Experitment

三个问题：根据篇幅决定加不加

(1)

(2)

(3)

## Experimental Settings

The experimental environment is a workstation with Intel Core 8 Xeon(R) @3.50 GHz, GeForce GTX 2080, and 32 GB memory. The source code implemented based on keras[[1]](#footnote-1) is available on GitHub[[2]](#footnote-2).

### Dataset and samples.

Our dataset was crawled from ProgrammableWeb, the largest online Web service registry, on July 25, 2016. We removed the mashups and services without content information and the mashups that had only one component service from the original dataset. The experimental dataset contains 1,979 mashups and 728 services, and 5802 mashup-service invocations.

The sample we used was specially designed for the online interactive scenarios and consisted of a mashup to be built, one or more selected services and a service to be tested. Take a mashup, m, actually composed of service (a,b,c,d) as an example, we randomly chose a part of its component services as selected services(e.g., (a,b) ), and built a positive sample by combining m, (a,b), and a random component service except a and b, e.g., c. The sample can be written as (m,(a,b),c). In a similar way, we can build a negative sample (m,(a,b),e), where service e is a random service not invoked by m. The size of negative samples is 12 times that of positive samples in our sample set.

The service to be predicted and selected services can be resampled to increase the scale of samples. For example, based on the mashup m, we can choose service set (a),(b),(a,b),(c,d),(a,b,c),(a,c,d) as selected services and build corresponding samples, respectively. The diversified samples empower our model to serve for developers in all circumstances. Considering that the number of a mashup’s component services seldom exceeds 4, we chose 1,2 and 3 component services as selected services when building our sample set.

### Evaluation Metrics.

In this study, we evaluated different approaches based on five-fold cross-validation. The 1,979 mashups were divided into five folds. We took one fold for testing, and the others for training in each time. We adopted the following metrics to measure the recommendation result of an approach and averaged the metrics on five folds as the final metric.

Precision, Recall, and F1-measure at top N services in the ranking list are defined as:

|  |  |  |
| --- | --- | --- |
|  |  | (21) |
|  |  | (22) |
|  |  | (23) |

where is the set of mashups in the test set and denotes the size of . For mashup , is the recommended service list, while is its actual component services.

Mean average precision (MAP) at top N services in the ranking list is defined as:

|  |  |  |
| --- | --- | --- |
|  |  | (24) |

where indicates whether a service at position in the ranking list is an actual component service of , is the number of component services of , and denotes the number of actual component services of occurred in the top services of the ranking list.

Normalized discounted cumulative gain (NDCG) at top N services in the ranking list is defined as:

|  |  |  |
| --- | --- | --- |
|  |  | (25) |

where represents the ideal maximum DCG score that can be achieved for .

### Baseline Approaches.

In order to demonstrate the recommendation result of our model in online interaction scenarios, we selected several typical service recommendation algorithms for comparison. They cover content-based methods, CF-based methods, and hybrid algorithms.

* BPR-MF: the method uses Bayesian Personalized Ranking (BPR) to learn MF-based service recommendation models with a pairwise ranking loss.
* DHSR: The deep-learning-based approach combines collaborative filtering and content information. Specifically, in the neural CF part, the approach maps mashup’s and service’s ID into their embeddings and learns their interaction with an MLP.
* SFTN: the method first calculates two probabilities that a mashup invokes a service in the next round according to their content similarity and the historical interaction between neighbor mashups and the service, and then multiplies them as a final rating according to Bayes’ theorem.
* PasRec: the method first constructs a HIN using various objects in the service recommendation scenario, then measures an overall similarity between two mashups based on the meta-paths between them, and finally adopts a user-based CF method based on the similarity. It designs a pairwise loss function and employs BPR for model optimization.
* IsRec: its user-based-CF-style service recommendation framework can work well in online interaction developing scenarios. Compared with PasRec, it improves the calculation of content similarity with the help of word embedding and speeds up the search for neighbor mashups by clustering existing mashups offline and classifying a new mashup online.
* DINRec: it applies Deep Interest Network designed for CTR to service recommendation. It exploits all available features, including the features of the mashup to be built, selected services and the service to be tested, and learns their interaction in a well-designed network. Specifically, it utilizes local activation unit to activate related selected services when making a prediction on a service. Note that we enable the model to utilize the same feature as ours for a fair comparison in model structure.

Different from others, BPR-MF and DHSR contain model-based CF modules and cannot work well in online interactive scenarios. In the first stage, they fail to work due to lack of the information about selected services. In order to apply BPR-MF and DHSR into the second stage, we update their models whenever developers select new services.

## Performance of MISR

The performance comparison of different approaches in the first stage and the second stage are presented in table 3 and table 4.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Models | Precision@5 | | Recall@5 | | F1@5 | | NDCG@5 | | mAP@5 | |
|  | Stage1 | Stage2 | Stage1 | Stage2 | Stage1 | Stage2 | Stage1 | Stage2 | Stage1 | Stage2 |
| HDP | 0.0614 | 0.0373 | 0.1203 | 0.1114 | 0.0785 | 0.0510 | 0.1076 | 0.0860 | 0.0741 | 0.0670 |
| BPR | - | 0.0625 | - | 0.1802 | - | 0.0845 | - | 0.1332 | - | 0.1017 |
| DHSR | - | 0.1335 | - | 0.3474 | - | 0.1859 | - | 0.2738 | - | 0.2111 |
| SFTN | 0.1910 | 0.1169 | 0.3460 | 0.3227 | 0.2360 | 0.1562 | 0.3324 | 0.2551 | 0.2490 | 0.2034 |
| IsRec\_best | **0.2857** | 0.1580 | 0.5392 | 0.4317 | **0.3598** | 0.2102 | 0.5468 | 0.3737 | 0.4593 | 0.3177 |
| PasRec | 0.2841 | **0.1631** | **0.5420** | **0.4454** | **0.3598** | **0.2170** | **0.5553** | **0.3932** | **0.4717** | **0.3360** |
| DINRec | 0.2739 | 0.1597 | 0.5176 | 0.4382 | 0.3452 | 0.2129 | 0.5153 | 0.3765 | 0.4245 | 0.3149 |
| CI | 0.2771 | 0.1558 | 0.5247 | 0.4253 | 0.3497 | 0.2074 | 0.5253 | 0.3713 | 0.4352 | 0.3122 |
| NI | 0.2786 | 0.1579 | 0.5228 | 0.4283 | 0.3500 | 0.2096 | 0.5421 | 0.3738 | 0.4560 | 0.3151 |
| Hybrid | **0.2962** | **0.1678** | **0.5692** | **0.4574** | **0.3766** | **0.2233** | **0.5673** | **0.3993** | **0.4747** | **0.3379** |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Models | Precision@10 | | Recall@10 | | F1@10 | | NDCG@10 | | mAP@10 | |
|  | Stage1 | Stage2 | Stage1 | Stage2 | Stage1 | Stage2 | Stage1 | Stage2 | Stage1 | Stage2 |
| HDP | 0.0484 | 0.0306 | 0.1854 | 0.1750 | 0.0743 | 0.0484 | 0.1356 | 0.1097 | 0.0853 | 0.0764 |
| BPR | - | 0.0426 | - | 0.2438 | - | 0.0675 | - | 0.1567 | - | 0.1114 |
| DHSR | - | 0.0855 | - | 0.4459 | - | 0.1400 | - | 0.3116 | - | 0.2290 |
| SFTN | 0.1198 | 0.0754 | 0.4365 | 0.4135 | 0.1815 | 0.1187 | 0.3717 | 0.2895 | 0.2707 | 0.2193 |
| IsRec\_best | **0.1714** | 0.1004 | **0.6326** | 0.5238 | **0.2609** | 0.1558 | 0.5889 | 0.4098 | 0.4867 | 0.3359 |
| PasRec | 0.1666 | 0.1027 | 0.6205 | 0.5371 | 0.2545 | 0.1598 | **0.5909** | **0.4298** | **0.4949** | **0.3545** |
| DINRec | 0.1638 | **0.1035** | 0.6025 | **0.5395** | 0.2492 | **0.1611** | 0.5540 | 0.4172 | 0.4491 | 0.3361 |
| CI | 0.1647 | 0.1022 | 0.6093 | 0.5291 | 0.2511 | 0.1584 | 0.5636 | 0.4128 | 0.4594 | 0.3335 |
| NI | 0.1670 | 0.1002 | 0.6132 | 0.5236 | 0.2538 | 0.1559 | 0.5827 | 0.4115 | 0.4816 | 0.3343 |
| Hybrid | **0.1771** | **0.1072** | **0.6617** | **0.5572** | **0.2705** | **0.1665** | **0.6090** | **0.4391** | **0.5017** | **0.3583** |

The recommendation of BPR-MF is the worst among all methods, which indicates that model-based CF is not applicable to interactive recommendation scenario. In order to get the embedding of a mashup built online, BPR-MF needs to update its model according to the selected services of the mashup. However, the number of selected services is usually too small to guarantee a high-quality embedding of the mashup.

Compared with BPR-MF, SFTN also considers content information during recommendation and get a performance improvement, but its result is still poor. This is because SFTN multiplies the two probabilities obtained from content and historical information, with the assumption that they are independent of each other. But the correctness of this assumption is doubtful because we are unaware of the way how these two information affect service recommendation.

DHSR integrates CF and content information into a network, but the performance of this deep-learning-based method is not as expected. Firstly, it adopts a double-tower structure to learn the interaction between a mashup to be built and a service to be tested, but does not obviously consider their interaction with selected services. Secondly, it suffers the same problem as BPR, that is, it cannot get an effective representation of a mashup built online according to the limited component services that the mashup selects online.

As a deep-learning-based method, DIN outperforms DHSR to a great extent, but is not the best in all baselines. The method designed for e-commerce enforces various features to interact with each other sufficiently. However, in the service recommendation scenario, the features are extracted from content information and historical invocation respectively and in two completely independent spaces, so there is no need to combine them across domains. Our hybrid model first learns two internal interactions in these two spaces and then fuses them with an MLP, reducing the difficulty and increasing the accuracy of interaction learning.

PasRec and IsRec perform best in all baselines. Compared with SFTN, they find more accurate neighbor mashups by organizing various information using HIN and employing meta-path-based similarity measurement. Meanwhile they carry out an effective optimization to their model based on BPR.

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Models | P@10 | Gain | R@10 | Gain | F1@10 | Gain | NDCG@10 | Gain | mAP@10 | Gain |
| 1 | IsRec\_best | 0.1067 | 0.3% | 0.6322 | -1.4% | 0.1716 | -0.4% | 0.5246 | -2.2% | 0.4551 | -3.4% |
| PasRec | 0.1060 | 0.4% | 0.6301 | -1.0% | 0.1707 | 0.1% | 0.5241 | -2.1% | 0.4549 | -3.4% |
| DINRec | 0.1024 | 4.0% | 0.6001 | 3.9% | 0.1646 | 3.9% | 0.4854 | 5.7% | 0.4119 | 6.7% |
| Hybrid | 0.1064 | - | 0.6236 | - | 0.1710 | - | 0.5130 | - | 0.4394 | - |
| 2 | IsRec\_best | 0.0987 | 6.8% | 0.5062 | 7.7% | 0.1526 | 7.2% | 0.3858 | 8.9% | 0.3105 | 9.0% |
| PasRec | 0.1001 | 5.3% | 0.5166 | 5.6% | 0.1553 | 5.3% | 0.4057 | 3.5% | 0.3284 | 3.0% |
| DINRec | 0.1019 | 3.4% | 0.5280 | 3.3% | 0.1584 | 3.3% | 0.4013 | 4.7% | 0.3198 | 5.8% |
| Hybrid | 0.1054 | - | 0.5453 | - | 0.1635 | - | 0.4200 | - | 0.3383 | - |
| 3 | IsRec\_best | 0.0958 | 14.5% | 0.4330 | 16.1% | 0.1431 | 15.3% | 0.3189 | 20.5% | 0.2422 | 22.7% |
| PasRec | 0.1021 | 7.5% | 0.4645 | 8.2% | 0.1532 | 7.6% | 0.3597 | 6.8% | 0.2804 | 6.0% |
| DINRec | 0.1062 | 3.3% | 0.4902 | 2.5% | 0.1602 | 2.9% | 0.3649 | 5.3% | 0.2767 | 7.4% |
| Hybrid | 0.1097 | - | 0.5027 | - | 0.1649 | - | 0.3842 | - | 0.2973 | - |

Compared with PasRec, the best baseline, our hybrid model achieves better results. The NDCG@5, MAP@5, Precision@5, Recall@5, and F1@5 values of ours were higher than those of PasRec by 22.6%,27.6% and 13.1%, respectively. The reason is that PasRec directly estimates a scalar score of a service over a new mashup based on the historical invocation between neighbor mashups and the service, without digging into their underlying interactions. Our model sufficiently exploits the complex interaction among the mashup, selected services and the service by an elaborate interaction layer equipped with an MLP and an attention block. Moreover, our model integrates content interaction and neighbor interaction when making final predictions.

2. Integration strategies

Our model predicts the probability of a mashup with selected services invoking a service based on their interaction. In order to learn the complex interaction more efficiently, we need to integrate the representation of each selected service and get an overall representation of all selected services in some way. We employ attention mechanism to accomplish this task in our model. To compare the impact of different integration strategies on the model's recommendation result, we adopt other three strategies to replace the attention mechanism, and propose three model variants.

MISR-average: it preforms average pooling on the representation of each service and assigns equal weight to them.

MISR-concatenate: it directly concatenates the representation of each service. We truncate or pad the selected service set to get a final representation with a fixed size.

MISR-None: it discards selected services and does not consider them when learning the complex interaction.

We classified the testing samples according to the number of mashup’s selected services and evaluated the performances of different models in these cases. The comparison results are shown in Figure 1.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | FI | | | | NI | | | |
| Mode | Size | F1@10 | Gain | NDCG@10 | Gain | F1@10 | Gain | NDCG@10 | Gain |
| Attention | 1 | 0.1644 | - | 0.4884 | - | 0.1621 | - | 0.4819 | - |
| 2 | 0.1563 | - | 0.3923 | - | 0.1543 | - | 0.3913 | - |
| 3 | 0.1543 | - | 0.3576 | - | 0.1512 | - | 0.3614 | - |
| None | 1 | 0.1615 | 1.8% | 0.4784 | 2.1% | 0.1618 | 0.1% | 0.5035 | 0.7% |
| 2 | 0.1462 | 6.9% | 0.3646 | 7.6% | 0.1485 | 3.9% | 0.3896 | 2.5% |
| 3 | 0.1402 | 10.1% | 0.3093 | 15.6% | 0.1427 | 6.0% | 0.3318 | 8.9% |
| Concate | 1 | 0.1626 | 1.1% | 0.4847 | 0.8% | 0.1607 | 0.9% | 0.4742 | 1.6% |
| 2 | 0.1517 | 3.0% | 0.3806 | 3.1% | 0.1475 | 4.6% | 0.3698 | 5.8% |
| 3 | 0.1487 | 3.8% | 0.3432 | 4.2% | 0.1443 | 4.8% | 0.3437 | 5.2% |
| Average | 1 | 0.1634 | 0.6% | 0.4821 | 1.3% | 0.1616 | 0.3% | 0.4763 | 1.2% |
| 2 | 0.1522 | 2.7% | 0.3833 | 2.4% | 0.1519 | 1.6% | 0.3857 | 1.4% |
| 3 | 0.1481 | 4.2% | 0.3388 | 5.5% | 0.1470 | 2.9% | 0.3493 | 3.5% |

The performance of MISR-None is always the worst, indicating that selected services do play an important role in online interactive recommendation and we should make the best of them. When developers have selected only one service, there is no obvious difference among the performances of MISR, MISR- concatenate and MISR- average. When developers have selected two and more service, our attention-based model performs better than MISR-average and MISR-concatenate. This is because it pays more attention to the selected services that are more related to current prediction, assigns distinct weight to each selected service, and obtains an adaptable representation specific to different service to be tested.

However, the improvement of attention mechanism relative to average pooling and concatenation is not as prominent as expected. A possible reason is that the number of selected services in our experiments is small, i.e., not bigger than 3. This can be verified by the observation that the improvement gradually increases as the number of selected services increases.

3.Top K

In the neighbor interaction module, we infer the interaction among a new mashup and a service based on those among neighbor mashups and the service. This subsection will look for an optimal value for neighbor size,, that helps the neighbor interaction module achieves best result. The candidate value is set from 10 to 50 by step 10.

Table 3. Recommendation Performance with Different K

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| K | P@10 | R@10 | F1@10 | NDCG@10 | mAP@10 |
| 10 | 0.0981 | 0.5433 | 0.1552 | 0.4340 | 0.3607 |
| 20 | **0.1005** | **0.5542** | **0.1587** | **0.4434** | **0.3686** |
| 30 | 0.0982 | 0.5398 | 0.1549 | 0.4348 | 0.3621 |
| 40 | 0.0977 | 0.5357 | 0.1539 | 0.4290 | 0.3561 |
| 50 | 0.0968 | 0.5341 | 0.1528 | 0.4264 | 0.3539 |

As shown in Table 3, the indicators gradually increase as the value of K increases from 10 to 30. The reason is that the module can absorb and exploit more beneficial information from neighbor interactions. However, the opposite is true when K exceeds 30. This may be because the introduction of noisy data causing negative interference to interactive learning. Therefore, we set to 30 in our experiments.

1. https://keras.io [↑](#footnote-ref-1)
2. https://github.com/ssea-lab/MISR [↑](#footnote-ref-2)