> matrix; 3 -> 3D tensor).



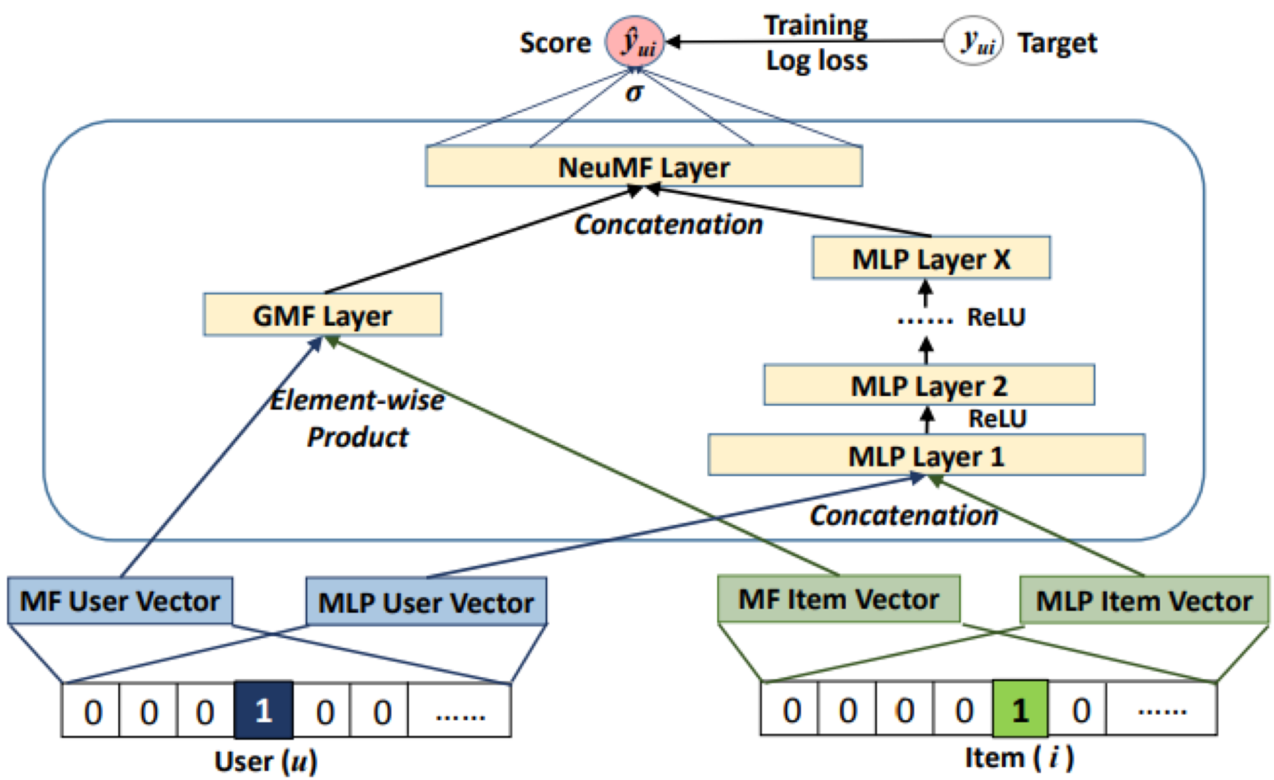
MLP can learn low-rank relation, but is inefficient in doing so!

- Need to use 100 neurons to fit a rank-1 matrix.

Insight: need to augment DNN with multiplicative relation modeling!

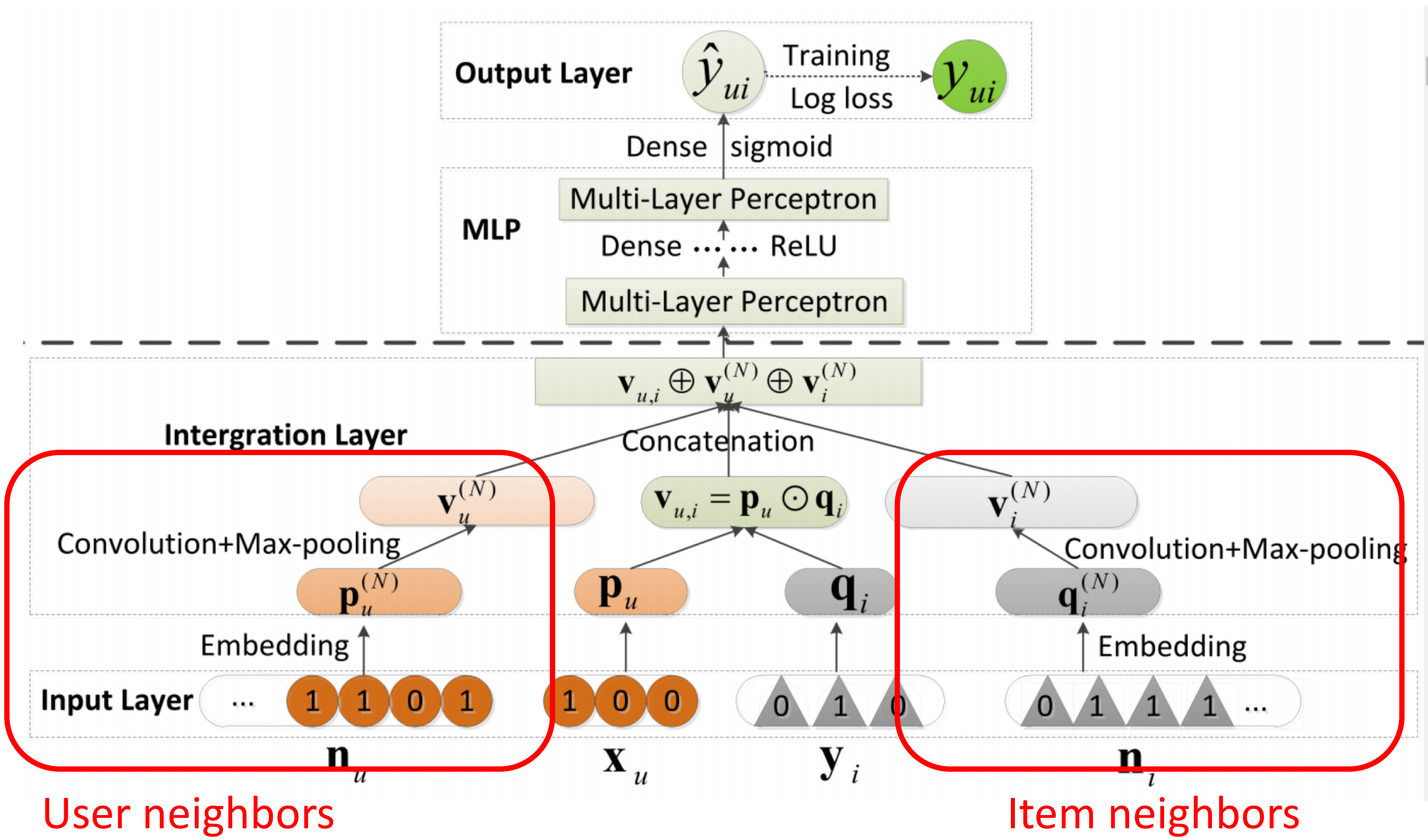
NeuMF: Neural Matrix Factorization (He et al, WWW'17)

- NeuMF unifies the strengths of MF and MLP in learning the matching function:
  - MF uses inner product to capture the multiplicative relation
  - MLP is more flexible in using DNN to learn the matching function.



NNCF: Neighbor-based NCF (Bai et al, CIKM'17)

- Feeding user and item neighbors into the NCF framework
  - Direct neighbors or indirect community neighbors are considered.



Experiment Results(Bai et al, CIKM'17)

Datasets	#Interaction	# Users	#Items	Sparsity
Delicious	437,593	1,867	69,223	99.66%
MovieLens	1,000,209	3,706	6,040	95.53%

Performance Comparison on Item Recommendation (%)

Datasets	Delicious		MovieLens	
Models	HR@5	NDCG@5	HR@5	NDCG@5
ItemPop	5.41	3.22	31.49	20.18
ItemKNN	59.69	55.90	45.01	30.14
MF-BPR	73.77	74.11	51.03	36.21
NeuMF	85.53	80.68	56.55	38.30
NNCF	87.31	84.58	62.00	42.21

CF method is better than non-personalized method

Model-based CF is better than memory-based CF

Deep NCF models are better than shallow MF models by a large margin.

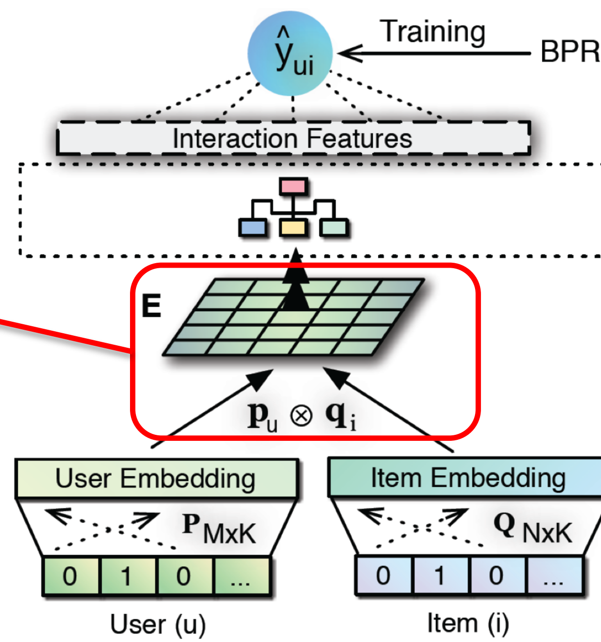
ONCF: Outer-Product based NCF (He et al, IJCAI'18)

- The above NCF models merge user embedding and item embedding with element-wise product or concatenation:
  - Implicitly assume that embedding dimensions are independent.
- How to model the relations between embedding dimensions?
- ONCF applies outer-product on user embedding and item embedding:
  - Explicitly models pairwise correlations between embedding dimensions:

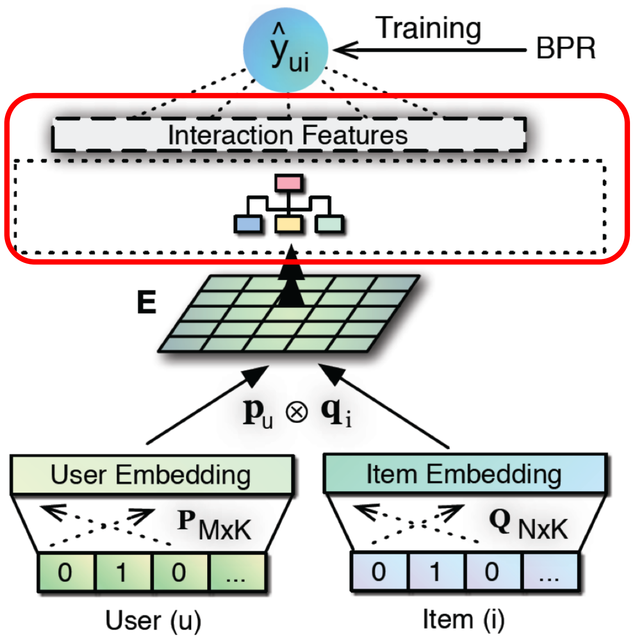
Outer-product gets a 2D "interaction map":

$$e_{k_1,k_2} = p_{u,k_1} q_{i,k_2}$$

- Diagonal elements are inner product



ONCF: Outer-Product based NCF (He et al, IJCAI'18)



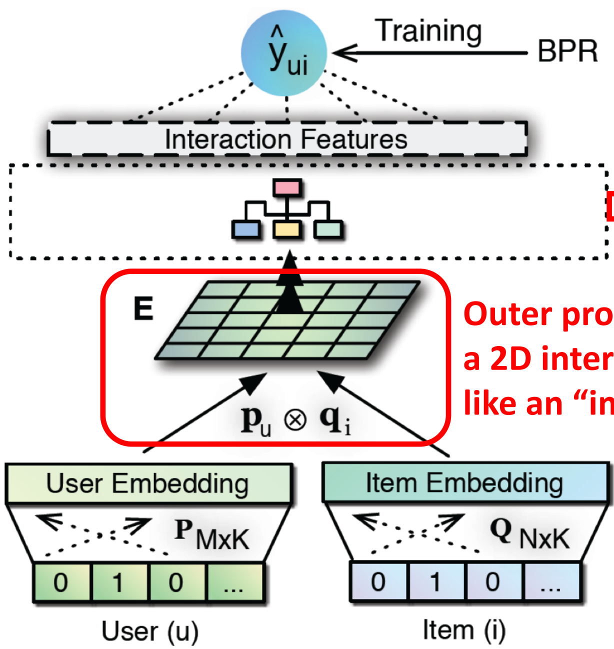
Above the interaction map are hidden layers, which aim to extract useful signal from the 2D interaction map.

A straightforward solution is to use MLP, however it results in too many parameters:

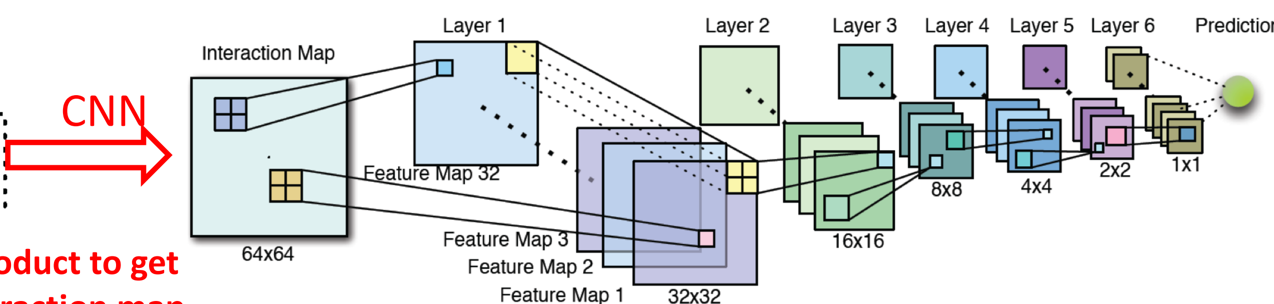
- Interaction map **E** has K×K neurons (K is embeddings size usually hundreds)
- Require **large memories** to store the model
- Require **large training data** to learn the model well

Convolutional NCF (ConvNCF) (He et al, IJCAI'18)

- ConvNCF uses locally connected CNN as hidden layers in ONCF:
  - CNN has much fewer parameters than MLP
  - Hierarchical tower structure: higher layer integrates more information from larger area.
  - Final prediction summarizes all information from interaction map.



Outer product to get a 2D interaction map like an "image"!



- 2 Fully Connected Layers: > 10M parameters
- **6 Convolutional Layers: 20K parameters, but achieve better performance!**

Experiment Results (He et al, IJCAI'18)

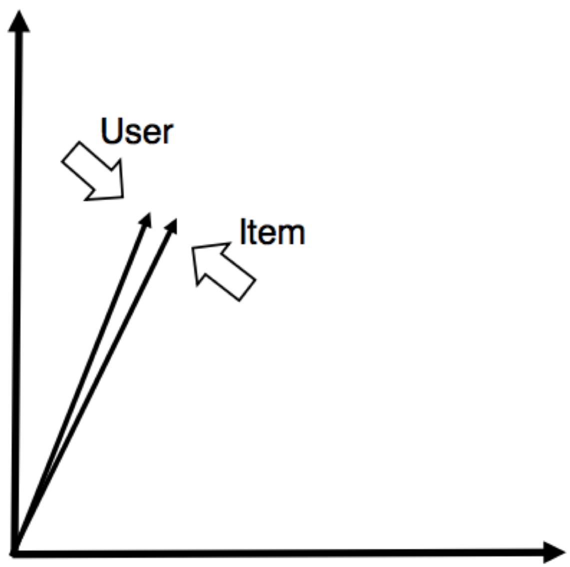
Datasets	#Interactions	#Users	#Items	Sparsity
Yelp	730,791	25,815	25,677	99.89%
Gowalla	1,249,703	54,156	52,400	99.95%

Datasets	Gowalla		Yelp	
Models	HR@5	NDCG@5	HR@5	NDCG@5
ItemPop	20.03	10.99	7.10	3.65
MF-BPR	62.84	48.25	17.52	11.04
MLP	63.59	48.02	17.66	11.03
IRGAN	63.89	49.58	18.61	11.98
NeuMF	67.44	53.19	18.81	11.89
ConvNCF	69.14	54.94	19.06	12.09

ConvNCF are better than NeuMF and MLP with much fewer parameters.

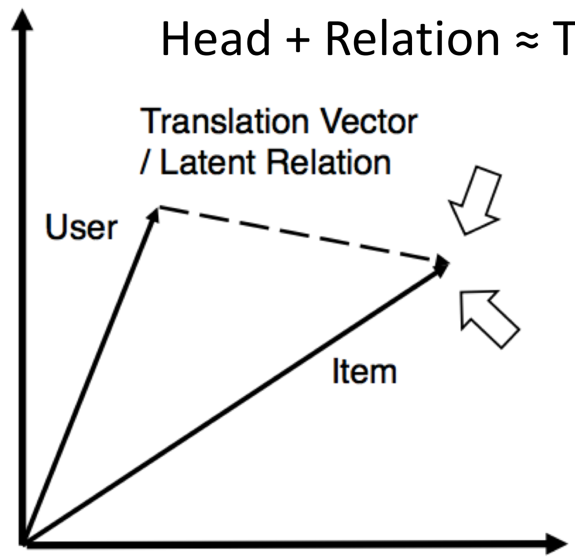
Overview of Translation-based Models (Tay et al, WWW'18)

MF-based model:



- Push ***user vector*** close to ***item vector***
- Measure closeness by inner product (or cosine similarity)

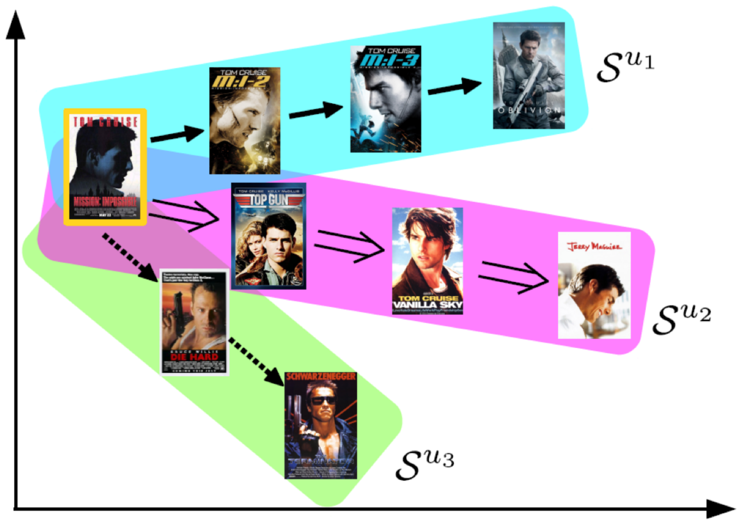
Translation-based:

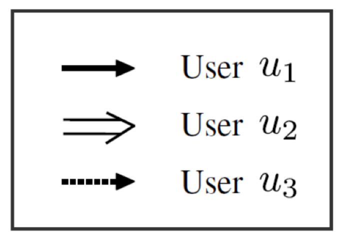


- Push ***user vector*** + ***relation vector*** close to ***item vector***
- Measure closeness by Euclidean distance

TransRec (He et al, Recsys'17)

- Focused on next-item recommendation
  - Third-order relationship between <user, current item, next item>
  - Define relation vector as the current item:





Translation operation:

$\overrightarrow{prev.\ item} + \overrightarrow{user} \approx \overrightarrow{next\ item}$

Predictive Model:

$$Prob(j \mid u, i) \propto \beta_j - d(\vec{\gamma}_i + \vec{T}_u, \vec{\gamma}_j),$$

Item bias

Translation distance

Figure 1: *TransRec* as a sequential model: Items (movies) are embedded into a 'transition space' where each user is modeled by a *translation* vector. The transition of a user from one item to another is captured by a user-specific translation operation.

Latent Relational Metric Learning (Tay et al, WWW'18)

- Distance-based predictive model:

$$s(p, q) = || p + r - q ||_2^2$$

where r is the latent relation vector, formed by an attentive sum over memory vectors:

$$r = \sum_i a_i m_i$$

Attentive weight, with inner product $s = p \odot q$ as input.
(the relation vector is dependent on user and item)

Memory vector, which can encode user attributes/interest.

LRML model:

$$s(p, q) = || p + r - q ||_2^2$$

Latent relation vector:

$$r = \sum_i a_i m_i$$

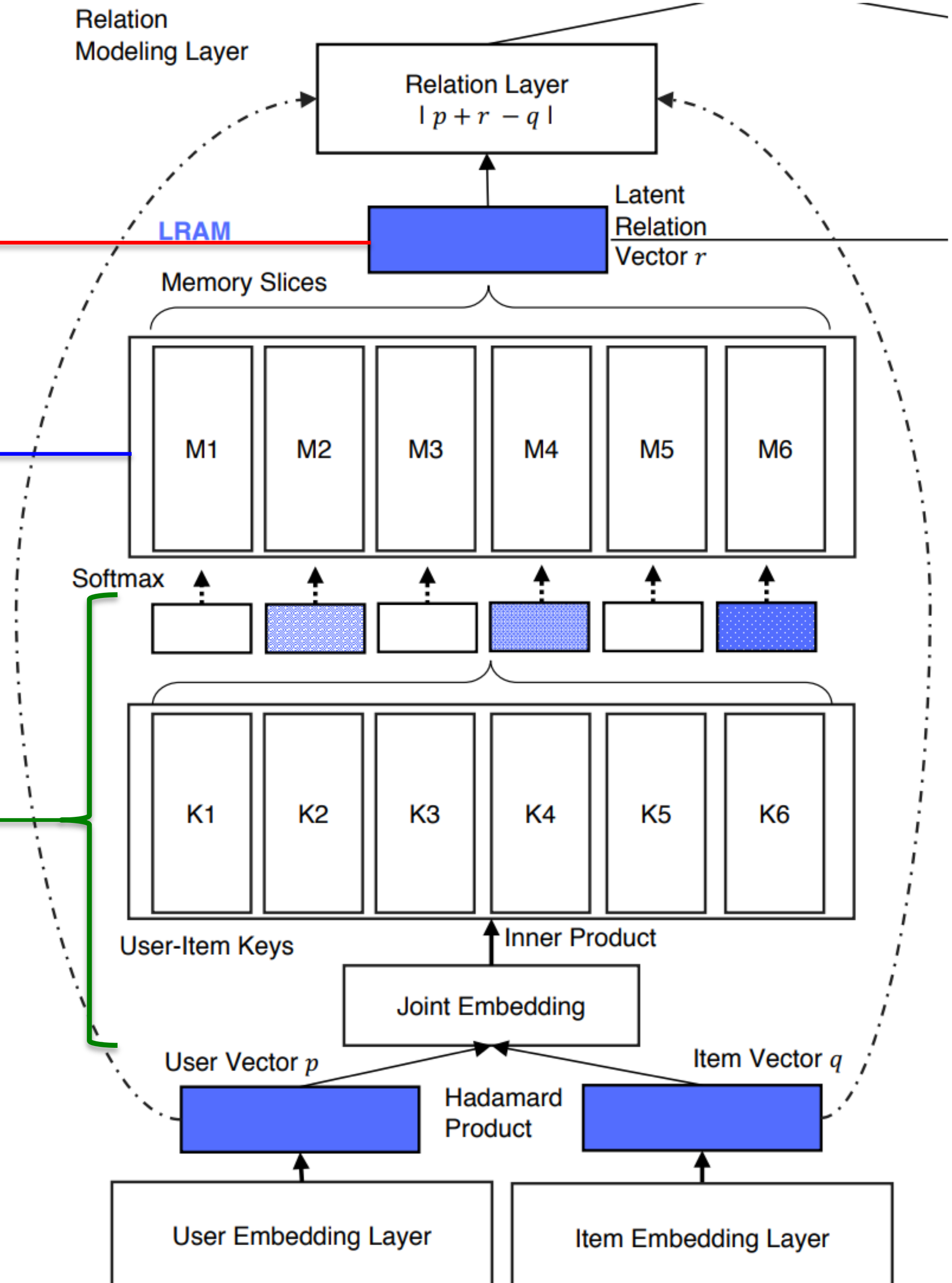
Memory vectors m_i (free parameters to learn)

Generate attentive weights:

$$a_i = (\mathbf{p} \odot \mathbf{q})^T \underline{\mathbf{k}_i})$$

Key for memory i

Normalize using softmax

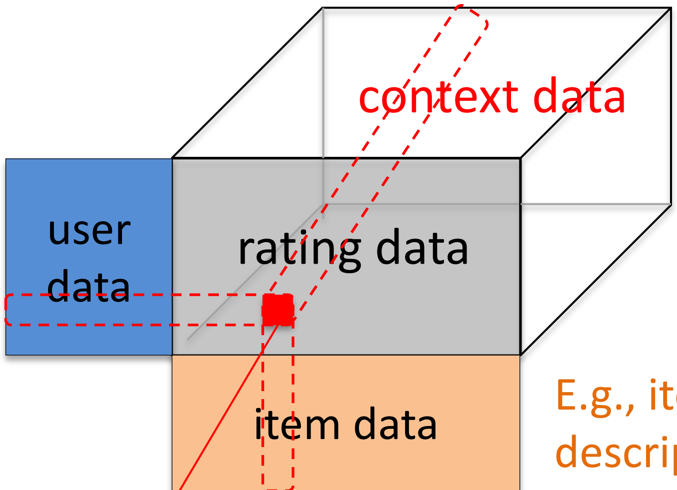


Next: Methods of Matching Function Learning

1. CF models:
   - Based on Neural Collaborative Filtering (NCF) framework:
     NeuMF: Neural Matrix Factorization (He et al, WWW’17)
     ConvNCF: Outer Product-based NCF (He et al, IJCAI’18)
   - Based on Translation framework:
     TransRec: Translation-based Recommendation (He et al, Recsys’17)
     LRML: Latent Relational Metric Learning (Tay et al, WWW’18)
2. Feature-based models:
   - Based on Multi-Layer Perceptron:
     Wide&Deep (Cheng et al, DLRS’16),
     Deep Crossing (Shan et al, KDD’16)
   - Based on Factorization Machines (FM):
     Neural FM (He and Chua, SIGIR’17),
     Attentional FM (Xiao et al, IJCAI’17),
     DeepFM (Guo et al, IJCAI’17)

Recall: Input to Feature-based Models

E.g., user gender, age, occupation personality …



E.g., recent history, location, time, weather, mood …

E.g., item category, description, image …

One-hot encoding

Input Features:

1. Categorical features:
   user/item ID, bag-of-words, historical features…
2. Numerical features:
   textual/visual embeddings, converted features (e.g. TFIDF, GBDT)…

Each row encodes all info for a rating

Feature vector **x**

	User				Movie					Other Movies rated					Target y	
	A	B	C	...	TI	NH	SW	ST	...	TI	NH	SW	ST	...		
$\mathbf{x}^{(1)}$	1	0	0	...	1	0	0	0	...	0.3	0.3	0.3	0	...	5	$y^{(1)}$
$\mathbf{x}^{(2)}$	1	0	0	...	0	1	0	0	...	0.3	0.3	0.3	0	...	3	$y^{(2)}$
$\mathbf{x}^{(3)}$	1	0	0	...	0	0	1	0	...	0.3	0.3	0.3	0	...	1	$y^{(3)}$
$\mathbf{x}^{(4)}$	0	1	0	...	0	0	1	0	...	0	0	0.5	0.5	...	4	$y^{(4)}$
$\mathbf{x}^{(5)}$	0	1	0	...	0	0	0	1	...	0	0	0.5	0.5	...	5	$y^{(5)}$
$\mathbf{x}^{(6)}$	0	0	1	...	1	0	0	0	...	0.5	0	0.5	0	...	1	$y^{(6)}$
$\mathbf{x}^{(7)}$	0	0	1	...	0	0	1	0	...	0.5	0	0.5	0	...	5	$y^{(7)}$

Key to Feature-based Models

- Feature vector is high-dimensional but sparse
  - Consider the CF case: feature vector = user ID + item ID
  - Need to discover prediction patterns in nonzero features
- The interactions between features are important

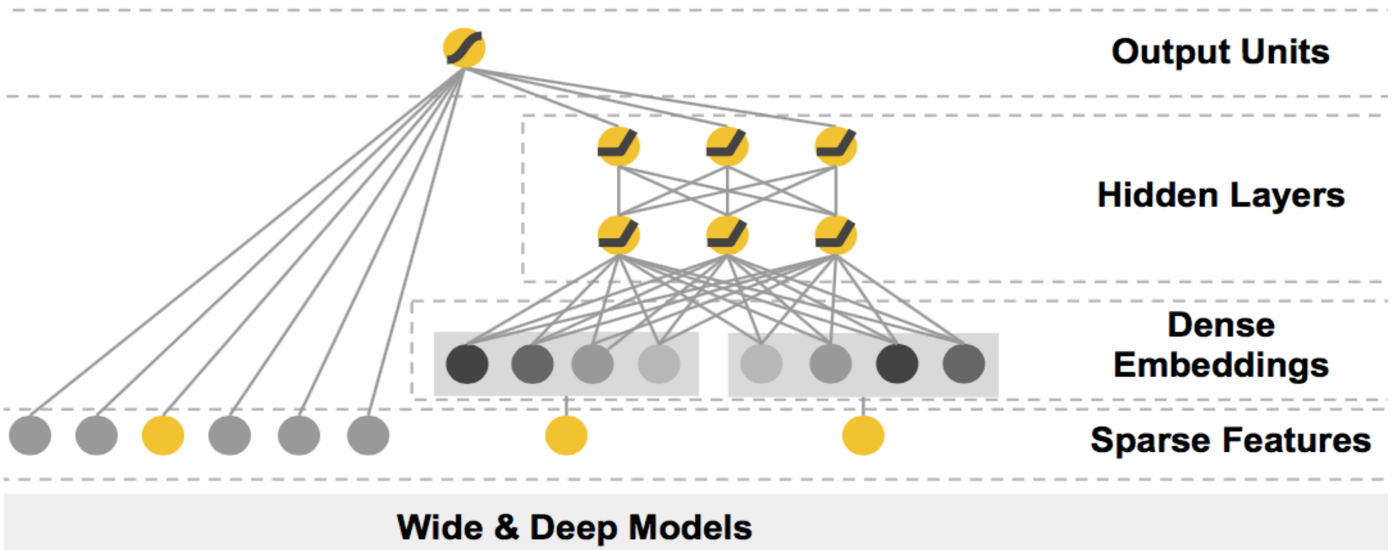
  E.g., users like to use food delivery apps at meal-time

  => Order-2 interactions between app category and time

  E.g., male teenagers like shooting games

  => Order-3 interactions between gender, age, and app category.
- Crucial for feature-based models to capture feature interactions (aka., cross features)

Wide&Deep (Cheng et al, Recsys'16)



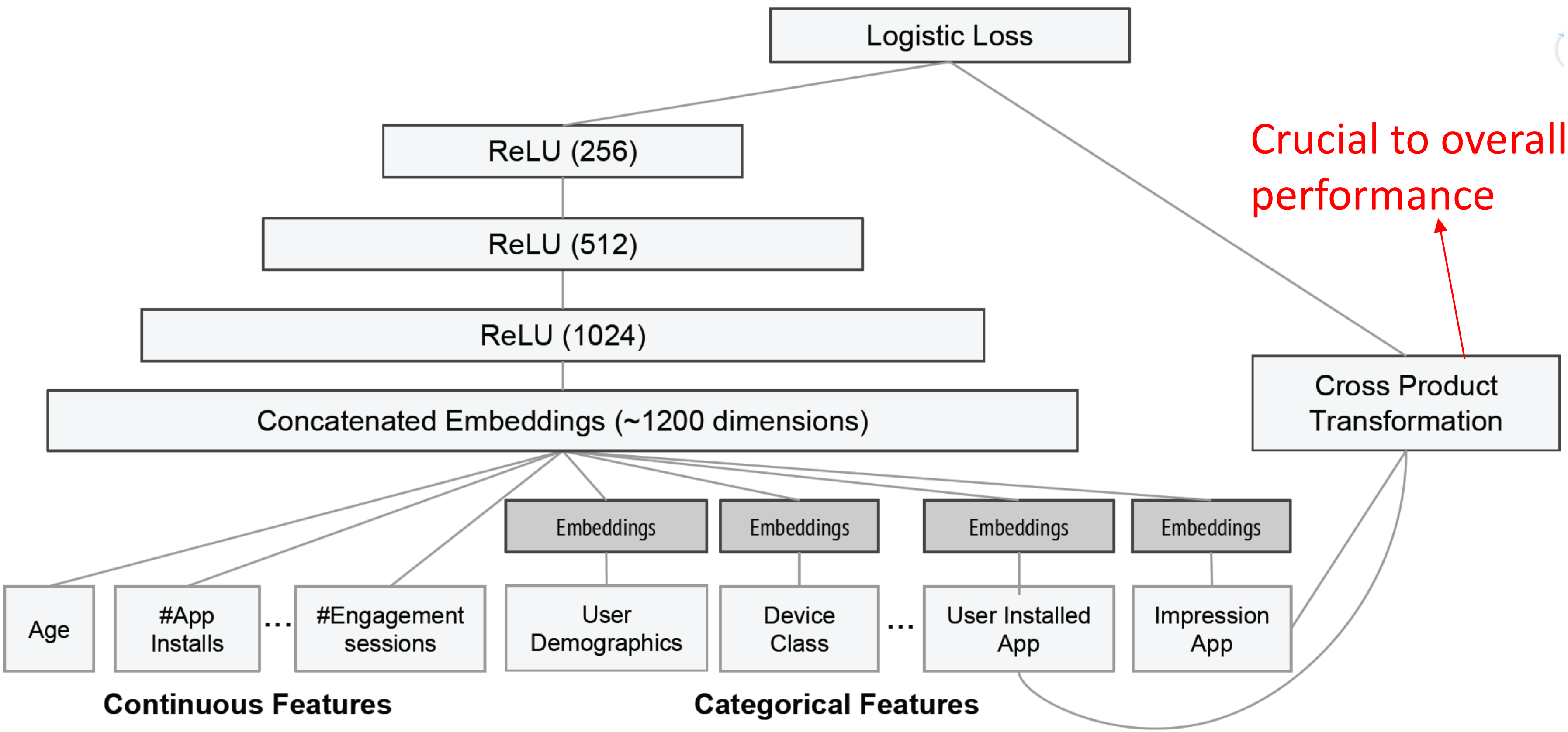
Standard MLP

Embedding Concatenation

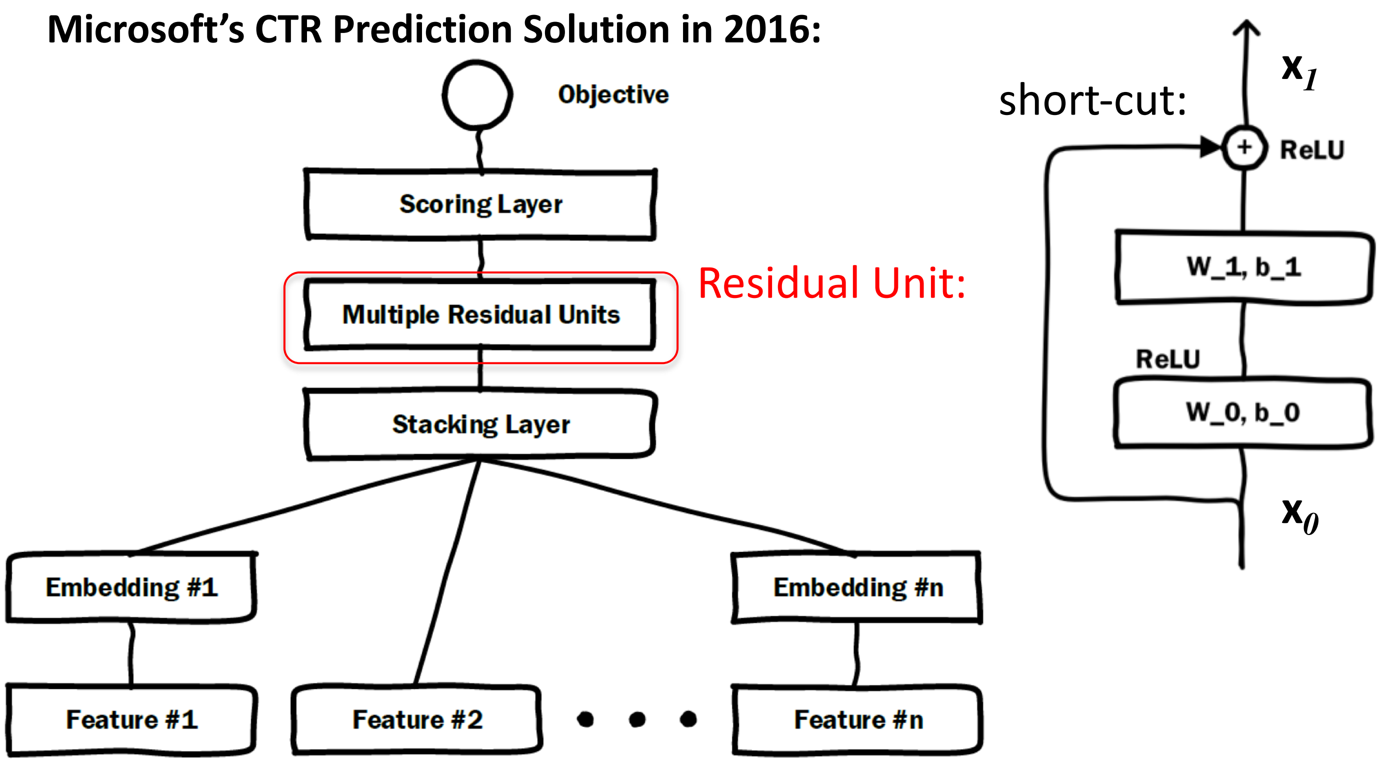
- The wide part is linear regression for memorizing seen feature interactions, which requires careful engineering on cross features.
  E.g., *AND(gender=female, language=en)* is 1 iff both single features are 1

- The deep part is DNN for generalizing to unseen feature interactions.
  Cross feature effects are captured in an implicit way.

Wide&Deep for Google App Recommendation (Cheng et al, Recsys'16)



Deep Crossing (Shan et al, KDD'16)

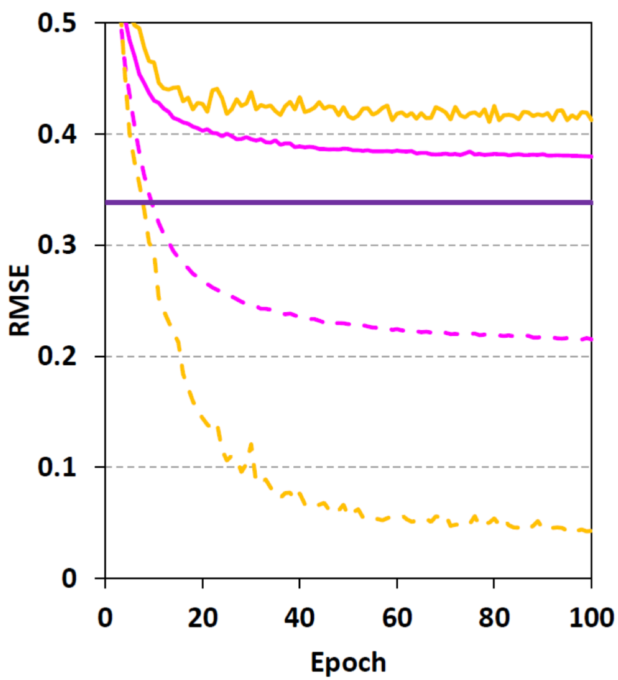


The main difference from Wide&Deep is the use of residual layers, which allow deeper network to be built (~10 layers).

Empirical Evidence (He and Chua, SIGIR'17)

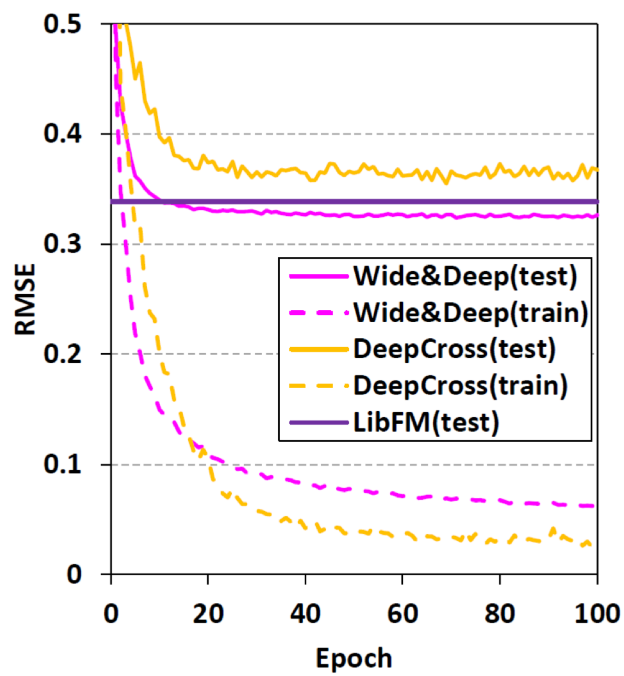
- However, when only raw features are used, both DL models don't perform well in learning feature interactions.

Solid line: testing loss; Dashed line: training loss



(a) Random initialization

With random initialization, deep methods underperform FM.



(b) FM as pre-training

With FM embeddings as pre-training, Wide&Deep slightly outperforms FM.

Some issues of DL methods:

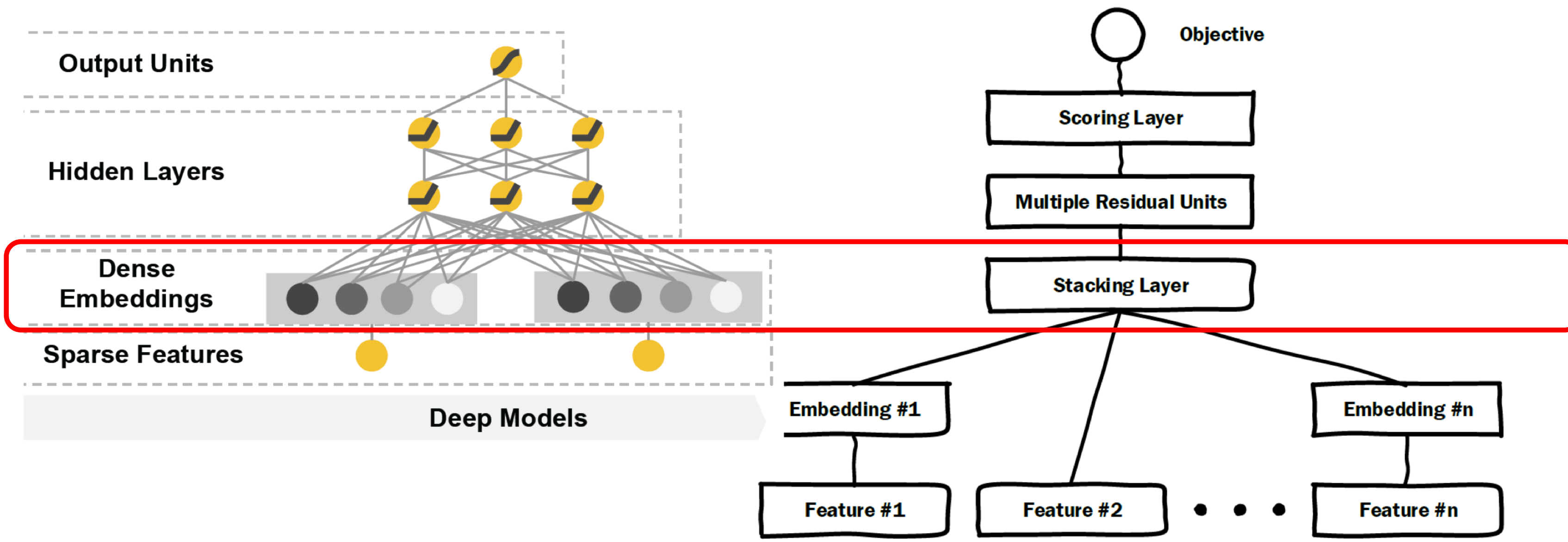
Easy to overfit

Hard to train well

Need good init.

Why MLP is Ineffective?

Besides optimization difficulties, one reason is in model design:



1. Embedding concatenation carries little information about feature interactions in the low level!
2. The structure of Concat+MLP is ineffective in learning the multiplicative relation (Beutel et al, WSDM'18).

Recap: Factorization Machine

- FM explicitly models second-order interactions between feature embeddings with inner product:

Only nonzero features are considered

$$\hat{y}(\mathbf{x}) = w_0 + \underbrace{\sum_{i=1}^{p} w_i x_i}_{\text{First-order: Linear Regression}} + \underbrace{\sum_{i=1}^{p} \sum_{j>i}^{p} < \mathbf{v}_i, \mathbf{v}_j > \boxed{x_i x_j}}_{\text{Second-order: pair-wise interactions between nonzero features}}$$

  – Note: self-interaction is not included: $<\cancel{\mathbf{v}_i, \mathbf{v}_i}>$.

NFM: Neural Factorization Machine (He and Chua, SIGIR’17)

- Neural FM “deepens” FM by placing hidden layers above second-order interaction modeling.

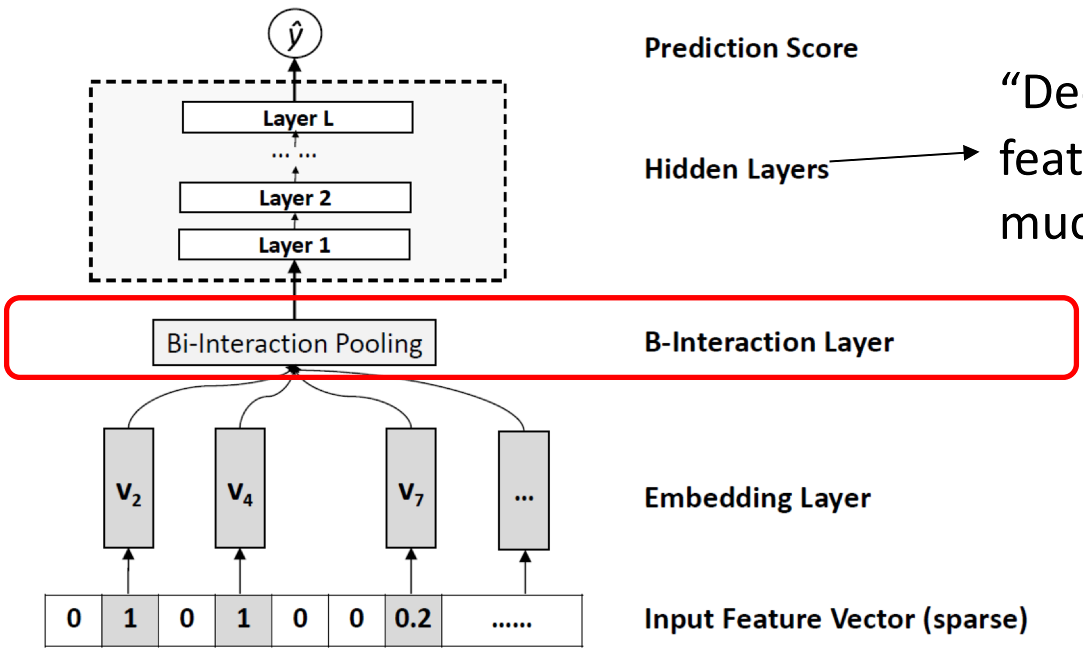


Figure 2: Neural Factorization Machines model (the first-order linear regression part is not shown for clarity).

“Deep layers” learn **higher-order** feature interactions only, being much easier to train.

Bilinear Interaction Pooling:

$$f_{BI}(\mathcal{V}_x) = \sum_{i=1}^{n} \sum_{j=i+1}^{n} x_i \mathbf{v}_i \odot x_j \mathbf{v}_j,$$

Experiment Results (He and Chua, SIGIR'17)

All methods are fed into raw features without any feature engineering

Task #1: Context-aware App Usage Prediction
- Frappe data: instance #: 288,609, feature #: 5,382

Task #2: Personalized Tag Recommendation
- MovieLens data: Inst #: 2,006,859, Feat #: 90,445

Table: Parameter # and testing RMSE at embedding size 128

	Frappe		MovieLens	
Method	**Param#**	**RMSE**	**Param#**	**RMSE**
Logistic Regression	5.38K	0.5835	0.09M	0.5991
FM	1.38M	0.3385	23.24M	0.4735
High-order FM	2.76M	0.3331	46.40M	0.4636
Wide&Deep (3 layers)	4.66M	0.3246	24.69M	0.4512
DeepCross (10 layers)	8.93M	0.3548	25.42M	0.5130
Neural FM (1 layer)	**1.45M**	**0.3095**	**23.31M**	**0.4443**

1. Shallow embedding methods learn interactions, better than simple linear models

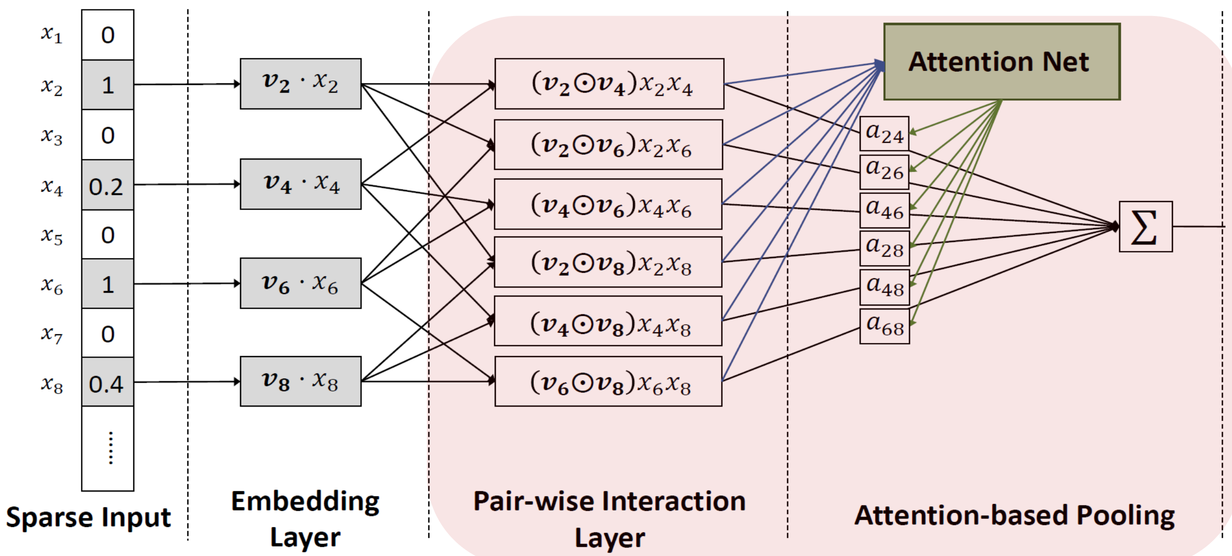
2. Deep embedding methods:
Wide&Deep = Concat+3 layers
DeepCross = Concat+10 layers

3. Neural FM
= BI pooling + 1 layer
Shallower but outperforming existing deeper methods with less parameters.

Codes: github.com/hexiangnan/neural_factorization_machine

AFM: Attentional Factorization Machine (Xiao et al, IJCAI’17)

- Neural FM treats all second-order feature interactions as contributing equally.
- Attentional FM uses an attention network to learn the weight of a feature interaction.



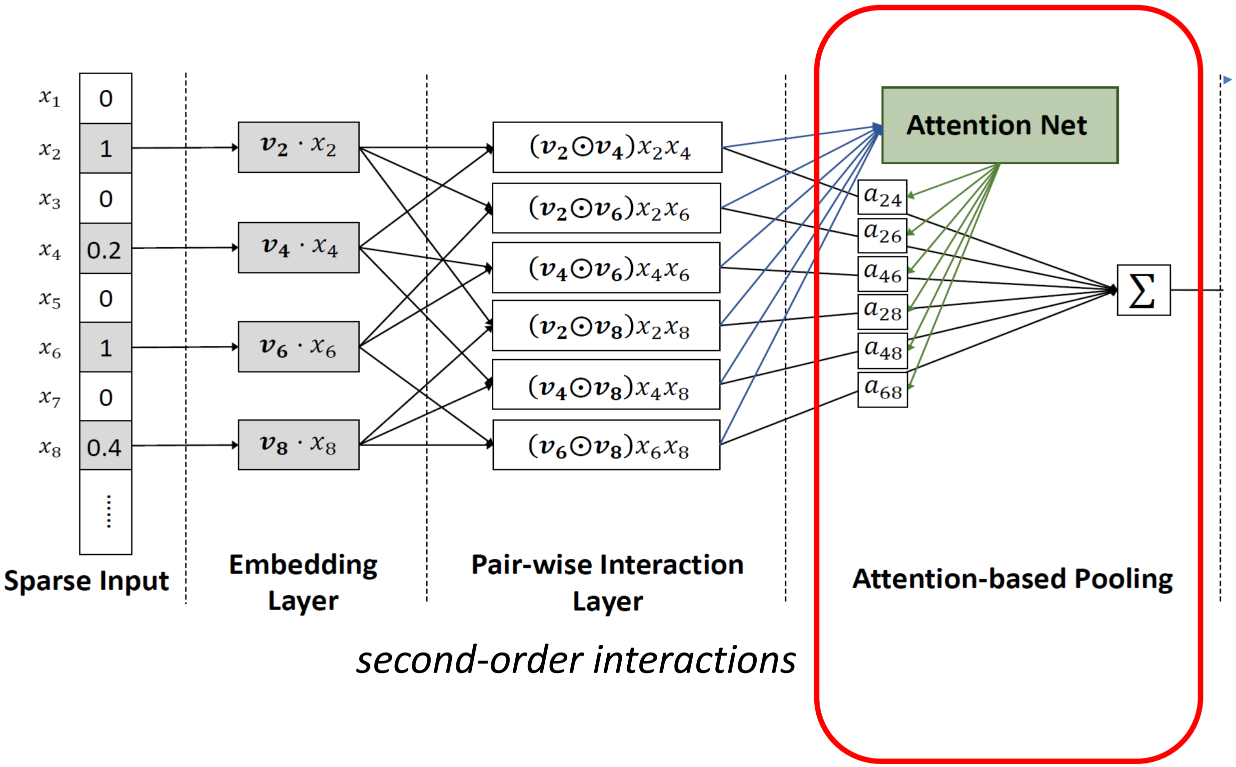
$$f_{ABI}(\mathcal{V}_x) = \sum_{i=1}^{n}\sum_{j=i+1}^{n}(x_i\mathbf{v}_i \odot x_j\mathbf{v}_j)a_{ij}$$

$$a'_{ij} = \mathbf{h}^T ReLU(\mathbf{W}(\mathbf{v}_i \odot \mathbf{v}_j)x_i x_j + \mathbf{b}),$$

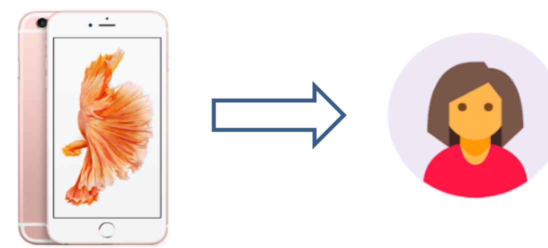
$$a_{ij} = \frac{\exp(a'_{ij})}{\sum_{(i,j)\in\mathcal{R}_x}\exp(a'_{ij})},$$

Explaining Recommendation with AFM

The attention scores can be used to select the most predictive second-order feature interactions as explanations.



Example: explainable recommendation with second-order cross features:



<Female, Age 20>

<Age 20, iPhone>

<Female, Color Pink>

......

Experiment Results

Task #1: Context-aware App Usage Prediction
- Frappe data: instance #: 288,609, feature #: 5,382

Task #2: Personalized Tag Recommendation
- MovieLens data: Inst #: 2,006,859, Feat #: 90,445

Table: Parameter # and testing RMSE at embedding size 128

	Frappe		MovieLens	
Method	**Param#**	**RMSE**	**Param#**	**RMSE**
Logistic Regression	5.38K	0.5835	0.09M	0.5991
FM	1.38M	0.3385	23.24M	0.4735
High-order FM	2.76M	0.3331	46.40M	0.4636
Wide&Deep (3 layers)	4.66M	0.3246	24.69M	0.4512
DeepCross (10 layers)	8.93M	0.3548	25.42M	0.5130
Neural FM (1 layer)	**1.45M**	**0.3095**	**23.31M**	**0.4443**
Attentional FM (0 layer)	**1.45M**	**0.3102**	**23.26M**	**0.4325**

AFM without hidden layers can be better than NFM with 1 hidden layer.

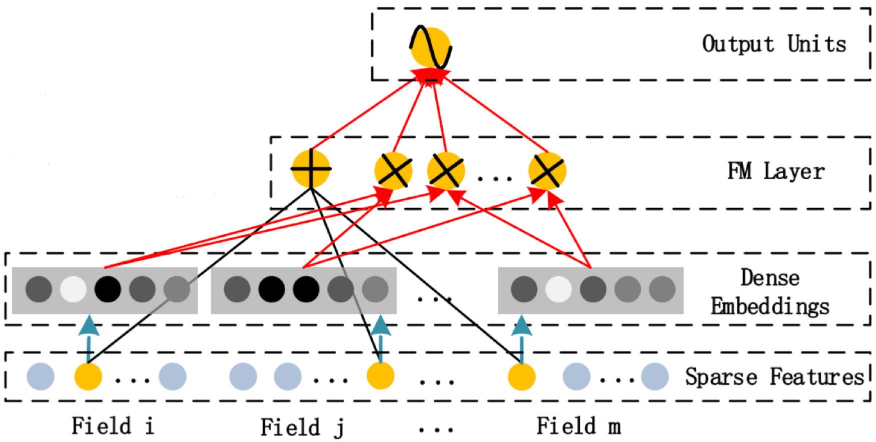
Codes: github.com/hexiangnan/attentional_factorization_machine

Adding hidden layers to AFM further improves.

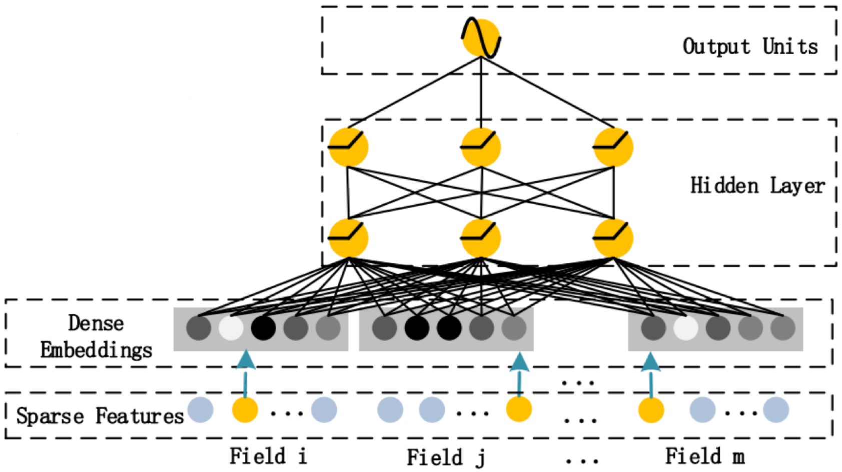
DeepFM (Guo et al., IJCAI'17)

- DeepFM ensembles FM and DNN and to learn both second-order and higher-order feature interactions:

FM structure:



DNN structure:



Prediction Model: $$\hat{y}_{DeepFM} = \hat{y}_{FM} + \hat{y}_{DNN}$$

- Note: FM and DNN share the embedding layer.
- DeepFM learns DNN from the residual of FM
- NeuralFM learns DNN based on the latent space of FM

Short Summary

- ✓ Feature interaction learning is crucial for matching function learning in recommendation.
  - Many models have been explored, e.g., DNN, FM, Attention Net etc.
- ✓ One insight is that doing early cross on raw features (or feature embeddings) is important to performance. E.g.,
  - Wide&Deep do manual cross on raw features
  - FM-based methods do second-order cross on feature embeddings
- Most models learn higher-order interactions with DNN, making higher-order effects hard to explain.
- It remains challenging to do higher-order interaction learning in an explainable way.

References

- Paul Covington, Jay Adams, and Emre Sargin. Deep neural networks for youtube recommendations. In Recsys 2016.
- Xiang Wang, Xiangnan He, Liqiang Nie, and Tat-Seng Chua. Item silk road: Recommending items from information domains to social users. In SIGIR 2017.
- Hong-Jian Xue, Xin-Yu Dai, Jianbing Zhang, Shujian Huang, and Jiajun Chen. Deep matrix factorization models for recommender systems. IJCAI 2017.
- Suvash Sedhain, Aditya Krishna Menon, Scott Sanner, and Lexing Xie. Autorec: Autoencoders meet collaborative filtering. In WWW 2015.
- Yao Wu, Christopher DuBois, Alice X. Zheng, and Martin Ester. Collaborative denoising auto-encoders for top-n recommender systems. In WSDM 2016.
- Sheng Li, Jaya Kawale, and Yun Fu. Deep collaborative filtering via marginalized denoising auto-encoder. In CIKM 2015.
- Xue Geng, Hanwang Zhang, Jingwen Bian, and Tat-Seng Chua. Learning image and user features for recommendation in social networks. In ICCV 2015.
- Jingyuan Chen, Hanwang Zhang, Xiangnan He, Liqiang Nie, Wei Liu, and Tat-Seng Chua. Attentive collaborative filtering: Multimedia recommendation with item-and component-level attention. In SIGIR 2017.
- Fuzheng, Zhang, Nicholas Jing Yuan, Defu Lian, Xing Xie, and Wei-Ying Ma. Collaborative knowledge base embedding for recommender systems. In KDD 2016.
- Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. Neural collaborative filtering. In WWW 2017.
- Ting Bai, Ji-Rong Wen, Jun Zhang, and Wayne Xin Zhao. A Neural Collaborative Filtering Model with Interaction-based Neighborhood. CIKM 2017.

References

- Xiangnan He, Xiaoyu Du, Xiang Wang, Feng Tian, Jinhui Tang, and Tat-Seng Chua. Out Product-based Neural Collaborative Filtering. In IJCAI 2018.
- Tay, Yi, Shuai Zhang, Luu Anh Tuan, and Siu Cheung Hui. "Self-Attentive Neural Collaborative Filtering." *arXiv preprint arXiv:1806.06446* (2018).
- Ruining He, Wang-Cheng Kang, and Julian McAuley. Translation-based Recommendation. In Recsys 2017.
- Yi Tay, Luu Anh Tuan, and Siu Cheung Hui. Latent Relational Metric Learning via Memory-based Attention for Collaborative Ranking. In WWW 2018.
- Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson et al. Wide & deep learning for recommender systems. In DLRS 2016.
- Ying Shan, T. Ryan Hoens, Jian Jiao, Haijing Wang, Dong Yu, and J. C. Mao. Deep crossing: Web-scale modeling without manually crafted combinatorial features. In KDD 2016.
- Xiangnan He, and Tat-Seng Chua. Neural factorization machines for sparse predictive analytics. In SIGIR 2017.
- Jun Xiao, Hao Ye, Xiangnan He, Hanwang Zhang, Fei Wu, and Tat-Seng Chua. Attentional factorization machines: Learning the weight of feature interactions via attention networks. IJCAI 2017.
- Guo, Huifeng, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. Deepfm: A factorization-machine based neural network for CTR prediction. IJCAI 2017

Outline of Tutorial

- Unified View of Matching in Search and Recommendation
- Part 1: Traditional Approaches to Matching
- Part 2: Deep Learning Approaches to Matching
- Summary

Slides: http://comp.nus.edu.sg/~xiangnan/sigir18-deep.pdf

Summary

- Search and Recommendation are two sides of the same coin

  Search -> *Information Pull* with *explicit* info request (query)

  Recommendation -> *Information Push* with *implicit* info request (user profile, contexts)
- Technically, they can be unified under the same matching view
  - Though they are studied by different communities: SIGIR vs. RecSys
- Deep learning-based matching methods
  - Representation learning-focused
  - Matching function learning-focused
- Matching is a generic problem for a wide range of applications

  E.g., online advertising, question answering, image annotation, drug design

Challenges

- Data: building better benchmarks
  - Large-scale text matching data
  - Large-scale user-item matching data with rich attributes/contexts.
- Model: data-driven + knowledge-driven
  - Most current methods are purely data-driven
  - Prior information (e.g., domain knowledge, large-scale knowledge based) is helpful and should be integrated into data-driven learning in a principled way.
- Task: multiple criteria
  - Existing work have primarily focused on similarity
  - Different application scenarios should have different matching goals
  - Other criteria such as novelty, diversity, and explainability should be taken into consideration

Thanks!

Slides: http://comp.nus.edu.sg/~xiangnan/sigir18-deep.pdf