Complementary-Similarity Learning using Quadruplet Network

Mansi Ranjit Mane, Stephen Guo, Kannan Achan
E-commerce Recommender Systems

Nancy
ecommerce Customer

Millions of items in eCommerce Catalog/Warehouse
Item Relationships

- Similarity/Substitutes
- Show similar items during exploration phase
Item
Relationships

- Complementary
- Show after purchase
  add example e.g. if someone has purchased dress, additional suggestions like sandals, purse etc
Challenges

• "Functional" or "Stylistic" Complementarity is different than "bought together"
• Lack of ground truth for complementary items
• Cold-start items
• Differentiation between similar and complementary items
Prior Work

• **Triple2vec (Wan, CIKM 18)**
  • Dual item embeddings
  • Maximize dot product between item pairs and user vectors
• Challenges:
  • Solves co-purchasing more than "complementary"
  • Transitivity leads to similar items in recommendations
  • Does not handle cold-start items
Prior Work

• Neural Complementary Recommender (ENCORE) - (Zhang, RecSys 18)

\[
P_{\text{comp}} = \sigma \left( d(a_f, c_f) \right) = \frac{1}{1 + e^{d(a_f, c_f) - \eta}} \quad \text{[4]}
\]

Challenges:
• Transitivity
Amazon Dataset

- Amazon Clothing, Shoes, and Jewelry data[2]
  - Category information and title
  - Complementary pairs: bought together by users from different categories
  - Similar pairs: items that lie in same category
  - Negative items: Randomly sample items which do not meet above criteria
  - Quads: anchor, complementary, similar, negative items
  - Train quads: 3.3M, Test quads: 0.3M

https://github.com/mansimane/quadnet-comp-sim
## Amazon Dataset Attribute Availability

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Coverage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Image</td>
<td>99.99</td>
</tr>
<tr>
<td>Description</td>
<td>5.68</td>
</tr>
<tr>
<td>Title</td>
<td>99.95</td>
</tr>
<tr>
<td>Price</td>
<td>38.29</td>
</tr>
<tr>
<td>Brand</td>
<td>6.25</td>
</tr>
</tbody>
</table>
Example Quadruplet

Anchor
Lee Dungarees Men’s Big, Tall Carpenter Jean

Complementary
Key Apparel Men's Big-Tall Short Sleeve Heavyweight Pocket Tee Shirt

Similar
Wrangler Men's Rugged Wear Relaxed Straight Fit Jean

Negative
Black and White Herringbone Wool Suiting Extra Long Tie
Motivation

• Why not just optimize for complementary items?

\[ P_{\text{comp}} = \sigma \left( d(a_f, c_f) \right) = \frac{1}{1 + e^{d(a_f, c_f) - \eta}} \] [4]
Motivation

Learnt Representation Space
Goal

• Learn representation space which can differentiate between similar and complementary items
Problem Formulation

- \(a\): Anchor item
- \(c\): Complementary item to anchor item
- \(s\): Similar item to anchor item
- \(n\): Negative item to anchor item
- \(a_f', c_f', s_f', n_f'\): Normalized learnt feature representation for \(a, c, s, n\)
Network Architecture

GUSE

Embedding

Shared non-trainable parameters

GUSE

Embedding

Shared trainable parameters

GUSE

Embedding

Shared non-trainable parameters

GUSE

Embedding

Shared trainable parameters

Quadruplet loss

$a_t$ $c_t$ $s_t$ $n_t$ $a_f$ $c_f$ $s_f$ $n_f$
Feature Representation

- Universal Sentence Encoder [1]
Similarity Loss

\[ L_{sim} = \max(d(a_f' - s_f') - m_s, 0) \]
Complementary Loss

\[ L_{comp} = \max\left( d(a_f, c_f) - m_c, 0 \right) + \max\left( m_s - d(a_f, c_f), 0 \right) \]
Negative Loss

\[ L_{neg} = \max(m_n - d(a'_f, n'_f), 0) \]
Quadruplet Loss

\[
\begin{align*}
L_{\text{sim}} &= \max(d(a'_f - s'_f) - m_s, 0) \\
L_{\text{comp}} &= \max(d(a'_f, c'_f) - m_c, 0) + \max(m_s - d(a'_f, c'_f), 0) \\
L_{\text{neg}} &= \max(m_n - d(a'_f, n'_f), 0) \\
L_{\text{quad}} &= L_{\text{sim}} + L_{\text{comp}} + L_{\text{neg}} + \lambda L_{l2}
\end{align*}
\]
Hyperparameters

• Input feature dimension: 512
• Epochs: 50
• Weight Initialization: Xavier
• Learning rate: 0.001
• $m_s : 0.1$
• $m_n : 0.4$
• $m_c : 0.8$
• Mapping function:

```
512 → GUSE → FC1 + ReLU → FC2 → 128
```
Experiments - Distance Distribution

Before Training

After Training
## Experiments - Distance Distribution

<table>
<thead>
<tr>
<th></th>
<th>Similar</th>
<th></th>
<th>Complementary</th>
<th></th>
<th>Negative</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Mean</td>
<td>Std Dev</td>
<td>Mean</td>
<td>Std Dev</td>
<td>Mean</td>
<td>Std Dev</td>
</tr>
<tr>
<td>Train Data Before training</td>
<td>0.82119</td>
<td>0.17611</td>
<td>0.81975</td>
<td>0.15910</td>
<td>0.99804</td>
<td>0.13286</td>
</tr>
<tr>
<td>Test Data Before training</td>
<td>0.82752</td>
<td>0.17853</td>
<td>0.83086</td>
<td>0.15937</td>
<td>1.0037</td>
<td>0.12949</td>
</tr>
<tr>
<td>Train Data After training</td>
<td>0.24069</td>
<td>0.11226</td>
<td>0.45845</td>
<td>0.11485</td>
<td>0.86774</td>
<td>0.27724</td>
</tr>
<tr>
<td>Test Data After training</td>
<td>0.24772</td>
<td>0.11485</td>
<td>0.45181</td>
<td>0.09963</td>
<td>0.86023</td>
<td>0.27182</td>
</tr>
</tbody>
</table>
Experiments Accuracy

<table>
<thead>
<tr>
<th>Method</th>
<th>Ranking Acc</th>
<th>Complementary Acc</th>
<th>Similarity Acc</th>
</tr>
</thead>
<tbody>
<tr>
<td>Universal Sentence Encoder</td>
<td>37.68</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Veit et al. [2]</td>
<td>14.92</td>
<td>91.05</td>
<td>56.45</td>
</tr>
<tr>
<td>Quadruplet Network</td>
<td><strong>67.15</strong></td>
<td>86.92</td>
<td><strong>68</strong></td>
</tr>
</tbody>
</table>

- Ranking accuracy is calculated as: $d_s < d_c < d_n$
- Complementary Accuracy: $\text{margin}_s < d_c < \text{margin}_c$
- Similarity Accuracy: $d_s < \text{margin}_s$
Inference (offline or RT)

- Use Approximate Nearest Neighbor (ANN) indices (FAISS, annoy, nmslib) to perform top-k embedding retrieval
- Divide all items into C disjoint groups (via taxonomy or clustering on embeddings)
- Create a set of ANN indices:
  - 1 global item index
  - 1 cluster centroid index
  - C separate cluster indices (of item embeddings associated with that cluster)
• Given query item, obtain query embedding from quadruplet network

• Given a query embedding:
  • Top K similar items:
    • Nearest neighbor (NN) query on the item index
  • Top K complementary items:
    • Goal: Find K closest items whose distance is in range \([ \text{margin}_c, \text{margin}_s ]\)
    • 1) NN query on the cluster index, consider the centroids in range
    • 2) For each in-range cluster, find the top K closest items. Merge the sets and retain the top K in range \([ \text{margin}_c, \text{margin}_s ]\)
Future Work

• Modelling asymmetry between relationships
• Large scale experiments on the Amazon dataset (and others) with more evaluation metrics
• Clustering analysis on learnt embedding space
References

Thank You

Stephen Guo: sguo@walmartlabs.com
Mansi Mane: mansi.mane@walmartlabs.com,
mansimane5@gmail.com