Attentional Matrix Factorization with Document-context awareness and Implicit API Relationship for Service Recommendation

Mo Nguyen
thithuymo.nguyen@aut.ac.nz
Auckland University of Technology
Auckland, New Zealand

Jian Yu
jian.yu@aut.ac.nz
Auckland University of Technology
Auckland, New Zealand

Sira Yongchareon
sira.yongchareon@aut.ac.nz
Auckland University of Technology
Auckland, New Zealand

Yanbo Han
yhan@ncut.edu.cn
North China University of Technology
Beijing, China

Guiling Wang
wangguiling@ncut.edu.cn
North China University of Technology
Beijing, China

ABSTRACT

The rapid development of the mashup approach significantly plays a pivot role in building Web-based and mobile applications. Easy accessibility of data and functions are the main advantages for programmers to develop mashups from abundant sources of APIs. However, it simultaneously brings more difficulties to choose suitable APIs for a mashup, especially when the historical relations between APIs and mashups are very sparse. Existing probabilistic matrix factorization (PMF) recommender systems can effectively exploit the latent features of the invocations with the same weight. However, not all features are equally significant and predictive, and the useless features may bring noises to the model. Also, many current works explored the influence of mashups' relationships, but few of them sheds lights on the relationship between APIs and their contextual interaction, which can be mined from their content description. This paper improves the PMF model by distinguishing the importance of latent feature interactions. We present an Attentional PMF model, which leverages a neural attention network to learn the significance of feature interactions and uses Doc2Vec technique for mining the contextual information. We also exploit the relationship between APIs from both their contextual similarities and invocation history and add them to the prediction model as a regularization part. Our experiments are performed with datasets from ProgrammableWeb. The results show that our model significantly outperforms some state-of-art recommender systems in mashup service applications.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
Melbourne'20, Feb 02–07, 2020, Melbourne, Australia
© 2020 Association for Computing Machinery.
ACM ISBN 978-1-4503-7697-6/20/02…$15.00
https://doi.org/10.1145/3373017.3373034

CCS CONCEPTS

• **Information systems** → **Data mining**; **Web searching and information discovery**; **Web services**; *Collaborative and social computing systems and tools.*

KEYWORDS

recommender systems, contextual interaction, invocation, attention network

ACM Reference Format:
Mo Nguyen, Jian Yu, Sira Yongchareon, Yanbo Han, and Guiling Wang. 2020. Attentional Matrix Factorization with Document-context awareness and Implicit API Relationship for Service Recommendation. In *Melbourne '20: ACM Australasian Web Conference 2020, Feb 02–07, 2020, Melbourne, Australia.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3373017.3373034

1 INTRODUCTION

Web APIs have been increasingly developed and widely used for Web-based and mobile applications over past years. For instance, programmers in different application fields are able to integrate Google Maps into their products with several coding lines in Google Maps API. ProgrammableWeb has been an largest online repository in providing Internet-based APIs to programmers who are seeking for the real power of APIs to integrate them for mashup services. Furthermore, ProgrammableWeb is also the most extensively-cited source of data with incorporation of API-related statistics in key media, journals, conferences, etc. However, the accelerated growth in the quantity of APIs has brought difficulties for programmers to choose the right APIs from a large number of candidates and a more sparse rating data also brought more challenges for the APIs recommendation system to propose a suitable list of APIs. Hence, it is highly demanded to develop better recommendation technologies for programmers to pinpoint the proper APIs for mashups developing. Much research effort using Collaborative Filtering (CF) method has been done to address this problem, However, there are still some limitation as below.

First, existing CF methods generally can be classified into memory-based and model-based [2]. Model-based methods are more well-known with better accurate prediction and PMF [11] is the most popular model-based method. A lot of PMF-based recommender

RIGHTSLINK

systems have been developed and widely used in service recommendation [2]. However, current PMF-based models aim to learn the latent features of mashups and APIs on a latent space with the same weight and predict their invocations by computing the dot product of their two latent feature vectors. In reality, different historical invocations usually have different power of prediction and not all latent features contribute useful information to anticipate the future invocations. Hence, in order to prevent the invocation with less useful features from deteriorating the recommender system, it should be attached a lower weight or attentional score. However, current PMF-based models are short of this ability to make distinction for the importance of invocations or feature interactions, which may produce substandard prediction.

Second, some hybrid models leverage the auxiliary information such as trustworthy information of users [8], [13], and description document of items [7]. Namely, the description of items such as reviews, introductions are from different sources, and they represent for items' attributes. Hence they have been used as descriptive features that can increase the rating prediction accuracy, especially in a sparse rating dataset. Some document modelling methods were used to mine such documents such as Latent Dirichlet Allocation (*LDA*) [1] and Stacked Denoising Auto-Encoder (*SDAE*). Nevertheless, there are few such context-aware models applied in Web APIs applications, and they also have some limitations in APIs rating prediction. Zhong et al. [19] explore the description of both APIs and mashups with their historical interactions to reconstruct the profile for services which are used as auxiliary information for rating prediction. However, this study only considers the bag-of-words technique to explore the content of the description document, while the words' sequences and surrounding words are ignored. Hence, such valuable information is not effectively explored for prediction when the rating is very sparse.

Third, much research effort has been put in exploring the mashup-API invocation history, but few of them discover APIs' co-invocation. Some APIs are either always in the same mashup or never work together. Therefore, we believe that the compatibility of APIs is essential for the invocation preferences of a mashup and the learning of latent features of APIs and mashups. Yao et al. [18] use TF-IDF and Poisson stochastic process to explore the similarities and the implicit relationship between APIs, and then integrate it to a general matrix factorization framework as a pre-trained element. However, this work disregards the contextual information and only learns the latent features from mashup-API invocation history when defining the API's similarity.

This paper improves the conventional PMF model by considering the importance of the binary mashup-API invocations through learning their latent features. Namely:

- We proposed an Attentional PMF model, which leverages the attention mechanism in neural network modelling [4] to weight the latent features interaction of mashup-API invocations.

- We define an attentional network to parametrize the attention score and learn it together with the Attentional PMF model.

- We use Doc2Vec technique to exploit the contextual information of APIs and mashups, and uses it as auxiliary data to learn the latent features. We do statistical analysis for API co-invocation data and learn the distribution of co-invocation frequency for each pair of APIs. Later on, we integrate the contextual similarity and co-invocation prevalence as regularization parts of the Attentional PMF model.

- We conduct experiments on ProgrammableWeb dataset with different percentages masked training dataset, and the results show that out model outperforms all the baseline models.

The rest of paper is organized as follows. Section 2 provides some related works of recommender systems and presents the problem statement. Section 3 first gives an overview of our proposed approach, which describes the framework of APIs recommendation for mashup development and then defines the Attentional PMF model and regularization parts. Section 4 demonstrates some experimental results. Finally, Section 5 concludes the paper.

2 PRELIMINARIES

2.1 Related Works

One of the most popular model-based Collaborative Filtering is a matrix factorization model (PMF) [11], which leverages the mashup-API invocation matrix to obtain the latent features of mashups and APIs. Koren et al. [5] propose SVD based Matrix Factorization applies singular value decomposition. Such methods decompose the large invocation matrix into smaller dimension matrices which contain the latent features. Later on, in order to overcome the problem of over-fitting of such models, regularization is added to PMF. to factorize the mashup-API invocation matrix into three matrices A, Σ, and M. This factorization model can be simplified further as the product of two lower-dimensional latent matrices $R = A^T M$ where A and M are composed of the respective vectors a_i and m_j such that $r_{ij} = {a_i}^T m_j$. Finding the vectors is solved by the optimization process of below objective function:

$$\min_{A,M} \sum_{i=1}^{x} \sum_{j=1}^{y} (r_{ij} - {a_i}^T m_j)^2 \tag{1}$$

Recently, the significance of the full neural network structure for learning useful word representations has been shed in light. Mikolov [9] has proposed the skip-gram and continuous bag-of-word which is a simple single-layer architecture by implementing the inner product between pairs of word vectors. Mnih and Kavukcuoglu [10] defined vLBL and ivLBL as closely-related vector log-bilinear models. The skip-gram and ivLBL aim to anticipate the context of a word provided the word itself while CBOW and vlBL models anticipate a word based on its context. Such models leverage the linear relationship among word vectors to learn the linguistic motif. Different from PMF, the shallow window-based methods have the limitation that they ignore the co-occurrence statistics of the corpus. Instead, they just move through the entire corpus and scan context windows, which fails to take advantage of the abundant repeated data.

To learn document embedding, Le [6] recently proposed Doc2Vec technique, which is an extension of Word2Vec [9]. Doc2Vec has two approaches distributed DBOW and distributed memory DMPV. DBOW is similar to CBOW which calculate the document vector by concatenating all the paragraph vectors without considering word order. Reversely, DMPV uses the idea that is adding a unique vector presenting for the document into the learning word vectors model. This paper leverages Doc2Vec with DMPV approach to pre-train

RIGHTSLINK

the document embedding vectors for all descriptions of mashups and APIs.

Also, several service recommender systems currently focus on context to calculate the contextual similarity of services' description and recommend suitable APIs which have the highest similarity. Yao et al. [17] define a novel approach to unify collaborative filtering and content-based recommender systems. Naïm et al. [12] integrate pattern mining into probabilistic topic models to capture the maximal common context of API for recommender system. Cao et al. [3] use the content similarity between services to recommend APIs for mashups by matrix factorization approach. Rahman et al. [14] propose a matrix factorization method which integrates content and network-based service clustering and makes a recommendation from a comparable shortlist of suitable APIs.

2.2 Problem statement

In APIs discovery platform such as ProgrammableWeb, when a programmer wants to build a mashup application, the system generates a list of suitable APIs which are supposed to be closest to mashup's requirements. This paper assumes these requirements as descriptions provided by the programmer, which includes functions, business tasks, dependencies, efficiencies of the mashup. Namely, a new mashup application is required to provide a food ordering and delivery service, that can be used for customers to order and have food delivered. The mashup is required to provide a notorious and easy interface to explore, seek, reserve, make a secure payment, and give feedback for the restaurant's service. Available APIs are abundant and diverse, so the programmer has to choose just a small number of APIs in this enormous repository. Figure 1 clearly shows that with over 3500 available APIs, the average number of APIs used in most of the mashups is just around 5. Figure 2 shows the distributions of APIs over mashups. Each red dot is presented for the invocation between a mashup and an API. It can be seen that there are many APIs that have never been used by any mashup. Therefore, finding and suggesting a set of most compatible APIs is a challenging problem for recommendation systems. As traditional approaches [2], we focus on the invocation history of APIs and mashups to learn the latent feature and define the similarity between APIs and mashups from a matrix factorization model. However, we differently integrate the attentional score to invocation through the learning latent features process and investigates the context-aware documents of APIs and mashups, which has not been considered by many types of research. Also, some APIs are compatible with the same mashup, and some others are not. Thus, we explore how to derive the implicit relationship of APIs and integrate it to Attentional PMF model for better rating prediction.

3 PROPOSED APPROACHES

3.1 Overview

To address the above challenges, this paper proposes a framework of APIs recommendation for mashups development, as demonstrated in Fig.3. Namely, when an active user wants to make a new mashup, she would type some descriptions about the mashup. On the one hand, available APIs currently in the repository also have their descriptions. The descriptions from both new mashup and current APIs are used as input for a word embedding model which uses Doc2vec technique [6] to generate words to vectors and then a document matrix. The output of this model is the document matrix of APIs and the new mashup. After that, the Cosine methods used to obtain the similarity between the mashup and each APIs by using their document matrix. On the other hand, APIs invocation history provides information about their participation in the mashups. The implicit relationship of APIs is learned from their prevalence of being in the same mashups. In the next step, mashup-API similarity and APIs implicit relationship is integrated into an Attentional PMF model as regularization terms. A list of APIs with highest recommendation scores will be added to a recommendation list for the new mashup development.

3.2 Attentional PMF Model

We suppose we have an invocation record of x APIs and y mashups. We denote the invocation relationship between APIs and mashups by a matrix R$\in R^{x\times y}$. We use a binary variable r_{ij}, which indicates an invocation pairing of API a_i and mashup m_j. If API a_i is invoked by mashup m_j then r_{ij} is equal to 1, and 0 otherwise. We define $\hat{r_{ij}}$ as the probability that API a_i will be invoked by mashup m_j.

$$r_{ij} = \left\{ \begin{array}{cc} 1 & \text{if } m_j \text{ invoke } a_i \\ 0 & \text{otherwise} \end{array} \right\}. \tag{2}$$

3.2.1 Attentional network. Figure. 4 demonstrates the neural network architecture of our proposed model. The input layer is a sparse invocation matrix, and it is factorized into two matrices A$\in R^{x\times d}$ and matrix M$\in R^{y\times d}$ which represent for latent features of mashups and APIs respectively. In the embedding layer, different from PMF which defines $\hat{r}_{ij}$ as the dot product of two latent feature vectors $a_i \in R^d$ and $m_j \in R^d$, we add an attentional score to present the importance of the interaction into the prediction of $\hat{r_{ij}}$ as below:

$$\hat{r_{ij}} = \alpha_{ij} {a_i}^T m_j \tag{3}$$

Where α_{ij} is the attentional weight for the invocation between API a_i and mashup m_j, which represents for the importance of the dot product ${a_i}^T m_j$ to the predicted value. To learn α_{ij}, we use a similar idea with Xiao et al. [16] who proposed an attentional network with a multi-layer perceptron (MLP). The advantage of this work [16] over PMF is that it can estimate the attentional scores of these invocations which have never happened in the training data. We differently define attentional score as a scalar, and it is formally defined as below:

$$\alpha'_{ij} = hReLU({a_i}^T m_j + b) \tag{4}$$

$$\alpha_{ij} = \frac{e^{a'_{ij}}}{\sum_{i=1}^{x}\sum_{j}^{y} a'_{ij}} \tag{5}$$

We simultaneously learn α_{ij} and compute the features of mashups and APIs by minimizing an attentional sum of squared errors loss function of a PMF model in the following section.

3.2.2 Probabilistic matrix factorization model. Figure 5 shows an overview of how to integrate attentional network into PMF in the form of probabilistic graphical model. The loss function is defined

RIGHTSLINK

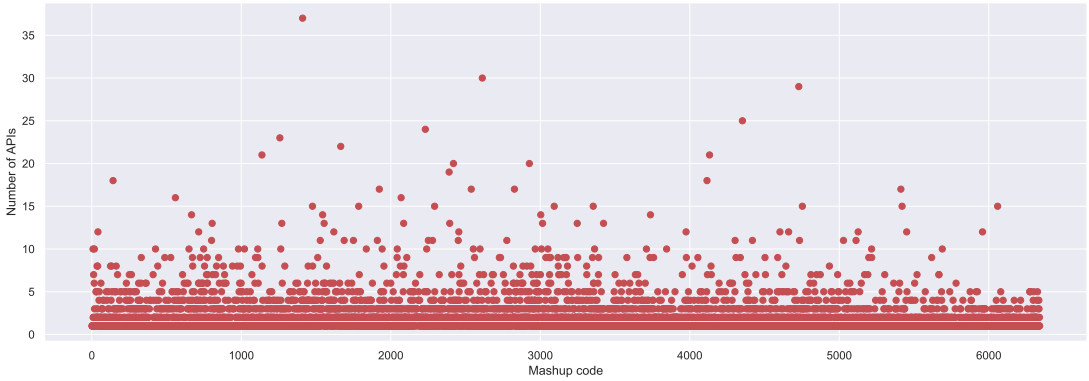


Figure 1: Number of APIs in a Mashup

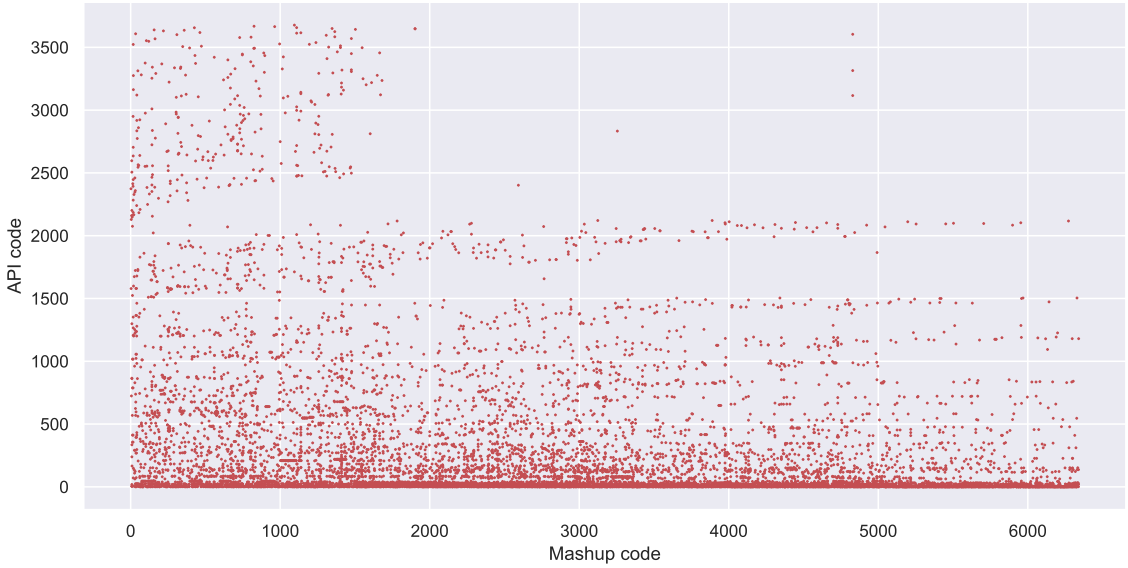


Figure 2: Distribution of APIs and mashup

as follows:

$$\min_{A,M} L = \sum_{i=1}^{x}\sum_{j=1}^{y}\frac{1}{2}I_{ij}(r_{ij} - \alpha_{ij}{a_i}^T m_j)^2 + \frac{\lambda_a}{2}\|A\|_F^2 + \frac{\lambda_m}{2}\|M\|_F^2 \quad (6)$$

, where I_{ij} equals 1 if API a_i is invoked by mashup m_j, and 0 otherwise. Regularization parameters are λ_a and λ_m. $\|.\|_F$ is the Frobenius norm.

3.3 Mashup recommendation with integrated with document-context awareness and implicit API relationship

We leverage the significant advantage of PMF model which allows to integrate the auxiliary information to reduce the model overfitting and enhance the prediction accuracy. Namely, we exploit the information of document-context and implicit API relationship for the proposed Attentional PMF model. Generally, in recommender systems, both explicit and implicit reviews can be utilized to derive users preference over items. While the explicit reviews indicate the occurrence of user-item interactions, the implicit reviews obliquely

RIGHTSLINK

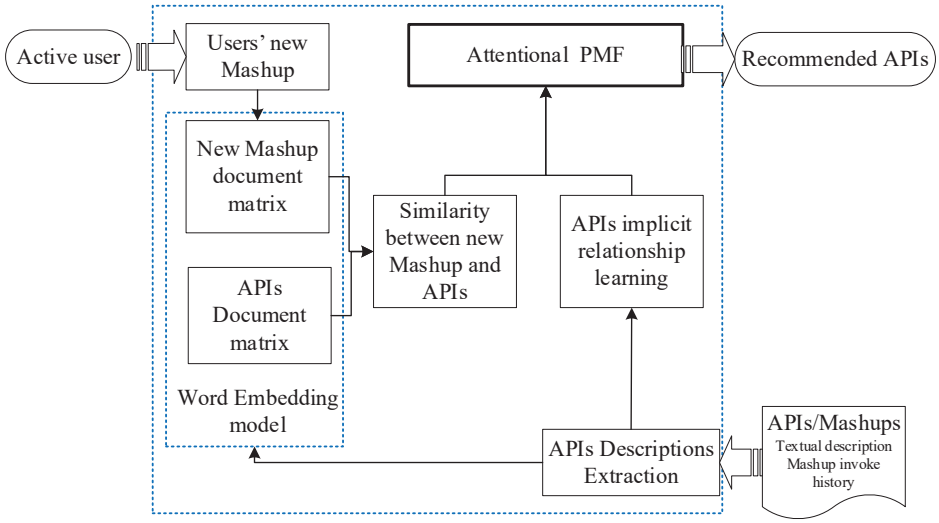


Figure 3: The framework of service recommender system.

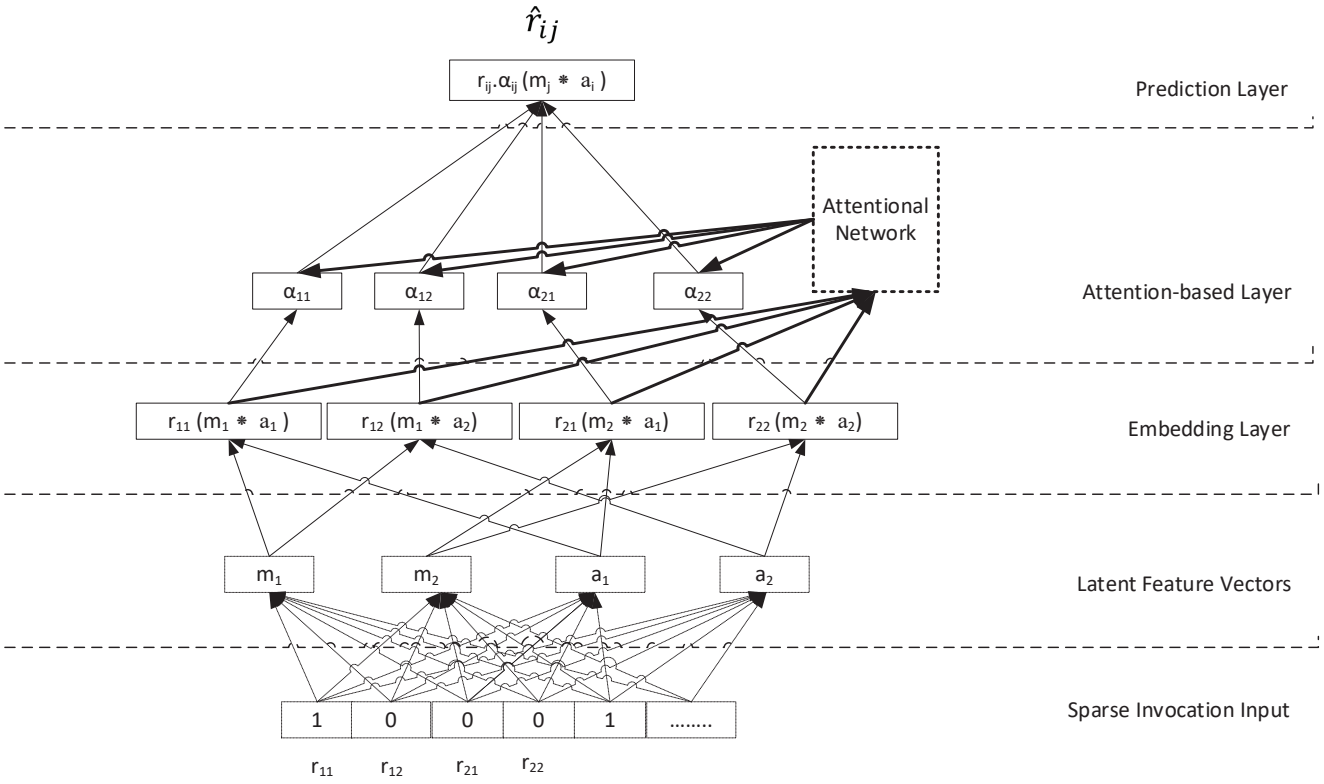


Figure 4: The neural network architecture of Attentional PMF Model

reflects users' sentiment by paying attention to their previous historical interaction. In service recommender systems, mashups are considered as users, APIs are corresponding to items, and reviews are invocations. We interpret the service co-invocation as implicit API relationship and incorporate it with document-context awareness

RIGHTSLINK

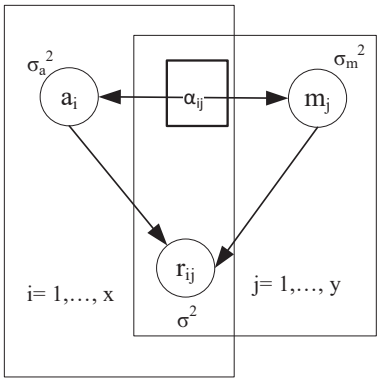


Figure 5: The graphical model for Attentional PMF

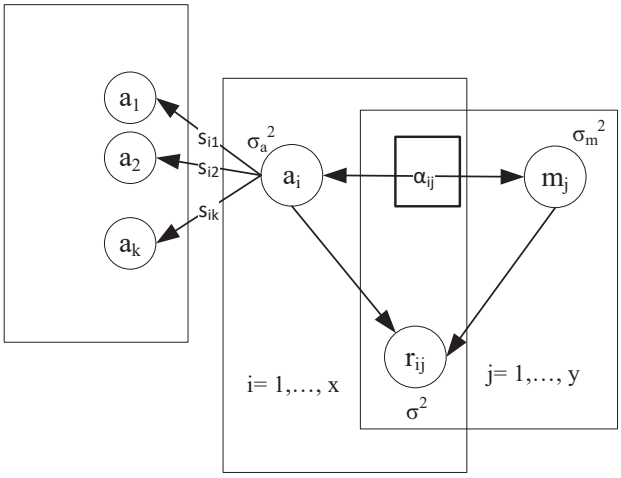


Figure 6: The graphical model for Attentional PMF with regularization.

to execute recommendation based on such implicit co-invocations learned from the existing historical data.

Namely, we integrate the document-context and the implicit relationship for API recommendation to our Attentional PMF model by two regularizing terms. We implement a document mining with Doc2vec technique for the first regularization term and then use distribution fitting for co-invocation analysis to build the second term as following.

3.3.1 Document mining with Doc2vec word embedding technique . To prepare the input for the word embedding model, we first collect the data of APIs descriptions from ProgrammableWeb, including name, category, tags, introduction, and historical mashups. After that, we execute the preprocessing to extract stemming. We do not remove stop words because they all influence the context of the document. In the next step, we generate document latent vector D from the descriptions which will be used in our recommendation

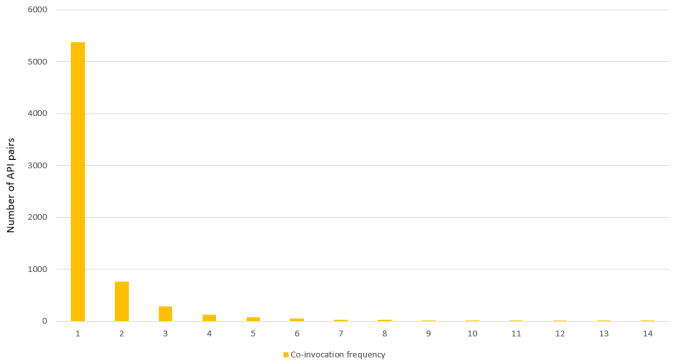


Figure 7: Co-invocation frequency of API pairs

system model. Namely,

$$D = \begin{bmatrix} | & | & | & & | \\ v_D & v_{w_1} & v_{w_2} & \dots & v_{w_V} \\ | & | & | & & | \end{bmatrix} \tag{7}$$

In Equation 7, D$\in R^{N\times(V+1)}$, V is the length of the document, and N is the number of latent features according to embedding dimensions for each word w_v with v$\in${1,..,V}. Contextual similarity between a new mashup m_j with $m_j \in R^y$ and API a_i with $a_i \in R^x$ is denoted by s_{ij} and obtained by the distance between two vector $v_D(m_j)$ and $v_D(a_i)$ by Euclidean norm as below:

$$s_{ij} = ||v_D(m_j), v_D(a_i)|| \tag{8}$$

It is noted that the similarity s_{ij} is added to attach the contextual information to the prediction as a regularization term as below:

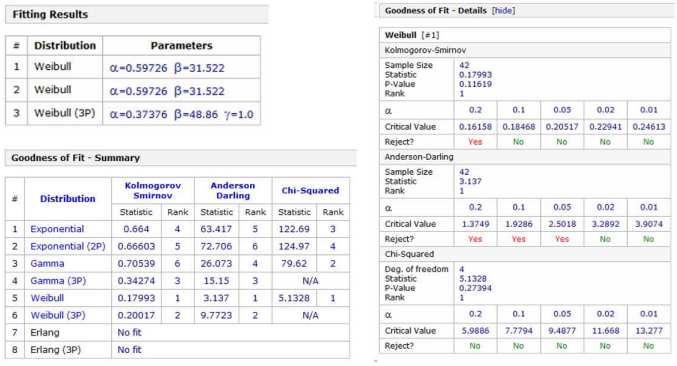
$$\frac{\lambda_S s_{ij}}{2} \|MA^T\|_F^2 \tag{9}$$

3.3.2 Implicit APIs relationship mining. The recommended APIs have to be appropriate for the new mashup and also do not conflict with each other. In other words, APIs used in the same mashup must be compatible and strong correlated. Therefore, there must be some latent elements influencing the co-invocation of APIs, and such latent relationship cannot be directly observed from APIs' description. [18] proposed a pre-train model to learn this implicit correlation from textual profile and co-invocation history, and add it to the main PMF as a regularization part. This work uses TF-IDF to explore the contextual similarity of APIs and assumes the Poisson distribution for co-invocation counts for each pair of APIs. In this paper, we do not pre-train the model but implement statistical analysis for the co-invocation counts and withdraw a suitable distribution, which effectively represents for the probability of being invoked by the same mashup of APIs. Using Easyfit, we fit the distributions for this data and obtain that Weibull [15] is the best distribution with parameters $\lambda = 0.59726$ and $k = 31.522$ as in Figure 8.

$$P_{ik} = \frac{k}{\lambda}\left(\frac{f_{ik}}{\lambda}\right)^{k-1} e^{\left(\frac{-f_{ik}}{\lambda}\right)^k} \tag{10}$$

Where f_{ik} is the co-invocation frequency of API a_i and mashup m_j. APIs which are used to be invoked in the same mashup have much chance to be together in the future because of their latent relevant features. Namely, if API a_i has a co-invocation list of APIs

RIGHTSLINK



Fitting Results

#	Distribution	Parameters
1	Weibull	α=0.59726 β=31.522
2	Weibull	α=0.59726 β=31.522
3	Weibull (3P)	α=0.37376 β=48.86 γ=1.0

Goodness of Fit - Summary

#	Distribution	Kolmogorov Smirnov		Anderson Darling		Chi-Squared	
		Statistic	Rank	Statistic	Rank	Statistic	Rank
1	Exponential	0.664	4	63.417	5	122.69	3
2	Exponential (2P)	0.66603	5	72.706	6	124.97	4
3	Gamma	0.70539	6	26.073	4	79.62	2
4	Gamma (3P)	0.34274	3	15.15	3	N/A	
5	Weibull	0.17993	1	3.137	1	5.1328	1
6	Weibull (3P)	0.20017	2	9.7723	2	N/A	
7	Erlang	No fit					
8	Erlang (3P)	No fit					

Goodness of Fit - Details [hide]

Weibull [#1]

Kolmogorov-Smirnov

Sample Size	42
Statistic	0.17993
P-Value	0.11619
Rank	1

α	0.2	0.1	0.05	0.02	0.01
Critical Value	0.16158	0.18468	0.20517	0.22941	0.24613
Reject?	Yes	No	No	No	No

Anderson-Darling

Sample Size	42
Statistic	3.137
Rank	1

α	0.2	0.1	0.05	0.02	0.01
Critical Value	1.3749	1.9286	2.5018	3.2892	3.9074
Reject?	Yes	Yes	Yes	No	No

Chi-Squared

Deg. of freedom	4
Statistic	5.1328
P-Value	0.27394
Rank	1

α	0.2	0.1	0.05	0.02	0.01
Critical Value	5.9886	7.7794	9.4877	11.668	13.277
Reject?	No	No	No	No	No

Figure 8: Distribution fitting for frequency of API pairs

$a_k \in K(i)$, then we can assume that a_i's latent vector should be closed to a_k. We fully demonstrate the implicit relationship between APIs by both contextual similarity and prevalence of co-invocation by integrating these two elements into a single measurement of Re_{ik}. Re_{ik} is the implicit relationship of API a_i and a_k and is defined by weighted summation of contextual similarity and co-invocation prevalence as below:

$$Re_{ik} = \theta s_{ik} + (1-\theta)P_{ik} \tag{11}$$

, where θ is the weight for contextual similarity. Hence, we change the regularization term as below:

$$\frac{\gamma}{2}\sum_{i=1}^{x} \|(A_i - \frac{\sum_{k\in K(i)} Re_{ik} \times A_k}{\sum_{k\in K(i)} Re_{ik}})\|_F \tag{12}$$

The implicit relationship allows the regularization term to treat APIs' co-services differently. If API a_i is very similar to a_k, say $Re_{ik} = 0.99$, then a_k should contribute more influences on features of a_i. Reversely, if $Re_{ik} = 0.01$, then a_k should influences less. Hence, we have an objective function of minimizing a confidence-weighted sum of squared errors loss function as in Equation 13.

$$\begin{aligned}\min_{A,M} L = &\sum_{i=1}^{x}\sum_{j=1}^{y}\frac{1}{2}I_{ij}(r_{ij} - \alpha_{ij}{a_i}^T m_j)^2 \\ &+ \frac{\gamma}{2}\sum_{i=1}^{x}\|(A_i - \frac{\sum_{k\in K(i)} Re_{ik}\times A_k}{\sum_{k\in K(i)} Re_{ik}})\|_F \quad + \frac{\lambda_S s_{ij}}{2}\|MA^T\|_F^2 \\ &+ \frac{\lambda_a}{2}\|A\|_F + \frac{\lambda_m}{2}\|M\|_F\end{aligned} \tag{13}$$

The relationship function $Re(i,k)$ allows the contextual and implicit relationship regularization term treat APIs' correlations differently. If APIs a_i and a_k are very compatible and very similar, then $Re(i,k)$ closes to 1. Reversely, if these two APIs are contextually different and have never been simultaneously invoked in any mashup, $Re(i,k)$ should be close to 0.

We use gradient descent in feature vectors A_i and M_j to obtain the local minimum of the objective function given in Equation 13.

$$\begin{aligned}\frac{\partial L}{\partial A_j} = &-\sum_{i=a}^{x} I_{ij}(r_{ij} - \alpha_{ij}{a_i}^T m_j)M_j + s_{ij}\lambda_S MA^T M + \lambda_a A_i \\ &+ \gamma(A_i - \frac{\sum_{k\in K(i)} Re_{ik}\times A_k}{\sum_{k\in K(i)} Re_{ik}}) \\ &+ \gamma \sum_{g\in K^-(i)} \frac{(-Re_{ig})(A_g - \frac{\sum_{k\in K(g)} Re_{gk}\times A_k}{\sum_{k\in K(g)} Re_{gk}})}{\sum_{k\in K(g)} Re_{gk}}\end{aligned} \tag{14}$$

$$\frac{\partial L}{\partial M_j} = -\sum_{j=m}^{y} I_{ij}(r_{ij} - \alpha_{ij}{a_i}^T m_j)A_i + s_{ij}\lambda_S MA^T A + \lambda_m M_j \tag{15}$$

4 EXPERIMENTS

4.1 Set up data

We use the dataset of ProgrammableWeb[1] to demonstrate the effectiveness of the proposed model. The dataset consists of textual descriptions of 17829 APIs and 6340 mashups, and their historical invocation. Because ProgrammableWeb dataset does not have the ratings between mashups and APIs, we adopt their invocation data as the ratings. After that, we do preprocessing for the dataset as follows: (1) We remove blank APIs and mashups, and obtain 5691 mashups and 1170 APIs; (2) We obtain the document vectors of all descriptions by pre-train model Doc2Vec by using library Gensim Doc2Vec and calculate the contextual similarity between descriptions by Cosine equation. (3) We use Python to do programming to define the co-invocation frequency between pairs of APIs. (4) We also calculate the history invocation between pairs of mashup-API; (5) We randomly hide a certain percentage of the mashup-API interactions to create our training set, and the original data becomes the test set.

4.2 Baselines

We compare the proposed model with the following baselines.

(1) SVD++ [5]: SVD based Matrix Factorization Model applies singular value decomposition.
(2) PMF [11]: Probabilistic Matrix Factorization Model purely uses ratings for collaborative filtering.
(3) LDA-PMF: This model uses the LDA [1] instead of Doc2Vec techniques to pre-train the contextual similarity of services and mashup, then attaches them into PMF as a regularization part.
(4) Yao [18]: Matrix Factorization based model with API Co-invocation regularization, which uses TF-IDF for text mining.
(5) Our Model$^-$: Our Attentional PMF model without regularization parts of context-based awareness and API co-invocation relationships.

Table 2 shows the comparison between these baselines with six elements, including ratings, context-awareness, pre-train word embedding, words sequence, attention, and API relationship. It can be seen that the proposed model leverages all these elements to

[1]We download the database from https://dev.maxmind.com/in November 2018.

RIGHTSLINK

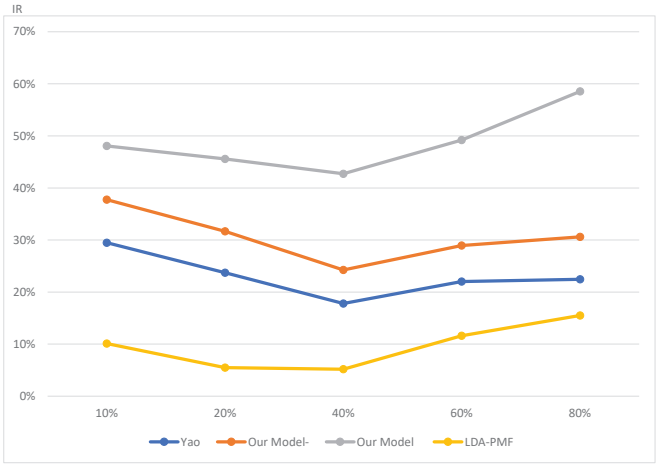


Figure 9: Improvement rate comparison.

Table 1: Improvement rate comparison.

Mask%	10%	20%	40%	60%	80%
Model					
LDA-PMF	10%	5%	5%	12%	16%
Yao	29%	24%	18%	22%	22%
Our Model$^-$	38%	32%	24%	29%	31%
Our Model	48%	46%	43%	49%	59%

obtain a better prediction. We consider our model without context-awareness as Our Model$^-$ to observe how much contextual information and API implicit relationship influences this prediction model. Existing models such as PMF, SVD++ do not integrate contextual information and API implicit relationship. For this paper, we set a number of latent features d = 20. For simplicity, we set regularization parameters $\lambda_a = \lambda_m = 0.01$.

4.3 Evaluation metric

We test the performance of the above baseline models on the ProgrammableWeb datasets. We take the original mashup-API matrix and mask a percentage of original invocations where a mashup-API invocation has been taken place for use as a test set. The test set will contain all of the original invocations, while the training set replaces the specified percentage (10%) of them with a zero in the original invocation matrix. We use the root squared error (RMSE) and mean absolute error (MAE) to evaluate the accuracy of the proposed model as in equation below with n is the number of invocations. Smaller values of RMSE and MAE indicate better performance.

$$RMSE = \sqrt{\frac{1}{n}\Sigma_{i=1}^{x}\Sigma_{j=1}^{y}\left(r_{ij} - \hat{r_{ij}}\right)^2} \tag{16}$$

$$MAE = \frac{1}{n}\sum_{i=1}^{x}\sum_{j=1}^{y}|r_{ij} - \hat{r_{ij}}| \tag{17}$$

4.4 Experimental results

We run the six above models with the 10% masked training dataset. The result of the training set and test set is demonstrated in Table 3. It can be seen that both PMF and SVD++ obtain very good RMSE and MAE in the training set. However, the overfitting makes their results in the test dataset less accurate than the others. LDA-PMF achieves average results in both training set and test set. Yao, our Model$^-$ and our model all achieve very good RMSE and MAE in both datasets, but our model outperforms with RMSE at 0.3805 and MAE at 0.2840.

4.4.1 Impact of different size of training datasets. Also, we further test the effectiveness of our model over various sparseness of the training set. Namely, we respectively use the mask percentage of original invocation matrix as 10%, 20%, 40%, 60%, and 80%. We sketch these results in two charts in Figure 10 for easy to observe. It shows that our model is quite better than Yao and significantly outperforms the other baselines over all ranges of the sparseness of training dataset.

4.4.2 Improvement rate of our model over the baselines. To demonstrate how much Our Model outperforms other baselines, we compare the RMSE improvement rate (IR) of Yao, Our Model$^-$, and Our Model over PMF. Namely, improvement rate is defined as below:

$$IR_{baseline} = \frac{RMSE_{baseline}}{RMSE_{PMF}} \times 100\% \tag{18}$$

It is noticeable that the improvement of Yao, Our Model$^-$, and Our Model over PMF consistently decrease over 10% to 40% and increase over 40% to 80% masked training sets as in Figure 9. LDA-PMF has the least improvement with IR only from 5% to 16%. While Yao model only improves 18% to 29%, Our Model$^-$ outperforms with 24% to 38%, and Our Model significantly outplays with 43% to 59% over PMF model. This result demonstrates that integrating API implicit relationship and contextual information with PMF to optimize the prediction effectively, which ultimately makes Our Model more effective to sparse datasets.

5 CONCLUSION

In this paper, we have enhanced the PMF by learning the importance of mashup-API invocation through their latent features by an MLP attentional network, which significantly improves the representation accuracy. We also studied the influence of contextual information and APIs' relationship on the prediction of recommender systems for Web service. We observe that the pre-training document vectors by Doc2Vec to obtain the contextual similarity of services and doing statistic analysis for APIs' co-invocation contribute effective steps to regularize our Attentional PMF model. Namely, our proposed model obtains better results of RMSE and MAE than the conventional PMF and also outperforms other mentioned baselines. Experimental results in different percentages of sparseness with training datasets also show that our model achieves better RMSE and MAE values.

ACKNOWLEDGMENTS

This work is partially supported by the National Key Research and Development Program of China (No. 2018YFB1402500) and National

RIGHTSLINK

Natural Science Foundation of China under Grant 61832004 and Grant 61672042.

REFERENCES

[1] David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *Journal of machine Learning research* 3, Jan (2003), 993–1022.
[2] Jesús Bobadilla, Fernando Ortega, Antonio Hernando, and Abraham Gutiérrez. 2013. Recommender systems survey. *Knowledge-based systems* 46 (2013), 109–132.
[3] C Buqing, Mingdong Tang, and Xing Huang. 2014. Cscf: A mashup service recommendation approach based on content similarity and collaborative filtering. *International Journal of Grid & Distributed Computing* 7, 2 (2014).
[4] Jingyuan Chen, Hanwang Zhang, Xiangnan He, Liqiang Nie, Wei Liu, and Tat-Seng Chua. 2017. Attentive collaborative filtering: Multimedia recommendation with item-and component-level attention. In *Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval*. ACM, 335–344.
[5] Yehuda Koren. 2010. Factor in the neighbors: Scalable and accurate collaborative filtering. *ACM Transactions on Knowledge Discovery from Data (TKDD)* 4, 1 (2010), 1.
[6] Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *International conference on machine learning*. 1188–1196.
[7] Guang Ling, Michael R Lyu, and Irwin King. 2014. Ratings meet reviews, a combined approach to recommend. In *Proceedings of the 8th ACM Conference on Recommender systems*. ACM, 105–112.
[8] Carmen Martinez-Cruz, Carlos Porcel, Juan Bernabé-Moreno, and Enrique Herrera-Viedma. 2015. A model to represent users trust in recommender systems using ontologies and fuzzy linguistic modeling. *Information Sciences* 311 (2015), 102–118.
[9] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* (2013).
[10] Andriy Mnih and Koray Kavukcuoglu. 2013. Learning word embeddings efficiently with noise-contrastive estimation. In *Advances in neural information processing systems*. 2265–2273.
[11] Andriy Mnih and Ruslan R Salakhutdinov. 2008. Probabilistic matrix factorization. In *Advances in neural information processing systems*. 1257–1264.
[12] Hafida Naïm, Mustapha Aznag, Nicolas Durand, and Mohamed Quafafou. 2016. Semantic pattern mining based web service recommendation. In *International Conference on Service-Oriented Computing*. Springer, 417–432.
[13] Chanyoung Park, Donghyun Kim, Jinoh Oh, and Hwanjo Yu. 2016. Improving top-K recommendation with truster and trustee relationship in user trust network. *Information Sciences* 374 (2016), 100–114.
[14] Md Mahfuzer Rahman, Xiaoqing Liu, and Buqing Cao. 2017. Web API recommendation for mashup development using matrix factorization on integrated content and network-based service clustering. In *2017 IEEE International Conference on Services Computing (SCC)*. IEEE, 225–232.
[15] Horst Rinne. 2008. *The Weibull distribution: a handbook*. Chapman and Hall/CRC.
[16] Jun Xiao, Hao Ye, Xiangnan He, Hanwang Zhang, Fei Wu, and Tat-Seng Chua. 2017. Attentional factorization machines: Learning the weight of feature interactions via attention networks. *arXiv preprint arXiv:1708.04617* (2017).
[17] Lina Yao, Quan Z Sheng, Anne HH Ngu, Jian Yu, and Aviv Segev. 2014. Unified collaborative and content-based web service recommendation. *IEEE Transactions on Services Computing* 8, 3 (2014), 453–466.
[18] Lina Yao, Xianzhi Wang, Quan Z Sheng, Boualem Benatallah, and Chaoran Huang. 2018. Mashup recommendation by regularizing matrix factorization with API co-invocations. *IEEE Transactions on Services Computing* (2018).
[19] Yang Zhong, Yushun Fan, Wei Tan, and Jia Zhang. 2016. Web service recommendation with reconstructed profile from mashup descriptions. *IEEE Transactions on Automation Science and Engineering* 15, 2 (2016), 468–478.

RIGHTSLINK

Table 2: Comparison between six models.

Model	Ratings	Context	pre-train	words sequences	Attention	API relationship
SVD++	Yes	No	No	No	No	No
PMF	Yes	No	No	No	No	No
LDA-PMF	Yes	Yes	Yes	No	No	Yes
Yao	Yes	Yes	Yes	No	No	Yes
Our Model^{-}	Yes	No	No	No	Yes	No
Our Model	Yes	Yes	Yes	Yes	Yes	Yes

Table 3: Experimental results.

No.	Model	SVD++	PMF	LDA-PMF	Yao	Our Model^{-}	Our Model
1	RMSE-Train	0.1999	0.1352	0.4238	0.2104	0.3054	0.2728
2	MAE-Train	0.1729	0.1211	0.4087	0.2011	0.2902	0.2415
3	RMSE-Test	0.7664	0.5634	0.5116	0.4351	0.4090	0.3805
4	MAE-Test	0.7648	0.4554	0.4117	0.3319	0.3140	0.2840

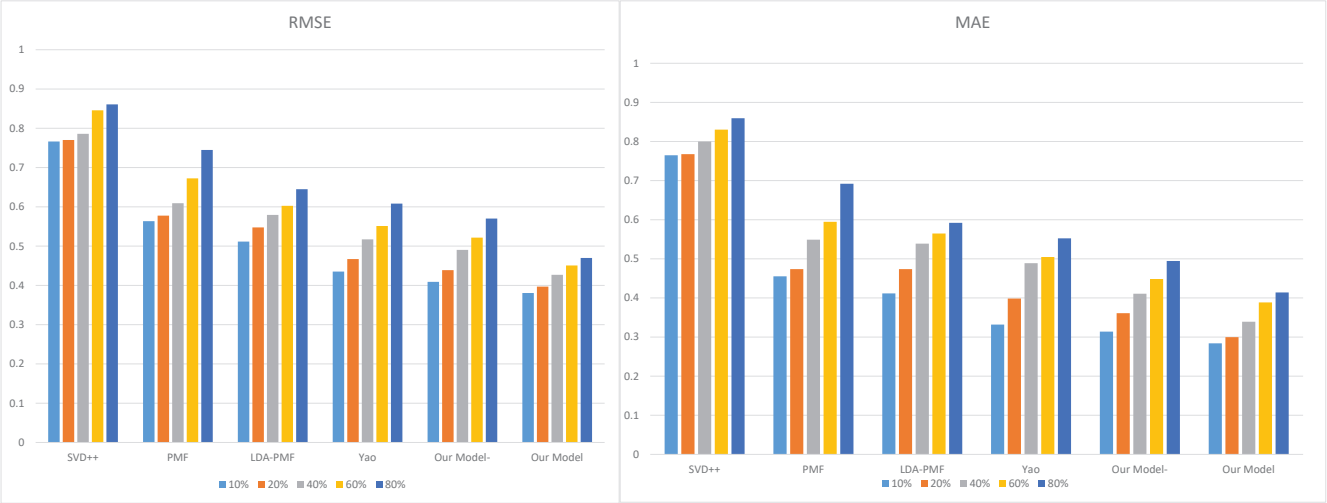


Figure 10: RMSE and MAE over various sparseness with test dataset

RIGHTSLINK