Translation-based Factorization Machines for Sequential Recommendation

Rajiv Pasricha
UC San Diego
rajiv.pasricha@gmail.com

Julian McAuley
UC San Diego
jmcauley@cs.ucsd.edu

ABSTRACT

Sequential recommendation algorithms aim to predict users' future behavior given their historical interactions. A recent line of work has achieved state-of-the-art performance on sequential recommendation tasks by adapting ideas from metric learning and knowledge-graph completion. These algorithms replace inner products with low-dimensional embeddings and distance functions, employing a simple translation dynamic to model user behavior over time.

In this paper, we propose *TransFM*, a model that combines translation and metric-based approaches for sequential recommendation with Factorization Machines (FMs). Doing so allows us to reap the benefits of FMs (in particular, the ability to straightforwardly incorporate content-based features), while enhancing the state-of-the-art performance of translation-based models in sequential settings. Specifically, we learn an embedding and translation space for each feature dimension, replacing the inner product with the squared Euclidean distance to measure the interaction strength between features. Like FMs, we show that the model equation for *TransFM* can be computed in linear time and optimized using classical techniques. As *TransFM* operates on arbitrary feature vectors, additional content information can be easily incorporated without significant changes to the model itself. Empirically, the performance of *TransFM* significantly increases when taking content features into account, outperforming state-of-the-art models on sequential recommendation tasks for a wide variety of datasets.

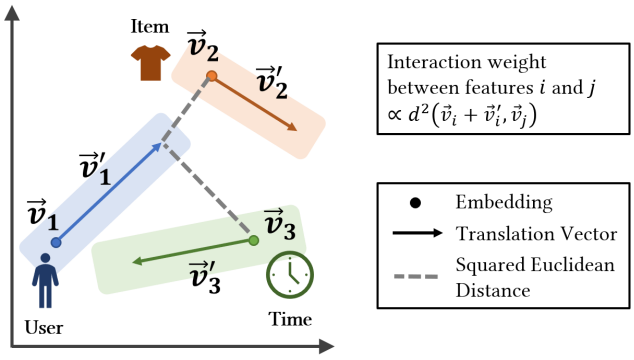


Figure 1: The general-purpose *TransFM* model. Unlike standard sequential and metric-based algorithms, *TransFM* models interactions between all observed features. For each feature i, the model learns two entities: a low-dimensional embedding $\vec{v}_i$ and a translation vector $\vec{v}'_i$. The interaction strength between pairs of features is then measured using the squared Euclidean distance $d^2(\cdot,\cdot)$. In the example above, we plot the embeddings and translation vectors for a user (feature 1), an item e.g. a movie or book (feature 2), and a temporal feature (feature 3). Interaction weights are given by the distance between the ending and starting points of the respective features.

1 INTRODUCTION

From e-commerce sites such as Amazon [18] to online multimedia sites such as Netflix [4] and YouTube [7], recommendation algorithms have become critical to the design and implementation of a successful online platform. Many traditional approaches seek to model 'global' interactions, e.g. by learning low-dimensional user and item embeddings and computing interactions in this space. These algorithms, such as Matrix Factorization [17] and derived models, are able to effectively model user preferences but fail to account for sequential dynamics, providing a static list of recommendations regardless of a user's sequence of recent interactions.

Sequential recommender systems add an additional dynamic: taking the order of previous interactions into account. Successfully modeling these *third order* interactions (between a user, an item under consideration, and the previous item consumed) facilitates a more engaging user experience, resulting in recommendations that are more responsive to recent user and item dynamics [27, 28]. A recent approach to sequential recommendation is *TransRec* [13], which operates by learning a latent item embedding space within which users are modeled as linear translation vectors. *TransRec* operates in a metric space, replacing inner products with distance functions ($d(\cdot,\cdot)$). This follows a line of work that adapts ideas from metric learning [23] and knowledge-graph completion [30, 38] into recommender systems, which has led to state-of-the-art performance on a variety of tasks.

A natural avenue to extending this recent work is to adapt such metric and translation-based methods to incorporate content features. A few specific approaches have been proposed, most notably from the domain of music recommendation. For example, [33] incorporates audio features using a specialized Convolutional Neural Network, and [3] proposes a variational Bayes technique for playlist generation using both collaborative and content features. However, offering a general-purpose technique to incorporate content features into metric-based approaches remains open.

RecSys '18, October 2–7, 2018, Vancouver, BC, Canada

ACM ISBN 978-1-4503-5901-6/18/10…$15.00
https://doi.org/10.1145/3240323.3240356

Factorization Machines achieve this goal in inner-product spaces, incorporating additional features without sacrificing model simplicity [25]. FMs operate on arbitrary real-valued feature vectors, and model higher-order interactions between pairs of features via factorized parameters. They can be applied to general prediction tasks and are able to replicate a variety of common recommender system models, such as matrix factorization and FPMC [27], simply by selecting appropriate feature representations.

In this paper, we propose *TransFM*, which adapts ideas from FMs into translation-based sequential recommenders. Doing so allows us to straightforwardly model complex interactions between features (as in FMs), while extending the state-of-the-art performance of metric/translation-based approaches.[1]

Specifically, we replace the inner product in the FM interaction term with a translation component between feature embeddings, employing the squared Euclidean distance to compare compatibility between pairs of feature dimensions (see Figure 1). As with Factorization Machines, we show that the *TransFM* model equation can be computed in linear time in both the feature and parameter dimensions, making it efficient to implement for large-scale sequential recommendation datasets.

The translation component of the model effectively learns relationships among collaborative and content-based features with minimal preprocessing and feature engineering. Quantitatively, we evaluate *TransFM* on datasets from Amazon [22], Google Local [13], and MovieLens [12], and find that *TransFM* with content features provides significant improvements over state-of-the-art baselines with and without additional features included.

We present a generalization of this approach and derive related models by merging FMs with similar baseline models. This leads to general-purpose recommendation approaches that incorporate the intuitions of other baseline approaches, consistently outperforming vanilla Factorization Machines.

2 RELATED WORK

2.1 Sequential Recommendation

Many sequential recommendation algorithms adapt Markov Chains to model sequential dynamics. Factorized Personalized Markov Chains (FPMC) factorizes a third-order transition 'cube' to predict a user's next basket of purchases, using independent factorization matrices to model pairwise interactions [27]. [9] introduces Personalized Ranking Metric Embedding (PRME), which replaces inner products with Euclidean distances to model user-item interactions.

TransRec [13] is also a sequential recommendation approach, modeling users as translation vectors through a shared item embedding space. This gives the following probability of observing a next item j given user u and previous item i:

$$P(j|u,i) \propto \beta_j - d(\vec{\gamma}_i + \vec{T}_u, \vec{\gamma}_j). \tag{1}$$

These models perform well given historical user sequences, but cannot take temporal, geographical, or other content features into account without significant changes to the model forms. Sequential recommenders that do take features into account (e.g. [19, 21]) involve specialized models derived for specific tasks and datasets.

[1]Source code: https://github.com/rpasricha/TransFM

2.2 Factorization Machines

Factorization Machines [25] are a general-purpose predictive framework for arbitrary machine learning tasks. They model all second-order interactions between features and can naturally be extended to handle arbitrary higher-order interactions. Each feature interaction is weighted according to the inner product between factorized parameters, resulting in the following model equation:

$$\hat{y}(\vec{x}) = w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} \langle \vec{v}_i \vec{v}_j \rangle x_i x_j. \tag{2}$$

FMs can be applied to arbitrary regression, classification, or ranking tasks by selecting an appropriate loss function. In this work, we focus on the implicit feedback setting, applying the Bayesian Personalized Ranking (BPR) framework to optimize the ranking of predicted items [26].

Given their simplicity and applicability to a variety of machine learning tasks, FMs have been extended in a variety of ways since their introduction. [37] incorporates features of skip-gram and other text mining algorithms to apply FMs to sentiment classification, and [29] includes FMs in a multi-stage predictive model to extract relevant reviews for recommendation. Finally, [29] propose domain-specific models applying FMs to content modeling on Twitter and CTR prediction in advertising.

2.3 Hybrid Recommendation

Hybrid recommendation algorithms merge aspects of collaborative filtering and content-based approaches, aiming to improve performance and make useful recommendations to users and items with few observed interactions. Potential additional sources of information include temporal [10, 16], social [5, 11], and geographical [24, 34] features. Recent hybrid approaches have incorporated image features to improve content or next POI recommendation [36], and applied deep learning techniques to automatically generate useful content features [31] or introduce additional modeling flexibility [14]. While these approaches achieve state-of-the-art performance compared to relevant baselines, they all rely on specialized models and techniques to incorporate additional features. In contrast, we present a generalized approach, which operates on arbitrary feature vectors and prediction tasks. With appropriate feature engineering, our model can incorporate temporal, geographical, demographic, and other content features without changing the model form itself.

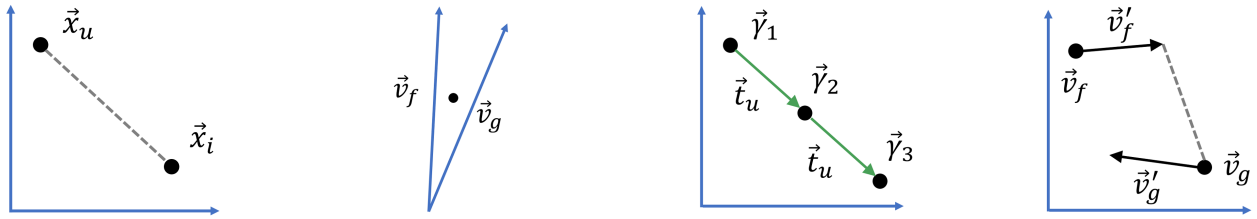
3 THE TRANSFM MODEL

3.1 Problem Formulation

TransFM combines the distance and translation components of the TransRec model with the ability of FMs to incorporate arbitrary real-valued features for the purpose of sequential recommendation.

Table 1 includes notation used throughout the paper. As with Factorization Machines, *TransFM* operates on real-valued feature vectors $\vec{x}$. In the sequential recommendation setting, $\vec{x}$ includes feature representations for the user u, the previous item i, and next item j, along with any additional content features.

Each dimension in $\vec{x}$ is associated with both an embedding and a translation vector. Formally, for feature x_i, we learn two vectors: an embedding vector $\vec{v}_i \in \mathbb{R}^k$ and a translation vector $\vec{v}'_i \in \mathbb{R}^k$. We



(a) PRME **(b) Factorization Machines** **(c) TransRec** **(d) *TransFM***

Figure 2: A visual comparison of translation, metric, and factorization-based recommender system models.

Table 1: Notation

Notation	Explanation
$\mathcal{U}, \mathcal{I}$	User set, item set
$\mathcal{S}^u$	Historical interaction sequence for user u
$\vec{x}_{u,i,j}$	Feature vector for user u, previous item i, and next item j
k	Dimensionality of embedding and translation spaces
n	Dimensionality of $\vec{x}_{u,i,j}$
w_0	Global bias term
$\vec{w}$	Linear terms; $\vec{w} \in \mathbb{R}^n$
$\mathbf{V}$	Feature embedding space; $\mathbf{V} \in \mathbb{R}^{n \times k}$
$\mathbf{V}'$	Feature translation space; $\mathbf{V}' \in \mathbb{R}^{n \times k}$
$d^2(\vec{a}, \vec{b})$	Squared Euclidean distance between $\vec{a}$ and $\vec{b}$

apply the translation operation to the previous item embedding and measure the distance to the next item embedding $\vec{v}_j$ by the squared Euclidean distance. The resulting distance gives the weight assigned to the corresponding feature interaction.

The model equation of *TransFM* is given by:

$$\hat{y}(\vec{x}) = w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} d^2(\vec{v}_i + \vec{v}'_i, \vec{v}_j) x_i x_j, \qquad (3)$$

where w_0 is a global bias term and w_i is the linear term for feature x_i. $\vec{v}_i$ and $\vec{v}'_i$ are the embedding and translation vectors (resp.) for feature x_i, and $d^2(\vec{a}, \vec{b})$ represents the squared Euclidean distance between the vectors $\vec{a}$ and $\vec{b}$:

$$d^2(\vec{a}, \vec{b}) = (\vec{a} - \vec{b}) \cdot (\vec{a} - \vec{b}) = \sum_{f=1}^{k} (a_f - b_f)^2.$$

Like other metric-based models, *TransFM* replaces the inner product term with the (squared) Euclidean distance. This leads to improved generalization performance, more effectively capturing the transitive property between feature embeddings. For example, if the feature pairs (a, b) and (b, c) exhibit high interaction weights, then features a and c will be closely related as well, even if there are few or no observed interactions between them.

Figure 2 provides a comparison of the prediction methods used by *TransFM* and various baseline models. PRME (2a) learns a personalized metric space in which the distance between embeddings measures user-item compatibility (the corresponding item-item sequential space is not shown); Factorization Machines (2b) measure interactions between arbitrary features with the inner product between corresponding factorized parameters; TransRec (2c) learns an embedding $\vec{\gamma}_i$ for each item, and a translation vector $\vec{t}_u$ for each user traversing their interaction sequence. Finally *TransFM* (2d) learns an embedding $\vec{v}_i$ and translation vector $\vec{v}'_i$ for each feature, using the squared Euclidean distance to measure feature interactions.

3.2 Computation

The model equation for Factorization Machines can be computed in linear time $O(kn)$, where k is the dimensionality of the model parameter vectors and n is the dimensionality of the input feature vectors [25]. In this section, we show that the same result applies to *TransFM*.

In order to simplify the squared Euclidean distance d^2, we take advantage of the ability to write d^2 in terms of inner products:

$$d^2(\vec{v}_i + \vec{v}'_i, \vec{v}_j) = (\vec{v}_i + \vec{v}'_i - \vec{v}_j) \cdot (\vec{v}_i + \vec{v}'_i - \vec{v}_j).$$

This allows us to rewrite the interaction term as follows:

$$\begin{aligned}
&\sum_{i=1}^{n} \sum_{j=i+1}^{n} d^2(\vec{v}_i + \vec{v}'_i, \vec{v}_j) x_i x_j \\
&= \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} d^2(\vec{v}_i + \vec{v}'_i, \vec{v}_j) x_i x_j - \frac{1}{2} \sum_{i=1}^{n} d^2(\vec{v}_i + \vec{v}'_i, \vec{v}_i) x_i x_i \\
&= \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \left((\vec{v}_i + \vec{v}'_i - \vec{v}_j) \cdot (\vec{v}_i + \vec{v}'_i - \vec{v}_j) x_i x_j \right) - \frac{1}{2} \sum_{i=1}^{n} (\vec{v}'_i \cdot \vec{v}'_i) x_i x_i \\
&= \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} (x_i x_j)(\vec{v}_i \cdot \vec{v}_i + \vec{v}'_i \cdot \vec{v}'_i + \vec{v}_j \cdot \vec{v}_j + 2\vec{v}_i \cdot \vec{v}'_i - 2\vec{v}_i \cdot \vec{v}_j - 2\vec{v}'_i \cdot \vec{v}_j) \\
&\qquad - \frac{1}{2} \sum_{i=1}^{n} (\vec{v}'_i \cdot \vec{v}'_i) x_i x_i .
\end{aligned}$$

The first sum above can be split into six individual sums, each of which multiplies the feature product $x_i x_j$ with one of the corresponding inner products. We present a simplified version of one of the six sums below:

$$\frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} (\vec{v}_i \cdot \vec{v}_i) x_i x_j = \frac{1}{2} \left(\sum_{i=1}^{n} (\vec{v}_i \cdot \vec{v}_i) x_i \right) \left(\sum_{j=1}^{n} x_j \right).$$

(others are similar and omitted for brevity).

Thus we see that all terms in Equation 3 can be computed with at most two sums over the input features, and at most one inner product between corresponding parameter vectors. Given input features of dimensionality n and parameters of dimensionality k,

this shows that the *TransFM* model can be computed in linear complexity in both k and n, or $O(kn)$.

As with FMs, the above feature vectors are sparse (e.g. one-hot user/item encodings), and the above sums need to be computed only over the nonzero elements of the input feature vectors. We denote by $\vec{x}_{u,i,j}$ the feature vector consisting of one-hot encodings for the (u, i, j) user, previous item, next item triplet.

3.3 Optimization

We consider the sequential recommendation setting with implicit feedback, i.e. rather than optimizing the precise output value of our model equation, we instead aim to rank the observed next item j ahead of all other items $j' \in \mathcal{I} \setminus j$ in the dataset. To this end, we adopt the Sequential Bayesian Personalized Ranking (S-BPR) optimization criterion [27].

Applying S-BPR, we optimize the total order $>_{u,i}$ given a user u and previous item i:

$$\begin{aligned}\hat{\Theta} &= \arg\max_{\Theta} \ln \prod_{u \in \mathcal{U}} \prod_{j \in \mathcal{S}^u} \prod_{j' \notin \mathcal{S}^u} \Pr(j >_{u,i} j' | \Theta) \Pr(\Theta) \\ &= \arg\max_{\Theta} \sum_{u \in \mathcal{U}} \sum_{j \in \mathcal{S}^u} \sum_{j' \notin \mathcal{S}^u} \ln \sigma(\hat{y}(\vec{x}_{u,i,j}) - \hat{y}(\vec{x}_{u,i,j'})) - \Omega(\Theta),\end{aligned}$$

where i is the item immediately preceding j in the consumption sequence. Accordingly, we also restrict j from being the first item in the sequence as it has no associated previous item. $\hat{y}(\vec{x})$ is the *TransFM* model described in Equation 3, Θ is the set of parameters $\{w_0, \vec{w}, \mathbf{V}, \mathbf{V'}\}$ to be learned by the model and $\Omega(\Theta)$ is a standard $\mathcal{L}_2$ regularization term.

3.4 Implementation and Inference

We implement the *TransFM* model in TensorFlow [1] and use mini-batch gradient descent with Adam Optimization to train our models [15]. Adam is effective for learning models with many parameters on sparse datasets and was the most effective optimization algorithm in our experiments.

We apply the standard BPR optimization process, based on stochastic gradient descent with bootstrap sampling [26]. For every positive triple (u, i, j), we randomly sample a negative item $j' \in \mathcal{I}$ on every iteration to add to our mini batch. This set of positive and negative triples is then used to update the parameters of the model.

All model parameters are randomly initialized within the interval $[-0.1, 0.1]$, and regularization parameter values are optimized using a grid search over the values $\{0.0, 0.001, 0.01, 0.1, 1.0, 10.0, 100.0\}$. We iterate until convergence, as measured by performance on a held-out validation set.

4 EXPERIMENTS

4.1 Datasets and Statistics

In order to evaluate the quantitative performance of the proposed *TransFM* model, we perform experiments using a variety of publicly available datasets that vary significantly in terms of size and sparsity. As we are concerned with learning models from implicit feedback, we first convert all observed ratings to be positive feedback, discarding the observed star ratings if present. We then remove all users and items with fewer than five observed interactions. Statistics of the datasets under consideration are included in Table 2.

Table 2: Dataset Statistics (after preprocessing)

Dataset	#users ($\lvert\mathcal{U}\rvert$)	#items ($\lvert\mathcal{I}\rvert$)	#actions	avg. #actions / user	avg. #actions / item
Office	16,716	22,357	128,070	7.66	5.73
Automotive	34,316	40,287	138,573	5.35	4.56
Video Games	31,013	23,715	287,107	9.26	12.11
Toys and Games	57,617	69,147	410,920	7.13	5.94
Cell Phones	68,330	60,083	429,231	6.28	7.14
North Carolina	4,573	7,846	31,167	6.82	3.97
Colorado	4,586	7,989	34,880	7.61	4.37
Washington	4,453	7,196	39,316	8.83	5.46
Florida	12,096	21,388	77,145	6.38	3.61
Texas	16,066	24,729	136,930	8.52	5.54
California	23,644	35,252	237,051	10.03	6.72
MovieLens	943	1,349	99,287	105.29	73.60
Total	274k	321k	2.05M	-	-

Amazon[2]: This dataset, originally introduced by [22], contains a large corpus of product ratings, reviews, and metadata, collected from Amazon.com from May 1996 to July 2014. The full dataset consists of 83 million ratings and reviews collected during this period, along with additional features including item metadata and visual features. Notable for its high sparsity, the Amazon dataset provides a useful benchmark to evaluate recommender system algorithms on sparse input data. The additional available metadata also makes it an appealing choice to evaluate algorithms combining collaborative filtering techniques with additional sources of information. We use purchases from five top-level categories covering a variety of distinct purchase domains.

Google Local[3]: This dataset contains a large collection of business ratings and reviews and was originally introduced by [13]. The dataset also includes many associated content features, including user demographics and business locations. The availability of GPS coordinates facilitates evaluating *TransFM* in a geographical recommendation setting. In this work, we evaluate datasets containing businesses from six U.S. states of varying sizes and populations.

MovieLens[4]: The MovieLens dataset has been used for many years to evaluate a large variety of recommendation algorithms [12]. Created by the GroupLens research group at the University of Minnesota, MovieLens allows its users to submit ratings and reviews for movies they have watched and recommends movies that those users may enjoy. From its inception, MovieLens and its associated datasets have been vital to the development of improved recommendation algorithms as well as related studies in psychology and other domains [2, 6, 20, 39]. In this work, we use the MovieLens-100k benchmark dataset. Compared to the Amazon datasets, this dataset exhibits a much higher degree of user and item density.

[2] https://jmcauley.ucsd.edu/data/amazon/
[3] http://jmcauley.ucsd.edu/data/googlelocal/
[4] https://grouplens.org/datasets/movielens/

4.2 Features

TransFM is intended to be a 'feature-agnostic' general-purpose model that can yield significant performance improvements when incorporating additional features, with no other changes to the model format. Thus our focus is not on complex feature design techniques, but rather to show that significant performance improvements can be achieved with minimal feature preprocessing. To that end, we extract the following content-based features from each dataset to evaluate our model:

Temporal Features: Temporal data has been widely used to improve recommendation performance [8, 16, 32]. Each of our datasets contain temporal information, specifically the time $t_{u,i}$ for each rating between user u and item i.

For the implicit feedback recommendation task with a ranking loss, each training example consists of a triplet (u, i, j) of a user, previous item, and next item. As a result, we add two additional features to $\vec{x}_{u,i,j}$: the time $t_{u,i}$ of user u's rating of previous item i, and the time $t_{u,j}$ of u's rating of next item j. All timestamps for each dataset are first normalized to have zero mean and unit variance.

During training, corresponding positive and negative instances are both associated with the same timestamp, so that we are optimizing a time-specific ranking loss.

Item Category Features: The Amazon datasets also provide a list of categories for each item. These categories form a hierarchical list of labels, which are useful as item features to improve performance and generalizability, especially in sparse settings. We convert the observed category labels into binary indicator vectors for previous and next items, and add them to the feature vector $\vec{x}_{u,i,j}$.

User and Item Content Features: MovieLens provides a variety of content features for both users and items. We use the following features in our models: *user age, user gender, user occupation, user zip code,* and *movie genre.* Movie genre, user occupation, and user zip code are encoded into binary features. We convert the user gender to a single binary feature and leave the user age unchanged.

Geographical Features: As the Google Local datasets capture ratings for various businesses, each 'item' is associated with corresponding latitude and longitude coordinates. We add these coordinates to *TransFM* to evaluate the model in a geographical setting. For each state, we first round the coordinates to a single decimal place, and then create binary feature vectors with one feature for each bin. For example, from the Google Washington dataset, we observe 38 and 78 latitude and longitude features respectively.

4.3 Evaluation Methodology

In the sequential recommendation setting, the prediction for each item depends on the previous items in the user's consumption sequence. As a result, we first partition the consumption sequence for user u into three sub-sequences. The most recent item $\mathcal{S}^u_{|\mathcal{S}^u|}$ is added to the test set, the previous item $\mathcal{S}^u_{|\mathcal{S}^u|-1}$ to the validation set, and the remaining $|\mathcal{S}^u| - 2$ items are kept in the training set.

We report the performance of each model according to the **Area Under the ROC Curve**, or AUC, defined below.

$$\text{AUC} = \frac{1}{|\mathcal{U}|} \sum_{u \in \mathcal{U}} \frac{1}{|\mathcal{I} \setminus \mathcal{S}^u|} \sum_{j' \in \mathcal{I} \setminus \mathcal{S}^u} \mathbb{1}(R_{u,g_u} < R_{u,j'}),$$

where g_u is the ground truth item for user u in the test set, and $R_{u,i}$ is the rank of item i for user u in the output list of recommendations. Finally, $\mathbb{1}(\cdot)$ is the indicator function that returns 1 if the ground truth item is ranked ahead of the unobserved item j'.

4.4 Models

We compare *TransFM* against the following baselines:

PopRec: This is a naive popularity baseline that ranks items in order of their overall popularity in the dataset. It is not personalized, so it provides the same list of recommendations to all users.

BPR-MF: This model uses the Bayesian Personalized Ranking (BPR) framework with Matrix Factorization (MF) as the underlying model [26]. It learns global personalized user-item dynamics but does not take sequential signals into account.

Factorized Markov Chain (FMC): This is a non-personalized sequential model that factorizes the global item-to-item transition matrix. It does not take personalized user interactions into account.

Factorized Personalized Markov Chain (FPMC): A combination of the MF and FMC models, FPMC factorizes the three dimensional sequential interaction tensor [27]. Predictions are computed by taking inner products between factorized parameter vectors:

$$P(j|u,i) \propto \langle \vec{v}^{U,J}_u, \vec{v}^{J,U}_j \rangle + \langle \vec{v}^{I,J}_i, \vec{v}^{J,I}_j \rangle, \tag{4}$$

where $V^{U,J}$, $V^{J,U}$, $V^{I,J}$, and $V^{J,I}$ are the four embedding spaces learned by the model.

Personalized Ranking Metric Embedding (PRME): This model replaces the inner products in FPMC with Euclidean distances, embedding users and items into two latent spaces to model personalized and sequential dynamics respectively [9]. The hyperparameter α modulates the relative importance between these two spaces:

$$P(j|u,i) \propto -(\alpha \cdot d(\vec{v}_u, \vec{v}_j) + (1-\alpha) \cdot d(\vec{w}_i, \vec{w}_j)). \tag{5}$$

Hierarchical Representation Model (HRM): This model introduces an aggregation component to FPMC to allow more flexibility in modeling interactions between users and items [35]:

$$P(j|u,i) \propto \langle f(\vec{v}_u, \vec{v}_i), \vec{v}_j \rangle. \tag{6}$$

We test average and max pooling for the aggregation function f.

TransRec: This is the model proposed in [13], which embeds each item in a shared embedding space and learns personalized translation vectors through this space for each user (see Equation 1). This allows the TransRec model to achieve state-of-the-art performance, excelling on datasets with the highest levels of sparsity.

CatCos: This is a naive extension to TransRec that incorporates content features. We follow the intuition that items with similar content features should have similar embeddings. This is enforced by adding a regularization term that computes the distance between consecutive pairs of item embeddings $\vec{\gamma}_i$ and $\vec{\gamma}_j$, weighted by the cosine similarity of their corresponding content vectors. Formally, we add the following regularization term:

$$\mathcal{R} = \sum_{u \in \mathcal{U}} \sum_{j \in \mathcal{S}^u} s(\vec{x}^c_{u,i}, \vec{x}^c_{u,j})(\vec{\gamma}_i - \vec{\gamma}_j)^2, \tag{7}$$

Table 3: Results of our baseline and proposed models on different datasets, with respect to the AUC (higher is better). The best performing baseline and proposed models for each dataset are bolded. The final row shows the percent improvement of *TransFM*$_{\text{content}}$ over the best baseline.

		Amazon						Google Local							
Model	content aware?	Office Products	Auto.	Video Games	Toys and Games	Cell Phones	Amazon Average	N.C.	Colo.	Wash.	Fla.	Tex.	Calif.	Google Average	MovieLens
PopRec		0.6427	0.5870	0.7497	0.6240	0.6959	0.6599	0.4888	0.5085	0.5123	0.4722	0.5612	0.5785	0.5203	0.7413
BPR-MF		0.6979	0.6307	0.8551	0.7289	0.7611	0.7347	0.7096	0.6826	0.6994	0.7275	0.7657	0.7969	0.7303	0.8602
FMC		0.6865	0.6442	0.8423	0.6948	0.7548	0.7245	0.6542	0.6164	0.6491	0.6432	0.7153	0.7284	0.6678	0.8515
FPMC		0.6859	0.6415	0.8523	0.7198	0.7376	0.7274	0.6698	0.6463	0.6662	0.6619	0.7239	0.7462	0.6857	0.8858
PRME		0.7006	0.6473	0.8601	0.7264	0.7887	0.7446	0.7064	0.6602	0.6837	0.7107	0.7532	0.7750	0.7149	0.8851
HRM_{avg}		0.6985	0.6703	0.8779	0.7581	0.7891	0.7588	**0.7691**	0.7219	**0.7440**	0.7812	**0.8207**	**0.8346**	**0.7786**	0.8856
HRM_{max}		0.6983	0.6560	0.8566	0.7263	0.7656	0.7406	0.7067	0.6666	0.6941	0.7109	0.7506	0.7692	0.7164	0.8844
TransRec		0.7383	0.6953	0.8885	0.7643	0.8080	0.7789	0.7507	0.7161	0.7313	0.7685	0.8030	0.8215	0.7652	**0.8873**
CatCos	✓	0.7402	0.7048	0.8878	**0.7762**	0.8099	0.7838	0.7524	0.7177	0.7352	0.7639	0.8021	0.8221	0.7656	0.8678
FM		0.7075	0.6572	0.8523	0.6994	0.7558	0.7344	0.6787	0.6504	0.6812	0.7057	0.7435	0.7732	0.7055	0.8575
FM_{time}	✓	0.7426	0.6671	0.8866	0.7488	**0.8153**	0.7721	0.6554	0.6392	0.6761	0.6757	0.7251	0.7608	0.6887	0.8617
$\text{FM}_{\text{content}}$	✓	**0.7586**	**0.7328**	**0.8912**	0.7761	0.7611	**0.7840**	0.7673	**0.7345**	0.7352	**0.7821**	0.8025	0.8107	0.7721	0.8660
TransFM		0.7169	0.6675	0.8584	0.7203	0.7767	0.7480	0.6454	0.6327	0.6498	0.6507	0.7072	0.7341	0.6700	0.8611
$\text{TransFM}_{\text{time}}$	✓	0.7430	0.6776	0.8778	0.7583	0.8209	0.7755	0.6257	0.6203	0.6289	0.6233	0.6857	0.7157	0.6499	0.8722
$\text{TransFM}_{\text{content}}$	✓	**0.8463**	**0.8319**	**0.9587**	**0.8673**	**0.8406**	**0.8690**	**0.7947**	**0.7535**	**0.7586**	**0.8095**	**0.8371**	**0.8379**	**0.7986**	**0.9381**
Improvement vs. best baseline		11.6%	13.5%	7.6%	11.7%	3.1%	10.8%	3.3%	2.6%	2.0%	3.5%	2.0%	0.4%	2.6%	5.7%

where i is the item in $\mathcal{S}^u$ immediately preceding j, $\vec{x}^c_{u,i}$ is the content feature vector for user u and item i, and $s(\cdot,\cdot)$ is the cosine similarity function:

$$s(\vec{u},\vec{v}) = \frac{\vec{u}\cdot\vec{v}}{\|\vec{u}\|\|\vec{v}\|}.$$

For MovieLens, we only take the movie genre into account and do not add user features. As we compare sequential item embeddings in a single consumption sequence, the user features of the previous and next item will always be identical.

FM: This is the standard Factorization Machine model, which models interactions between all pairs of features by using an inner product between corresponding parameter vectors:

$$\hat{y}(\vec{x}) = w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n}\sum_{j=i+1}^{n} \langle \vec{v}_i, \vec{v}_j \rangle x_i x_j.$$

We evaluate the FM model in three cases: (1) without additional features, (2) with temporal features, and (3) with category / content features. These are represented in the results as 'FM,' 'FM_{time},' and '$\text{FM}_{\text{content}}$' respectively.

TransFM: This is the *TransFM* model proposed in this paper. This model replaces the inner product of Factorization Machines with a translation operation followed by the computation of the squared Euclidean distance:

$$\hat{y}(\vec{x}) = w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n}\sum_{j=i+1}^{n} d^2(\vec{v}_i + \vec{v}'_i, \vec{v}_j) x_i x_j.$$

As with FMs, we evaluate *TransFM* in three cases: (1) without additional features, (2) with temporal features, and (3) with content features. These are represented in the results as '*TransFM*,' '$\textit{TransFM}_{\text{time}}$,' and '$\textit{TransFM}_{\text{content}}$' respectively.

The goal of our baselines is to compare (1) standard recommendation baselines, (2) specialized models for sequential recommendation (TransRec), (3) incorporating content features by adding additional constraints (CatCos), (4) general-purpose models with inner product interaction terms (FM), and (5) general-purpose metric/translation-based interaction models (*TransFM*). We also evaluate the relative performance improvements attained by adding temporal, category, or content features.

4.5 Performance and Quantitative Analysis

Results from our experiments are collected in Table 3. The number of factor dimensions k for all models is set to 10; we analyze the impact of changing the dimensionality later. For all datasets, the best performing model is *TransFM* with content features. The final row of Table 3 shows the improvement of *TransFM* over the best performing baseline. Our main findings are summarized below:

Baseline Models: As expected, all models outperform the simple popularity-based baseline. BPR-MF and FMC respectively model personalized and sequential components, and their respective performance shows that both components play a significant role in making successful recommendations. By adding a personalization component, FPMC outperforms FMC for all datasets and is among the best baselines for the (dense) MovieLens dataset. However, it loses to BPR-MF for most datasets, suggesting that learning multiple independent embeddings is not well-suited to sparse domains.

By replacing inner products with metric distances, PRME outperforms FPMC for all Amazon and Google datasets. HRM_{avg} outperforms PRME in most cases, demonstrating the effectiveness of an appropriate aggregation term. This contrasts with [35], in which the nonlinear max pooling operation performed best. We expect that

the increased sparsity of our data inhibits the ability of HRM_{max} to uncover appropriate nonlinear dynamics.

TransRec is the best performing content-agnostic method for Amazon and MovieLens but loses to HRM_{avg} for all Google Local datasets. This suggests that the translation vector intuition of TransRec does not effectively model interactions in Google Local as well as the simpler HRM model.

CatCos: The CatCos baseline model outperforms the non-content aware baselines for Amazon but loses to solely collaborative approaches for the other datasets, despite the addition of useful content features. This indicates that more specialized models or feature representations would be necessary to fully incorporate content information into the TransRec framework.

FMs: The standard FM model performs worse than more specialized sequential baseline models for all datasets. As opposed to many other baseline approaches, FMs do not explicitly model personalized sequential dynamics, and use inner products to model arbitrary feature interactions. Compared to the metric-based approach, these inner products are less effectively able to extract useful dynamics from extremely sparse datasets.

FMs with features: Factorization Machines are effectively able to incorporate content features and achieve significant performance benefits, without requiring any changes to the model format itself. Adding temporal data to FMs leads to significant performance improvements for Amazon and MovieLens, despite only adding two additional features to $\vec{x}_{u,i,j}$.

This highlights the importance of effectively modeling temporal data to improve recommendation performance and shows that strong temporal effects are present in these datasets. However, adding temporal features causes performance for Google Local to decline, as temporal dynamics do not play as significant a role in modeling review sequences for local businesses.

Adding content features also results in substantial improvements, especially for datasets with the highest sparsity. $\text{FM}_{\text{content}}$ outperformed FM_{time} in most cases, demonstrating the importance of content features to compensate for insufficient interaction data.

TransFM: Although *TransFM* (without features) does not outperform all baseline models, it does exceed standard FMs for the Amazon and MovieLens datasets. However, FMs perform better for the Google Local datasets, suggesting that without any additional features, inner products more effectively model interactions in this setting. This matches our observations of TransRec, which is outperformed by the inner product-based HRM_{avg} baseline.

TransFM with features: Adding temporal data to *TransFM* has a similar effect as the corresponding FM_{time} baseline. When temporal features play a significant role in the datasets (e.g. for Amazon and MovieLens), *TransFM* is able to extract these dynamics.

The $\textit{TransFM}_{\text{content}}$ approach achieves the highest AUC for all datasets. The translation technique is effective at modeling both content and collaborative feature interactions, resulting in more significant improvements over vanilla *TransFM* than the corresponding $\text{FM}_{\text{content}}$ approach. These improvements hold for all datasets: Amazon (with category features), Google Local (with geographical features), and MovieLens (with user/item content features). Despite the increased density of MovieLens, *TransFM* is still able to extract additional value from user and item content features to improve recommendation performance.

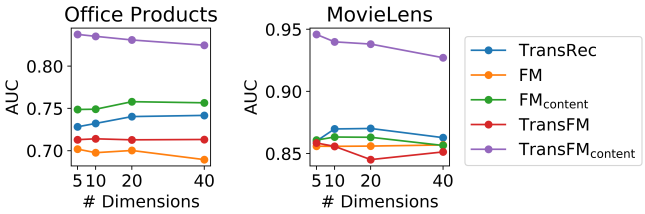


Figure 3: AUC of *TransFM* and various baselines with respect to increasing dimensionality k.

For the Google Local dataset, geographical features play a more significant role in user-item interactions than temporal data. The performance of $\textit{TransFM}_{\text{content}}$ on this dataset indicates the translation component is effectively able to model interactions between arbitrary user, item, and geographical features.

4.6 Sensitivity to Dimensionality

To analyze the sensitivity of *TransFM* to the parameter dimensionality, We adjust k in the set $k \in \{5, 10, 20, 40\}$ and plot the resulting AUC values for Office Products and MovieLens datasets in Figure 3 (other datasets exhibited similar performance and are withheld for brevity). We observe that in most cases, performance does not increase significantly with dimensionality. However, $\textit{TransFM}_{\text{content}}$ significantly outperforms all other models for all values of k.

4.7 Sign of the Interaction Term

Like FMs, *TransFM* adds the interaction term in the prediction equation (see Equation 3). This assigns features that are farther apart a higher interaction strength. In order to more closely match the intuition of standard metric-based models, where smaller distances correspond to higher interaction weights, we also tested a variant of *TransFM* with a negative interaction term. The resulting model displayed similar performance with no additional features or with temporal data but had significantly reduced performance with content features. This suggests that an additive distance term increases the model's flexibility to appropriately model interactions between users, items, and content features, with the $\mathcal{L}_2$ regularization term constraining the feasible set of embedding locations.

5 FMS APPLIED TO RELATED RECOMMENDATION APPROACHES

TransFM is a general-purpose model which adds elements from translation and metric-based sequential algorithms to the FM framework. TransRec, a similar translation-based model, was applied in [13] to the sequential recommendation task but lacked the ability to be natively extended with content features. In this section, we present related extensions of FMs that draw inspiration from similar baseline algorithms to achieve improved performance while retaining compatibility with arbitrary feature vectors.

We apply a similar approach to two baseline models: PRME and HRM, specialized sequential recommendation models similar to TransRec. PRME models sequential recommendations with

embeddings in a metric space, using single embedding locations rather than translation vectors. HRM, specifically HRM_{avg}, models a similar vector addition operation but relies on inner products rather than metric spaces. By incorporating both translation and metric-space intuitions, TransRec is able to outperform PRME for all datasets and HRM_{avg} for Amazon and MovieLens. We observe similar results for their FM-inspired counterparts, with the translation and distance components of *TransFM* providing improved performance over related models.

5.1 Personalized Ranking Metric Embedding (PRME)

PRME [9] extends FPMC by learning personalized and sequential embeddings and replacing inner products with Euclidean distances (see Equation 5). To apply the PRME approach to FMs, we replace the inner product with the squared Euclidean distance between corresponding features. This gives the following *PRME-FM* model:

$$\hat{y}(\vec{x}) = w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} d^2(\vec{v}_i, \vec{v}_j) x_i x_j. \tag{8}$$

Note that *PRME-FM* is simply *TransFM* without the translation space. The model learns a single embedding and computes interaction weights according to the (squared) distance between embeddings. This model also retains the general-purpose nature of FMs and *TransFM* and can be simplified (similar to Section 3.2) to be computed in linear time.

5.2 Hierarchical Representation Model (HRM)

We next present a combined model between FMs and HRM [35]. HRM aggregates user and item representations (see Equation 6) prior to taking the inner product. We found above that average pooling is more effective, and HRM_{avg} is the best performing baseline for most Google Local datasets. We apply a similar intuition to the FM framework. Specifically, we adapt the first term of the inner product to take the sum of both embeddings, giving the following *HRM-FM* model:

$$\hat{y}(\vec{x}) = w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} \langle \vec{v}_i + \vec{v}_j, \vec{v}_i \rangle x_i x_j. \tag{9}$$

The sum $\vec{v}_i + \vec{v}_j$ is the aggregation term that combines the learned representations for features i and j prior to taking the inner product. This model is also a general-purpose algorithm and can be computed in linear time with a similar simplification as in Section 3.2.

5.3 Experiments

We compare *PRME-FM* and *HRM-FM* against standard FMs and *TransFM*. We evaluate these models against three datasets: 'Amazon Automotive,' 'Google Florida,' and 'MovieLens'. As in our previous experiments, models are evaluated according to the AUC in the following settings: (1) without features, (2) with temporal features, and (3) with content features. Results are presented in Table 4.

The overall trends for *PRME-FM* and *HRM-FM* are similar to FMs and *TransFM*. *TransFM* outperforms both *PRME-FM* and *HRM-FM* on Automotive and MovieLens, indicating that the increased expressiveness of the translation space more effectively models feature interactions. The models are similar in performance on the Florida dataset, potentially indicating a simpler relationship between features that is captured by all three approaches.

Table 4: Results for alternative FM-derived approaches. Models are evaluated according to the AUC (higher is better).

Model	Amazon Automotive	Google Florida	MovieLens
FM	0.6572	0.7057	0.8575
FM_{time}	0.6671	0.6757	0.8617
$\text{FM}_{\text{content}}$	0.7328	0.7821	0.8660
TransFM	0.6675	0.6507	0.8611
$\text{TransFM}_{\text{time}}$	0.6776	0.6233	0.8722
$\text{TransFM}_{\text{content}}$	0.8319	0.8095	0.9381
PRME-FM	0.6674	0.6501	0.8639
$\text{PRME-FM}_{\text{time}}$	0.6749	0.6240	0.8701
$\text{PRME-FM}_{\text{content}}$	0.7422	0.8115	0.8557
HRM-FM	0.6662	0.6521	0.8581
$\text{HRM-FM}_{\text{time}}$	0.6720	0.6281	0.8744
$\text{HRM-FM}_{\text{content}}$	0.7411	0.8160	0.8606

We do not observe a significant difference between *PRME-FM* and *HRM-FM* in terms of AUC. The sum and distance operations both improve on the inner product operation of standard FMs, and both capture a similar amount of signal in all evaluated datasets.

Compared to standard Factorization Machines, *HRM-FM*, *PRME-FM*, and *TransFM* all provide significantly improved AUC performance with content features. This demonstrates that merging FMs with specialized sequential algorithms can consistently lead to effective general-purpose recommendation models.

6 CONCLUSIONS AND FUTURE WORK

We introduced *TransFM*, which combines translation and metric-based approaches for sequential recommendation with Factorization Machines. This model learns an embedding and translation space for each feature and replaces the inner product of FMs with a translation term and distance metric. This general-purpose model natively supports the addition of content features without requiring specialized constraints or adjustments. We evaluated *TransFM* on a variety of datasets and found that it achieves state-of-the-art performance when incorporating content features. We also found that applying a similar intuition, combining FMs with other baselines, consistently leads to improved general-purpose models.

Future research directions include (1) applying *TransFM* to arbitrary machine learning tasks besides sequential recommendation, (2) determining the impact of additional features or feature representations on the model's performance, (3) performing a user study to further validate the results of *TransFM*, and (4) further investigating the relationship between *TransFM* and the simpler HRM_{avg} model, which performed well on the Google Local dataset.

REFERENCES

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. 2016. TensorFlow: A System for Large-Scale Machine Learning. In *OSDI*.
[2] Gerard Beenen, Kimberly Ling, Xiaoqing Wang, Klarissa Chang, Dan Frankowski, Paul Resnick, and Robert E Kraut. 2004. Using Social Psychology to Motivate

Contributions to Online Communities. In *CSCW*.
[3] Shay Ben-Elazar, Gal Lavee, Noam Koenigstein, Oren Barkan, Hilik Berezin, Ulrich Paquet, and Tal Zaccai. 2017. Groove Radio: A Bayesian Hierarchical Model for Personalized Playlist Generation. In *WSDN*.
[4] James Bennett, Stan Lanning, et al. 2007. The Netflix Prize. In *Proceedings of KDD cup and workshop*.
[5] Allison JB Chaney, David M Blei, and Tina Eliassi-Rad. 2015. A Probabilistic Model for Using Social Networks in Personalized Item Recommendation. In *RecSys*.
[6] Yan Chen, F Maxwell Harper, Joseph Konstan, and Sherry Xin Li. 2010. Social Comparisons and Contributions to Online Communities: A Field Experiment on MovieLens. *American Economic Review* (2010).
[7] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep Neural Networks for YouTube Recommendations. In *RecSys*.
[8] Yi Ding and Xue Li. 2005. Time Weight Collaborative Filtering. In *CIKM*.
[9] Shanshan Feng, Xutao Li, Yifeng Zeng, Gao Cong, Yeow Meng Chee, and Quan Yuan. 2015. Personalized Ranking Metric Embedding for Next New POI Recommendation. In *IJCAI*.
[10] Flavio Figueiredo, Bruno Ribeiro, Jussara M Almeida, and Christos Faloutsos. 2016. TribeFlow: Mining & Predicting User Trajectories. In *WWW*.
[11] Peixin Gao, Hui Miao, John S Baras, and Jennifer Golbeck. 2016. Star: Semiring Trust Inference for Trust-aware Social Recommenders. In *RecSys*.
[12] F Maxwell Harper and Joseph A Konstan. 2016. The MovieLens Datasets: History and Context. *TiiS* (2016).
[13] Ruining He, Wang-Cheng Kang, and Julian McAuley. 2017. Translation-based Recommendation. In *RecSys*.
[14] Xiangnan He and Tat-Seng Chua. 2017. Neural Factorization Machines for Sparse Predictive Analytics. In *SIGIR*.
[15] Diederik P Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *arXiv preprint arXiv:1412.6980* (2014).
[16] Yehuda Koren. 2010. Collaborative Filtering with Temporal Dynamics. *Commun. ACM* (2010).
[17] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix Factorization Techniques for Recommender Systems. *Computer* (2009).
[18] Greg Linden, Brent Smith, and Jeremy York. 2003. Amazon. com Recommendations: Item-to-item Collaborative Filtering. *IEEE Internet computing* (2003).
[19] Duen-Ren Liu, Chin-Hui Lai, and Wang-Jung Lee. 2009. A Hybrid of Sequential Rules and Collaborative Filtering for Product Recommendation. *Information Sciences* (2009).
[20] Jian-Guo Liu, Tao Zhou, and Qiang Guo. 2011. Information Filtering via Biased Heat Conduction. *Physical Review E* (2011).
[21] Qiang Liu, Shu Wu, Diyi Wang, Zhaokang Li, and Liang Wang. 2016. Context-aware Sequential Recommendation. In *ICDM*.
[22] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton Van Den Hengel. 2015. Image-based Recommendations on Styles and Substitutes. In *SIGIR*.
[23] Brian McFee and Gert R Lanckriet. 2010. Metric Learning to Rank. In *ICML*.
[24] Tuan-Anh Nguyen Pham, Xutao Li, and Gao Cong. 2017. A General Model for Out-of-Town Region Recommendation. In *WWW*.
[25] Steffen Rendle. 2010. Factorization Machines. In *ICDM*.
[26] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian Personalized Ranking from Implicit Feedback. In *UAI*.
[27] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2010. Factorizing Personalized Markov Chains for Next-Basket Recommendation. In *WWW*.
[28] Guy Shani, David Heckerman, and Ronen I Brafman. 2005. An MDP-based Recommender System. *JMLR* (2005).
[29] Yi Tay, Luu Anh Tuan, and Siu Cheung Hui. 2018. Multi-Pointer Co-Attention Networks for Recommendation. *arXiv preprint arXiv:1801.09251* (2018).
[30] Théo Trouillon, Christopher R Dance, Éric Gaussier, Johannes Welbl, Sebastian Riedel, and Guillaume Bouchard. 2017. Knowledge Graph Completion via Complex Tensor Factorization. *JMLR* (2017).
[31] Trinh Xuan Tuan and Tu Minh Phuong. 2017. 3D Convolutional Networks for Session-based Recommendation with Content Features. In *RecSys*.
[32] Farman Ullah, Ghulam Sarwar, Sung Chang Lee, Yun Kyung Park, Kyeong Deok Moon, and Jin Tae Kim. 2012. Hybrid recommender system with temporal information. In *ICOIN*. IEEE.
[33] Aaron Van den Oord, Sander Dieleman, and Benjamin Schrauwen. 2013. Deep Content-based Music Recommendation. In *NIPS*.
[34] Hao Wang, Yanmei Fu, Qinyong Wang, Hongzhi Yin, Changying Du, and Hui Xiong. 2017. A Location-Sentiment-Aware Recommender System for Both Home-Town and Out-of-Town Users. In *KDD*.
[35] Pengfei Wang, Jiafeng Guo, Yanyan Lan, Jun Xu, Shengxian Wan, and Xueqi Cheng. 2015. Learning Hierarchical Representation Model for Next Basket Recommendation. In *SIGIR*.
[36] Suhang Wang, Yilin Wang, Jiliang Tang, Kai Shu, Suhas Ranganath, and Huan Liu. 2017. What Your Images Reveal: Exploiting Visual Contents for Point-of-Interest Recommendation. In *WWW*.
[37] Shuai Wang, Mianwei Zhou, Geli Fei, Yi Chang, and Bing Liu. 2018. Contextual and Position-Aware Factorization Machines for Sentiment Classification. *arXiv preprint arXiv:1801.06172* (2018).
[38] Fuzheng Zhang, Nicholas Jing Yuan, Defu Lian, Xing Xie, and Wei-Ying Ma. 2016. Collaborative Knowledge Base Embedding for Recommender Systems. In *KDD*.
[39] Yan-Bo Zhou, Ting Lei, and Tao Zhou. 2011. A Robust Ranking Algorithm to Spamming. *EPL* (2011).