Deep Learning for Matching in Search and Recommendation

Jun Xu
Renmin University of China

Xiangnan He
National University of Singapore

Hang Li
Bytedance AI Lab
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
Overview of Search Engine

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

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

**Content** is in
- Webpages, images, ...

**Key challenge**: query-document semantic gap
### Example of Query-Document Mismatch

<table>
<thead>
<tr>
<th>Query</th>
<th>Document</th>
<th>Term matching</th>
<th>Semantic matching</th>
</tr>
</thead>
<tbody>
<tr>
<td>seattle best hotel</td>
<td>seattle best hotels</td>
<td>partial</td>
<td>yes</td>
</tr>
<tr>
<td>pool schedule</td>
<td>swimming pool schedule</td>
<td>partial</td>
<td>yes</td>
</tr>
<tr>
<td>natural logarithm</td>
<td>logarithm transformation</td>
<td>partial</td>
<td>yes</td>
</tr>
<tr>
<td>transformation</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>china kong</td>
<td>china hong kong</td>
<td>partial</td>
<td>no</td>
</tr>
<tr>
<td>why are windows so</td>
<td>why are macs so expensive</td>
<td>partial</td>
<td>no</td>
</tr>
<tr>
<td>expensive</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Overview of Recommendation Engine

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

**User’s 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)

<table>
<thead>
<tr>
<th></th>
<th>Search</th>
<th>Recommendation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Delivery model</td>
<td>Pull</td>
<td>Push</td>
</tr>
<tr>
<td>Beneficiary (priority)</td>
<td>User</td>
<td>User and Provider</td>
</tr>
<tr>
<td>Unexpected good?</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Collective knowledge</td>
<td>Maybe</td>
<td>Maybe</td>
</tr>
<tr>
<td>Query available</td>
<td>Yes</td>
<td>Maybe</td>
</tr>
<tr>
<td>Context dependent</td>
<td>Maybe</td>
<td>Maybe</td>
</tr>
</tbody>
</table>
Unified View on Matching in Search and Recommendation (Hector et al, CACM’11)

**Common goal:** matching a need (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

Learning to Match
Organization of the 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
  – Overview
  – Deep matching models for search
  – Deep matching models for recommendation

• Summary
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
QUERY-DOCUMENT MATCHING
Overview of Web Search Engine

User Interface → Query Understanding → Retrieving

Ranking

Query-Doc Matching

Index

Crawling → Document Understanding → Indexing

User →

Web
Relation between Matching and Ranking

• In traditional IR: ranking = matching
  \[ f(q, d) = f_{BM25}(q, d) \text{ or } f(q, d) = f_{LMIR}(d|q) \]

• In Web search: ranking and matching become separated
  – Learning to rank: matching as features for ranking
    \[ f(q, d) = f_{BM25}(q, d) + \text{PageRank}(d) + \cdots \]

<table>
<thead>
<tr>
<th>Prediction</th>
<th>Matching</th>
<th>Ranking</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prediction</td>
<td>Matching degree between a query and a document</td>
<td>Ranking list of documents</td>
</tr>
<tr>
<td>Model</td>
<td>( f(q, d) )</td>
<td>( f(q, {d_1, d_2, \cdots }) )</td>
</tr>
<tr>
<td>Goal</td>
<td>Correct matching between query and document</td>
<td>Correct ranking on the top</td>
</tr>
</tbody>
</table>
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
MATCHING IN LATENT SPACE

Query space

Document space

Latent space

Mapping
BM25: Matching in Term Space

\[
f_{BM25}(q, d) = \sum_{q_i \in q} \text{IDF}(q_i) \frac{tf(q_i, d)}{tf(q_i, d) + k_1 \left(1 - b + b \cdot \frac{|d|}{\text{avgedl}}\right)}
\]

\[
= \sum_{w \in V} 1_{w \in q} \times \frac{\text{IDF}(w) \cdot tf(w, d)}{tf(w, d) + k_1 \left(1 - b + b \cdot \frac{|d|}{\text{avgedl}}\right)}
\]

\[
= \langle \mathbf{x}^q, \mathbf{x}^d \rangle
\]
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: $X \subset \mathbb{R}^m$ and $Y \subset \mathbb{R}^n$
- 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
  $$\arg\max_{L_X, L_Y} \sum_{r_i=+1} x_i^T L_X L_Y^T y_i - \sum_{r_i=-1} x_i^T L_X L_Y^T y_i$$
  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 $\ell_1$ and $\ell_2$ regularizations (sparse transformations)
- Objective function
  \[ \underset{L_X, L_Y}{\text{argmax}} \sum_{r_i=+1} x_i^T L_X L_Y^T y_i - \sum_{r_i=-1} x_i^T L_X L_Y^T y_i \]
  subject to $|L_X| \leq \lambda_X$, $|L_Y| \leq \lambda_Y$, $\|L_X\| \leq \vartheta_X$, $\|L_X\| \leq \vartheta_Y$
# PLS v.s. RMLSA

<table>
<thead>
<tr>
<th></th>
<th>PLS</th>
<th>RMLSA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transformation</td>
<td>orthonormal</td>
<td>L1 and L2 regularization</td>
</tr>
<tr>
<td>Assumption</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Optimization</td>
<td>singular value decomposition</td>
<td>coordinate ascent</td>
</tr>
<tr>
<td>Method</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Optimality</td>
<td>global optimum</td>
<td>local optimum</td>
</tr>
<tr>
<td>Efficiency</td>
<td>low</td>
<td>high</td>
</tr>
<tr>
<td>Scalability</td>
<td>low</td>
<td>high</td>
</tr>
</tbody>
</table>
Bridging the Semantic Gap

• Latent space models bridge semantic gap between words through
  – Reducing dimensionality of latent space (from term level matching to semantic matching)
  – Correlating semantically similar terms (matrices are not diagonal)

• Automatically learning mapping functions from data
References

• 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.
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
Modern RecSys Architecture

Phase 1: Candidate Generation
- Collaborative Filtering Models
- Generic Feature-based Recommender Models

Phase 2: Ranking
- User History
- All other Side Info

Data Sources:
- All Items
- User History
- All other Side Info

Models:
- Collaborative Filtering Models
- Generic Feature-based Recommender Models

Data Volumes:
- Millions
- Hundreds
- Tens
Collaborative Filtering

• Collaborative Filtering (CF) is the most well-known technique for recommendation.

  “CF makes predictions (filtering) about a user’s interest by collecting preferences information from many users (collaborating)” — Wikipedia

• Math formulation: matrix completion problem

<table>
<thead>
<tr>
<th>User</th>
<th>Movie</th>
<th>Rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>Titanic</td>
<td>5</td>
</tr>
<tr>
<td>Alice</td>
<td>Notting Hill</td>
<td>3</td>
</tr>
<tr>
<td>Alice</td>
<td>Star Wars</td>
<td>1</td>
</tr>
<tr>
<td>Bob</td>
<td>Star Wars</td>
<td>4</td>
</tr>
<tr>
<td>Bob</td>
<td>Star Trek</td>
<td>5</td>
</tr>
<tr>
<td>Charlie</td>
<td>Titanic</td>
<td>1</td>
</tr>
<tr>
<td>Charlie</td>
<td>Star Wars</td>
<td>5</td>
</tr>
</tbody>
</table>

Input Tabular data

```
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
...    ...        ...
```

Movie

```
|       | TI | NH | SW | ST | ...
|-------|----|----|----|----|----
| Alice | 5  | 3  | 1  | ?  | ...
| Bob   | ?  | ?  | 4  | 5  | ...
| Charlie| 1 | ?  | 5  | ?  | ...
| ...   | ...| ...| ...| ...| ...
```

Rating Matrix
(Interaction Matrix)
Solving Matrix Completion

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

![Rating Matrix](https://en.wikipedia.org/wiki/Singular-value_decomposition)

**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:

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

\[
= \arg\min_{U, \Sigma, V} \sum_{i=1}^{m} \sum_{j=1}^{n} (y_{ij} - (U\Sigma V^T)_{ij})^2
\]

- 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” model in the context of recommendation:

\[ \hat{y}_{ui} = \mathbf{v}_u^T \mathbf{v}_i \]

User latent vector \hspace{1cm} Item latent vector

• Regularized Loss function:

\[
L = \sum_u \sum_i \omega_{ui} (y_{ui} - \hat{y}_{ui})^2 + \lambda \left( \sum_u ||\mathbf{v}_u||^2 + \sum_i ||\mathbf{v}_i||^2 \right)
\]

Prediction error \hspace{1cm} 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 modelled using inner product (measure how much user latent “preferences” match with item “properties”)
  – Besides L2 regularized loss, other loss can also be used, e.g., cross-entropy, margin-based pairwise loss.
Factored Item Similarity Model (Kabbur et al., KDD’14)

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

\[ \hat{y}_{ui} = \left( \sum_{j \in R_u} (q_j)^T v_i \right) \]

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:

\[ \hat{y}_{ui} = (v_u + \sum_{j \in R_u} q_j)^T v_i \]

– 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:
  - E.g., user gender, age, occupation, personality ...
  - E.g., item category, description, image ...
  - E.g., location, time, weather, mood ...
  - rating data
  - context data
  - user data
  - item data
Generic Feature-based Recommendation

- 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)...

- Sparse Predictive Model

- Each row encodes all info for a rating

- E.g., context data:
  - recent history, location, time, weather, mood...

- E.g., item data:
  - item category, description, image...

- E.g., user data:
  - user gender, age, occupation, personality...

- User data
  - rating data
  - item data
  - context data
**FM: Factorization Machines (Rendle, ICDM’10)**

- FM is inspired from previous factorization models.
- It represents each feature an embedding vector, and models the second-order feature interactions:

\[
\hat{y}(x) = w_0 + \sum_{i=1}^{p} w_i x_i + \sum_{i=1}^{p} \sum_{j>i}^{p} <v_i, v_j> x_ix_j
\]

- Note: self-interaction is not included: \(<v_i, v_i>\).
- 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)>.

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

\[ \hat{y}(x) = w_0 + w_u + w_i + \langle v_u, v_i \rangle \]
Factored Item Similarity Model with FM

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

With this input, FM subsumes FISM with additional terms:

\[
\hat{y}(x) = \text{bias} + \sum_{j \in R_u} <v_j, v_i> + \sum_{j \in R_u, j' > j} <v_j, v_{j'}>
\]

Further input user ID into FM will resume SVD++
Explicit Feedback vs. Implicit Feedback

<table>
<thead>
<tr>
<th>User</th>
<th>Movie</th>
<th>TI</th>
<th>NH</th>
<th>SW</th>
<th>ST</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>5</td>
<td>3</td>
<td>1</td>
<td>?</td>
<td>?</td>
<td>...</td>
</tr>
<tr>
<td>B</td>
<td>?</td>
<td>?</td>
<td>4</td>
<td>5</td>
<td>?</td>
<td>...</td>
</tr>
<tr>
<td>C</td>
<td>1</td>
<td>?</td>
<td>5</td>
<td>?</td>
<td>?</td>
<td>...</td>
</tr>
<tr>
<td></td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

**Explicit Feedback** conveys user preference *explicitly*:
- Higher scores carry positive signal
- Lower scores carry negative signal

<table>
<thead>
<tr>
<th>User</th>
<th>Movie</th>
<th>TI</th>
<th>NH</th>
<th>SW</th>
<th>ST</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>?</td>
<td>?</td>
<td>...</td>
</tr>
<tr>
<td>B</td>
<td>?</td>
<td>?</td>
<td>1</td>
<td>1</td>
<td>?</td>
<td>...</td>
</tr>
<tr>
<td>C</td>
<td>1</td>
<td>?</td>
<td>1</td>
<td>?</td>
<td>?</td>
<td>...</td>
</tr>
<tr>
<td></td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

**Implicit Feedback** conveys user preference *implicitly*:
- Observed interactions do not mean positive signal
- Unobserved interactions do not mean negative signal
Rating Prediction is Suboptimal

• Old work on recommendation optimize **L2 loss**:

\[
L = \sum_u \sum_i w_{ui} (y_{ui} - \hat{y}_{ui})^2 + \lambda (\sum_u ||v_u||^2 + \sum_i ||v_i||^2)
\]

– 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

- Modern work on recommendation optimize pairwise ranking loss:
  - Known as the *Bayesian Personalized Ranking* loss (Rendle, UAI’09).
  - It optimizes relative ranking between two items, rather than absolute scores.

\[
L_{BPR} = \arg \max_\Theta \sum_{(u, i, j) \in R_B} \ln \sigma(\hat{y}_{ui} - \hat{y}_{uj}) - \lambda \|\Theta\|^2
\]

Pairwise training examples: \( u \) prefers \( i \) over \( j \)
References

• https: . en.wikipedia.org/wiki/Collaborative_filtering
• Steffen Rendle. Factorization Machines. In ICDM 2010.
• 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
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
  – 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 Approach to Matching

- **Methods of representation learning**
  
  - Query/user representation
  
  - Document/item representation

- **Methods of matching function learning**
  
  - Matching signals
  
  - Aggregation
Methods of Representation Learning

• Step 1: calculate representation $\phi(x)$
• Step 2: conduct matching $F(\phi(x), \phi(y))$
Methods of Matching Function Learning

- Step 1: construct basic 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
METHODS OF REPRESENTATION LEARNING
Representation Learning for Query-Document Matching

- Step 1: calculate query and document representation
- Step 2: conduct query-document matching
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)
  – **CA-RNN**: Representing one sentence with the other sentence as its context (Chen et al., AAAI ’18)
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 to capture the compositional sentence representations

Semantic vector $v_s$
- $d=300$
- $d=500$
- $d=500$
- $d=500$

Letter-trigram embedding matrix $W_s$
- $W_{s,1}$
- $W_{s,2}$
- $W_{s,3}$
- $W_{s,4}$

Letter-trigram encoding matrix (fixed)
- $d=500$
- $d=500$

Bag-of-words vector
- $dim = 100M$

Input word/phrase:
- $s$: "racing car"

$W_t$:
- $d=300$
- $d=500$
- $d=500$
- $d=500$

$t^+$:
- $dim = 50K$
- $dim = 50K$
- $dim = 100M$

$t^-$:
- $dim = 100M$
- $dim = 100M$

Initialization: Neural networks are initialized with random weights.

DSSM for semantic embedding

Huang, He, Gao, Deng, Acero, Heck, "Learning deep structured semantic models for web search using clickthrough data," CIKM, 2013

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 word order information

Convolution

Pooling
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

```
max pooling

max pooling

Convolution

bag of letter-trigram

30k
30k
30k

300
300
300

max
max
max

128

300
300
300

... ...

... ...

30k

the
cat
sat...
...
mat
```
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

Bag of letter-trigrams

![Diagram of LSTM-RNN](attachment:diagram.png)

Embedding vector for the whole sentence
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)
    \[
    s = W_2 \cdot \sigma \left( W_1 \cdot \begin{bmatrix} q \\ d \end{bmatrix} + b_1 \right) + b_2
    \]
Matching Functions (cont’)

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

\[ s = u^T f \left( q^T M^{[1:r]} d + V \begin{bmatrix} q \\ d \end{bmatrix} + b \right) \]
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
  – Representing one sentence using the other as context

• 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

sequence of word embeddings

6 word embeddings from query and document (N=3)
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

• Positions information of words is kept
Matching Function: 2D Convolution

- Discovering the matching patterns with CNN, stored in the kernels
Discovered Matching Patterns

$T_1$: PCCW's chief operating officer, Mike Butcher, and Alex Arena, the chief financial officer, will report directly to Mr So.

$T_2$: Current Chief Operating Officer Mike Butcher and Group Chief Financial Officer Alex Arena will report to So.
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
**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 sat ←→ the dog play
    - the cat sat on ←→ 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

\[
\begin{align*}
F'(\text{the, balls}) &= F(\text{the})F(\text{ball}) \\
\end{align*}
\]
Hypothesis: matching with distributed representations complements matching with local representations

- Local matching: matching function learning
- Distributed matching: representation learning

Binary matrix for capturing exact matching signals

Convolution + Fully connected

Matching signals

Neural Network

score 1
(local matching)

Matching score

sum

score 2
(distributed matching)

Fully connected

element-wise product

N-graph based input encoding

Convolution + Fully connected

Query rep.

Doc rep.
Short Summary

• Two steps
  – 1. Construct basic matching signals
  – 2. Aggregate matching patterns

• Basic matching signals
  – Matching matrix (exact match, dot product, cosine similarity)
  – Decomposable attention weight

• Aggregate matching patterns
  – CNN/Spatial RNN + MLP

• Combining representation learning (inexact match) and matching function learning (exact match)
QUERY-DOCUMENT RELEVANCE MATCHING
Similarity $\neq$ 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 (DRMM)
(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: Matching matrix based on embeddings bridge the semantic gap
- Word order: histogram mapping lost order information

Q: the cat sat on the mat...
D: the dog play balls on ...

Matching matrix based on word embeddings
Matching histogram mapping
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 and soft-TF bridge the semantic gap
- Word order: kernel pooling and sum operations *lost order information*

![Diagram showing matching matrix and kernel pooling](image)
Conv-KNRM (Dai et al., WSDM ‘18)

- Based on KNRM
- N-gram cross-matching to capture the word order information
Short Summary

• Methods based on global distributions of matching strengths
  – 1. calculating term matching strength distributions
  – 2. aggregating the distributions to a matching score
References

- Metz C. AI is Transforming Google Search. The Rest of the Web is Next. WIRED Magazine, 2016.
References

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
Deep Matching Models for Recommendation

• Methods of representation learning

user → Neural Network → User representation

Item → Neural Network → Item representation

• Methods of matching function learning

user → Neural Network → matching score

Item → matching score
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)
Matrix Factorization as a Neural Network (Wang et al, SIGIR’17)

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

\[ f_{MF}(u, i|p_u, q_i) = p_u^T q_i = \sum_{k=1}^{K} p_{uk} q_{ik}, \]

![Diagram](image)
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.

  ![Interaction Matrix](image)

  $\begin{array}{cccccccc}
  Y & u_i & 0 & 0 & 4 & \cdots & 0 & 0 \\
  &  & 3 & 1 & \cdots & 0 & 1 & \cdots \ 0 & \cdots & \cdots & 0 & 0 \\
  \end{array}$

  $M \times N$
Deep Matrix Factorization (Xue et al, IJCAI’17)

• Representation Function:
  – Multi-Layer Perceptron

![Diagram of Deep Matrix Factorization]

Matching Function: cosine similarity

\[ \text{cosine}(p_i, q_j) = \frac{p_i^T q_j}{\|p_i\| \|q_j\|} \]
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

\[
\begin{align*}
\text{Input reconstruction:} & \\
\min_{\theta} \sum_{i=1}^{n} \| r^{(i)} - h(r^{(i)}; \theta) \|_2^2 + \frac{\lambda}{2} \cdot (\| W \|_F^2 + \| V \|_F^2),
\end{align*}
\]

**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

\[ z_u = h \left( W^\top \tilde{y}_u + V_u + b \right), \]

\[ \hat{y}_{ui} = f \left( W_i'\top z_u + b'_i \right), \]
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 MLP (representation learning) + MF (matching function).
  
  - Nonlinear
  - 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 features reconstruction

\[
\arg\min_{U,V,W_1,P_1} \mathcal{L}_U(W_1, P_1, U) + \mathcal{L}_V(W_2, P_2, V) + \\
\alpha \| A \odot (R - UV^T) \|_F^2 + \beta (\| U \|_F^2 + \| V \|_F^2)
\]

Item features reconstruction

Matrix Factorization

User age, gender, city, occupation, locations ...

Item genres, title, texts
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:
  \[ \hat{y}_{ui} = \langle p_u, W^T \text{CNN}(f_i) \rangle, \]

- 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!
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 user’s representation

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

\[
\hat{R}_{ij} = \left( u_i + \sum_{l \in R(i)} \alpha(i, l) p_l \right)^T v_j
\]

Attention weights learned by a neural net ↔ Attentive SVD++ model.
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, ...
    
    \[ \mathbf{v}_h^r = \mathbf{v}_h \mathbf{M}_r, \quad \mathbf{v}_t^r = \mathbf{v}_t \mathbf{M}_r. \]
    
    \[ f_r(v_h, v_t) = \| \mathbf{v}_h^r + \mathbf{r} - \mathbf{v}_t^r \|_2^2. \]
  - 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.

  - [Diagram showing user and item feature extraction with DNNs for text, image, and video data, with matching function as inner product or cosine similarity.]
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)
     - **SA-NCF**: Self-Attentive NCF (Tay et al, 2018)
   - 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.

\[ \hat{y}_{ui} = f(p_u, q_i) \]

- Matching function based on NN

- Matching function: design whatever layers as you like.

- Input: design whatever features as you want.
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)
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 low-rank 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.
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)
     SA-NCF: Self-Attentive NCF (Tay et al, 2018)
   - 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., item category, description, image ...

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

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 $\mathbf{x}$

Target $y$

| $x^{(1)}$ | 1 0 0 ... | 1 0 0 0 ... | 0.3 0.3 0.3 0 ... |
| $x^{(2)}$ | 1 0 0 ... | 0 1 0 0 ... | 0.3 0.3 0.3 0 ... |
| $x^{(3)}$ | 1 0 0 ... | 0 0 1 0 ... | 0.3 0.3 0.3 0 ... |
| $x^{(4)}$ | 0 1 0 ... | 0 0 1 0 ... | 0 0 0.5 0.5 ... |
| $x^{(5)}$ | 0 1 0 ... | 0 0 0 1 ... | 0 0 0.5 0.5 ... |
| $x^{(6)}$ | 0 0 1 ... | 1 0 0 0 ... | 0.5 0 0.5 0 ... |
| $x^{(7)}$ | 0 0 1 ... | 0 0 1 0 ... | 0.5 0 0.5 0 ... |

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>User</th>
<th>TI</th>
<th>NH</th>
<th>SW</th>
<th>ST</th>
<th>Movie</th>
<th>TI</th>
<th>NH</th>
<th>SW</th>
<th>ST</th>
<th>Other Movies rated</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5</td>
<td>3</td>
<td></td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

114
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)

- The wide part is linear regression for memorizing seen feature interactions, which requires careful engineering on cross features. E.g., $\text{AND}(\text{gender}=\text{female}, \text{language}=\text{en})$ 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)

Crucial to overall performance
Deep Crossing (Shan et al, KDD’16)

Microsoft’s Sponsor Search Solution in 2016:

The use of residual layers makes the network be deep.
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).
Recall: Factorization Machine

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

\[
\hat{y}(x) = w_0 + \sum_{i=1}^{p} w_i x_i + \sum_{i=1}^{p} \sum_{j > i}^{p} \langle v_i, v_j \rangle x_i x_j
\]

- Note: self-interaction is not included: \( \langle v_i, v_i \rangle \).

First-order: Linear Regression

Second-order: pair-wise interactions between nonzero features

Only nonzero features are considered
NFM: Neural Factorization Machine (He and Chua, SIGIR’17)

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

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

Bilinear Interaction Pooling:

\[ f_{BI}(V_x) = \sum_{i=1}^{n} \sum_{j=i+1}^{n} x_i v_i \odot x_j v_j, \]

Figure 2: Neural Factorization Machines model (the first-order linear regression part is not shown for clarity).
DeepFM (Guo et al., IJCAI’17)

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

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

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

......
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

• Xiang Wang, Xiangnan He, Liqiang Nie, and Tat-Seng Chua. Item silk road: Recommending items from information domains to social users. In SIGIR 2017.
• Sheng Li, Jaya Kawale, and Yun Fu. Deep collaborative filtering via marginalized denoising auto-encoder. In CIKM 2015.
• Fuzheng, Zhang, Nicholas Jing Yuan, Defu Lian, Xing Xie, and Wei-Ying Ma. Collaborative knowledge base embedding for recommender systems. In KDD 2016.
• 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, and Tat-Seng Chua. Neural factorization machines for sparse predictive analytics. In SIGIR 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
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.

• **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!