Transferring Semantic Categories with Vertex Kernels: Recommendations with SemanticSVD++

Matthew Rowe

School of Computing and Communications, Lancaster University, Lancaster, UK
m.rowe@lancaster.ac.uk

Abstract. Matrix Factorisation is a recommendation approach that tries to understand what factors interest a user, based on his past ratings for items (products, movies, songs), and then use this factor information to predict future item ratings. A central limitation of this approach however is that it cannot capture how a user's tastes have evolved beforehand; thereby ignoring if a user's preference for a factor is likely to change. One solution to this is to include users' preferences for semantic (i.e. linked data) categories, however this approach is limited should a user be presented with an item for which he has not rated the semantic categories previously; so called *cold-start categories*. In this paper we present a method to overcome this limitation by transferring rated semantic categories in place of unrated categories through the use of vertex kernels; and incorporate this into our prior $SemanticSVD^{++}$ model. We evaluated several vertex kernels and their effects on recommendation error, and empirically demonstrate the superior performance that we achieve over: (i) existing SVD and SVD^{++} models; and (ii) $SemanticSVD^{++}$ with no transferred semantic categories.

1 Introduction

Recommender systems have become a ubiquitous information filtering device that is prevalent across the web. At its core is the profiling of a user based on his prior behaviour to then forecast how he will behave in the future: i.e. how he will rate and consume items (movies, songs, products) thereafter. One of the most widespread approaches to recommending items to users is matrix factorisation; this method functions by *mining* a given user's affinity with a range of latent factors, in doing so allowing a user's rating for a given item to be predicted based on the dot-product ($\langle \mathbf{p}_u \mathbf{q}_i \rangle$) between the user's latent factor vector ($\mathbf{p}_u$) and the item's latent factor vector ($\mathbf{q}_i$). The problem with such approaches is that one cannot capture how a given user's tastes have *evolved* over time, given that the *mined* latent factors vary depending on the input - the *factor consistency problem*: where the latent factor vector indices of equivalent factors differ each time factorisation is performed. If we can capture information about how a user's tastes have changed then we can understand whether a user is likely to diverge in their preferences, or not, in the future and use this information to inform

P. Mika et al. (Eds.) ISWC 2014, Part I, LNCS 8796, pp. 341–356, 2014.

what recommendations to give. This prompts two research questions: *how can we capture the evolution of a user's tastes over time?* And *how can we include taste evolution information within a recommender system?*

In our recent prior work we presented a recommendation approach based on matrix factorisation called $SemanticSVD^{++}$ [9] - a modification of the existing SVD^{++} model [4] - that captures a user's preferences for semantic categories of items, and how these preferences have *evolved* over time. One limitation of this approach, however, is the existence of *cold-start categories* where a user has not rated an item from a given category before. To overcome this limitation in this paper we present a technique to transfer existing, rated semantic categories via vertex kernels, thereby leveraging existing semantic category ratings into the $SemanticSVD^{++}$ model. Our contributions in this paper are as follows:

1. Identification and quantification of the *cold-start categories* problem, its implications on semantic recommendation approaches, and a method to *transfer* existing rated semantic categories via vertex kernels that function over the linked data graph - presenting four of such vertex kernels.
2. An extension of the $SemanticSVD^{++}$ recommendation approach that captures users' taste evolution, based on semantic categories, and transfers *rated* semantic categories to overcome the *cold-start categories* problem.
3. An empirical evaluation of our approach that demonstrates improved performance when transferring semantic categories over: (i) existing SVD and $SVD++$ models; and (ii) not transferring rated semantic categories.

We have structured this paper as follows: section 2 presents the related work within the area of semantic recommender systems and how such existing approaches have functioned to date; section 3 explains the movie recommendation dataset that we used and the URI alignment procedure that we followed; section 4 explains the transferring of semantic categories to overcome the *cold-start category* problem; section 5 presents our approach for modelling users' taste profiles and how this enables the *evolution* of user's tastes to be captured; section 6 describes the $SemanticSVD^{++}$ model, the factors within the model, and the model learning procedure that we followed; section 7 presents our experimental evaluation of the model; section 8 discusses the limitations of this work and plans for future work; and section 9 draws the paper to a close with the conclusions gleaned from this work.

2 Related Work

There has been a flurry of recent work to combine recommender systems with the semantic web, and in particular linked data. Earlier work in this space was presented by Passant [8] to perform music recommendations by using the taste profile of a user's listening habits. The linked data graph could then be used to find the different pathways that connect two arbitrary nodes (e.g. resources of

bands and artists) and how they can be linked together. These paths were then used to gauge how *close* artists are, and thus recommend similar bands to what the user had listened to. The linked data graph has also been used in recent work by Di Noia et al. [2] to extract movie features for a content-based recommender system in order to build user profiles. In this instance, features specific to movie items (e.g. actors, directors of movies) were extracted from linked data based on the surrounding context of the item. Work by Ostuni et al. [7] expanded on the work of Di Noia et al. [2] by mining path-based features that link the past ratings of users together; in doing so the authors were able to induce a top-n ranking function for recommendations. Such is the potential for linked data within recommender systems, that the Extended Semantic Web Conference 2014 has also held a recommendation challenge where participants must recommend books to users; where such books are aligned with their semantic web URIs on the web of linked data.[1]

Our recent work [9] combined semantic taste evolution information into a recommender system by modelling users' tastes at the semantic category level, thereby overcoming the *factor consistency problem* - i.e. where mined latent factor vector indices of equivalent factors differ each time factorisation is performed. However this approach is limited when presented with items assigned to *cold-start categories*; where a given user has not rated an item's categories beforehand. This paper extends our prior work by first explaining how the linked data graph can be harnessed to identify similar, previously rated semantic categories, for a given user, and transfer these into the recommendation approach - thereby overcoming the problem of *cold-start categories* - to do this we use vertex kernels. To aid reader comprehension, we provide a brief overview of how we model and track users' ratings for semantic categories, and how the $SemanticSVD^{++}$ model functions - for a more thorough overview please see our prior work [9].

3 Movie Recommendation Dataset and URI Alignment

For our experiments we used the MovieTweetings dataset,[2] obtained from the RecSys wiki page[3] after being provided with the data by Doom et al. [3]. This dataset was obtained by crawling Twitter for Tweets that contained Internet Movie DataBase (IMDB) links coupled with a rating (out of 10). To enable unbiased testing of our recommendation approach, we divided the dataset up as follows: we ordered the ratings by their creation time and maintained the first 80% as the training set, the next 10% for the validation set, and the final 10% as the test set. As the dataset covers a period of around 30-weeks (March 2013 to September 2013), this meant that the training set contained the first 24 weeks, with the subsequent 3 weeks' segments containing the validating and testing portions. In the following sections the training set is analysed for users' semantic taste evolution and used to train the recommendation model, we use

[1] `http://2014.eswc-conferences.org/important-dates/call-RecSys`
[2] This was the November 2013 version.
[3] `http://recsyswiki.com/wiki/Movietweetings`

the validation set for hyperparameter tuning, and the held-out test set for the final experiments.

Movie recommendation datasets provide numeric IDs for movie items, which users have rated, and assign metadata to these IDs: title, year, etc. As we shall explain below, the user profiling approach models users' affinity to semantic categories, therefore we need to *align* the numeric IDs of movie items to their semantic web URIs - to enable the semantic categories that the movie has been placed within to be extracted.[4]. In this work we use DBPedia SKOS categories as our semantic categories.

The URI alignment method functioned as follows: first, we used the SPARQL query from [7] to extract all films (instances of `dbpedia-owl:Film`) from DBPedia which contained a year within one of their categories.[5] Using the extracted mapping between the movie URI (`?movie`) and title (`?title`) we then identified the set of *candidate URIs* (C) by performing fuzzy matches between a given item's title and the extracted title from DBPedia - using the normalised reciprocal Levenshtein distance and setting the similarity threshold to 0.9. We used fuzzy matches here due to the different forms of the movie titles and abbreviations within the dataset and linked data labels. After deriving the set of candidate URIs, we then dereferenced each URI and looked up its year to see if it appears within an associated category (i.e. `?movie dcterms:subject ?category`). If the year of the movie item appears within a mapped category (`?category`) then we identified the given semantic URI as denoting the item. This *disambiguation* was needed here as multiple films can share the same title (i.e. film remakes). This approach achieved coverage (i.e. proportion of items mapped) of 69%: this reduced coverage is explained by the recency of the movies being reviewed and the lack of coverage of this on DBPedia at present. Table 1 presents the statistics of the dataset following URI alignment.

From this point on we use the following notations to aid comprehension: u, v denote users, i, j denote items, r denotes a known rating value (where $r \in [1, 10]$), $\hat{r}$ denotes a predicted rating value, c denotes a semantic category that an item has been mapped to, and $cats(i)$ is a convenience function that returns the set of semantic categories of item i.

Table 1. Statistics of the MovieTweetings dataset with the reduction from the original dataset shown in parentheses

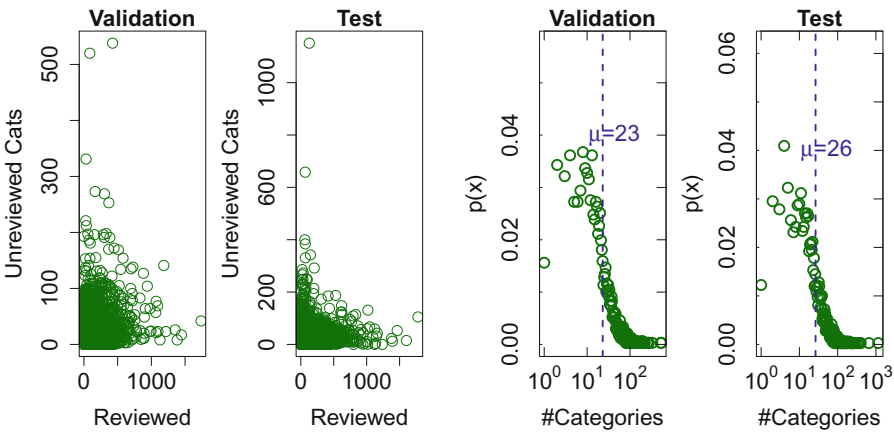
#Users	#Items	#Ratings	Ratings Period
14,749 (-22.5%)	7,913 (-30.8%)	86,779 (-25.9%)	[28-03-2013,23-09-2013]

[4] E.g. The 1979 Sci-Fi Horrow film Alien is placed within the semantic categories of `Alien_(franchise)_films` and `1979_horror_films`, by the `dcterms:subject` predicate.

[5] We used a local copy of DBPedia 3.9 for our experiments: `http://wiki.dbpedia.org/Downloads39`

4 Transferring Semantic Categories

A recommendation approach based upon semantic categories will assess how a user has rated items (books, movies, songs) beforehand, and associate those ratings with the items' categories to form a taste profile of the user. This taste profile, that captures the user's affinity for semantic categories, will then be used to predict how the user will rate a new item, given its semantic categories. A problem can arise here if the item's semantic categories have not been rated by the user: we define this as the *cold-start categories* problem. To demonstrate the extent of the problem, consider the plots shown in Fig. 1. In the left subfigure, the leftmost plot shows the association between the number of reviewed and unreviewed categories for each user when using the training dataset as background knowledge to form the profile of the user and the validation split to be the unseen data; while the rightmost figure shows the same association but with the training and validation splits as background data and the test split as the unseen data. In this instance we see that the more categories a user has rated, then the fewer categories they have missed. The right subfigure shows the relative frequency distribution of these missed categories, indicating a heavy-tailed distribution where a small number of categories are missed across many users. This indicates that cold-start categories hold additional information that a recommendation approach should consider.



(a) Reviewed vs. Unreviewed Categories (b) Distribution of Unreviewed Categories

Fig. 1. The left plot shows the association between the number of reviewed and unreviewed categories for each user, and the right plot showing the relative frequency distribution of unreviewed categories within each split

4.1 Category Transfer Function and Vertex Kernels

To overcome the problem of cold-start categories we can include a user's preferences for *rated* semantic categories, where such categories are *similar* to the

unreviewed categories. This is where the utility of the semantic web becomes evident: as the semantic categories are related to one another via SKOS relations (i.e. broader, narrower, related, etc.), we can identify semantic categories that are similar to the unreviewed categories by how they are related within the linked data graph - in a similar vein to prior work [8,2]:

Definition 1 (Linked Data Graph). *Let $G = \langle V, E, L \rangle$ denote a linked data graph, where $c \in V$ is the set of concept vertices within the graph (i.e. resources, classes), $e_{cd} \in E$ is an edge, or link, connecting $c, d \in V$ and $\sigma(e_{cd}) \in L$ denotes a label of the edge - i.e. the predicate associating c with d.*

Now, let C be the set of categories that a given user has rated previously, and D be the set of categories that a given item has been mapped to, then we define the *Category Transfer function* as follows:

$$f(C, D) = \{\arg\max_{c \in C} k\big(c, d\big) : d \in D\} \tag{1}$$

The codomain of the Category Transfer function is therefore a subset of the set of categories that the user has rated beforehand (i.e. $C' \subset C$). In the above function the vertex kernel k computes the similarity between categories c and d: this is often between the features vectors of the categories returned by the mapping function ϕ:

Definition 2 (Vertex Kernel). *Given graph vertices c and d from a linked data graph G', we define a vertex kernel (k) as a surjective function that maps the product of two vertices' attribute vectors into a real valued space, where $\phi(c)$ is a convenience function that returns kernel-specific attributes to be used by the function (i.e. an n-dimensional attribute vector of node c: $\phi(c) \in \mathbb{R}^n$). Hence:*

$$k : V \times V \to \mathbb{R} \tag{2}$$

$$k(\phi(c), \phi(d)) \longmapsto x \tag{3}$$

Given this formulation, we can vary the kernel function ($k(.,.)$) to measure the similarity between arbitrary categories based on the topology of the linked data graph that surrounds them. All the kernels considered in this paper function over two nodes' feature vectors. Therefore, to derive the feature vector for a given category node (c), we include information about the objects that c is linked to within the linked data graph. Let `<c ?p ?o>` define a triple where c appears within the subject position. We can then populate a vector ($\mathbf{x}$) based on the object concepts that c links to over 1-hop: $\phi^1 \in \mathbb{R}^m$ - where m denotes the dimensionality of the vector space. This can also be extended to n hops away from c by traversing edges away from c and collecting the objects within the traversed triples. Each element in the vector is weighted by the out-degree of c, thereby capturing the probability of c linking to a given category. Given the derivation of a *Triple-Object Vector*, using ϕ^n, for each category node, we varied the vertex kernel between the four functions shown in Table 2.

Table 2. Vertex kernels used to measure concept node similarities

Vertex Kernel	Function
Cosine	$k^n_{cos}(c,d) = \arccos \frac{\phi^n(c) \cdot \phi^n(d)}{\|\phi^n(c)\| \, \|\phi^n(d)\|}$
Dice	$k^n_{dice}(c,d) = \frac{2\left(\phi^n(c) \cdot \phi^n(d)\right)}{\sum_{i=1}^{\|\phi^n(c)\|} \phi^n_i(c) + \sum_{i=1}^{\|\phi^n(d)\|} \phi^n_i(d)}$
Squared Euclidean	$k^n_{se}(c,d) = \left(\sum_{i=1}^{\|\phi^n(c)\|} \left(\phi^n_i(c) - \phi^n_i(d)\right)^2 \right)^{-1}$
Jensen-Shannon Divergence	$k^n_{js}(c,d) = \left(\frac{1}{2} \sum_{i=1}^{\|\phi^n(c)\|} \phi^n_i(c) \ln \left(\frac{2\phi^n_i(c)}{\phi^n_i(c) \times \phi^n_i(d)} \right) + \frac{1}{2} \sum_{i=1}^{\|\phi^n(d)\|} \phi^n_i(d) \ln \left(\frac{2\phi^n_i(d)}{\phi^n_i(c) \times \phi^n_i(d)} \right) \right)^{-1}$

5 User Profiling: Semantic Categories

Semantic taste profiles describe a user's preferences for semantic categories at a given point in time, we are interested in understanding how these profiles change. Recent work [6] assessed user-specific evolution in the context of review platforms (e.g. BeerAdvocate and Beer Review) and found users to evolve based on their own '*personal clock*'. If we were to segment a user's lifetime (i.e. time between first and last rating) in our recommendation dataset into discrete lifecycle periods where each period is the same width in time, then we will have certain periods with no activity in them: as the user may go away, and then return later. To counter this we divided user's lifecycle into 5 stages where each stage contained the same number of reviews - this was run for users with ≥ 10 ratings within the training set. Prior work has used 20 lifecycle stages [6] to model user development, however we found this number to be too high as it dramatically reduced the number of users for whom we could mine taste evolution information - i.e. a greater number of stages requires more ratings.

To form a semantic taste profile for a given user we used the user's ratings distribution per semantic category within the allotted time window (provided by the lifecycle stage of the user as this denotes a closed interval - i.e. $s = [t, t'], t < t'$). We formed a discrete probability distribution for category c at time period $s \in S$ (where S is the set of 5 lifecycle stages) by *interpolating* the user's ratings within the distribution. We first defined two sets, the former ($D^{u,s,c}_{train}$) corresponding to the ratings by u during period/stage s for items from category c, and the latter ($D^{u,s}_{train}$) corresponding to ratings by u during s, hence $D^{u,s,c}_{train} \subseteq D^{u,s}_{train}$, using the following construct:

$$D^{u,s,c}_{train} = \{(u,i,r,t) : (u,i,r,t) \in D_{train}, t \in s, c \in cats(i)\} \tag{4}$$

We then derived the discrete probability distribution of the user rating category c favourably as follows, defining the set $C^{u,s}_{train}$ as containing all unique categories of items rated by u in stage s:

$$Pr(c|D^{u,s}_{train}) = \frac{\frac{1}{|D^{u,s,c}_{train}|}\sum_{(u,i,r,t)\in D^{u,s,c}_{train}} r}{\sum_{c'\in C^{u,s}_{train}} \frac{1}{|D^{u,s,c'}_{train}|}\sum_{(u,i,r,t)\in D^{u,s,c'}_{train}} r} \tag{5}$$

We only consider the categories that item URIs are directly mapped to; i.e. categories connected to the URI by the `dbterms:subject` predicate.

5.1 Taste Evolution

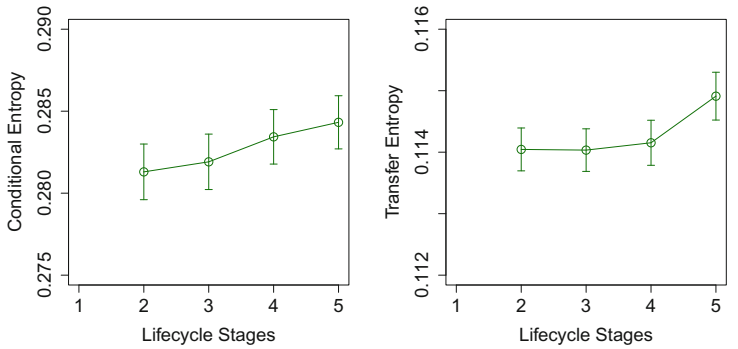
We now turn to looking at the evolution of users' tastes over time in order to understand how their preferences change. Given our use of probability distributions to model the lifecycle stage specific taste profile of each user, we can apply information theoretic measures based on information entropy. We used conditional entropy to assess the information needed to describe the taste profile of a user at one time step (Q) using his taste profile from the previous stage (P). A reduction in conditional entropy indicates that the user's taste profile is similar to that of his previous stage's profile, while an increase indicates the converse:

$$H(Q|P) = \sum_{\substack{x\in P,\\ y\in Q}} p(x,y)\log\frac{p(x)}{p(x,y)} \tag{6}$$

Our second measure captures the influence that users *in general* have on the taste profiles of individual users - modelling user-specific (local) taste development and global development as two different processes. We used transfer entropy to assess how the taste profile (P_s) of a user at one time step (s) has been influenced by (his own) local profile (P_{s-1}) and global taste profile (Q_{s-1}) at the previous lifecycle stage ($s-1$). For the latter taste profile (Q_{s-1}), we formed a global probability distribution (as above for a single user) using all users who posted ratings within the time interval of stage s. Now, assume that we have a random variable that describes the local categories that have been reviewed at the current stage (Y_s), a random variable of local categories at the previous stage (Y_{s-1}). and a third random variable of global categories at the previous stage (X_{s-1}), we then define the transfer entropy of one lifecycle stage to another as [10]: $T_{X\rightarrow Y} = H(Y_s|Y_{s-1}) - H(Y_s|Y_{s-1}, X_{s-1})$. Using the above probability distributions we can calculate the transfer entropy based on the joint and conditional probability distributions given the values of the random variables from Y_s, Y_{s-1} and X_{s-1}:

$$T_{X\rightarrow Y} = \sum_{\substack{y\in Y_s,\\ y'\in Y_{s-1},\\ x\in X_{s-1}}} p(y,y',x)\log\frac{p(y|y',x)}{p(y|y')} \tag{7}$$

We derived the conditional entropy and transfer entropy over the 5 lifecycle periods in a pairwise fashion, i.e. $H(P_2|P_1)$, for each user, and plotted the curve of the mean conditional and transfer entropy in Figure 2 using the training split - including the 95% confidence intervals. For conditional entropy, we find that users tend to diverge in their ratings and categories over time, given the increase in the mean curve towards later portions of the users' lifecycles. While for transfer entropy, we find that users' transfer entropy increases over time, indicating that users are less influenced by global taste preferences, and therefore the ratings of other users.



(a) Conditional Entropy (b) Transfer Entropy

Fig. 2. Taste evolution of users at the semantic category level: (i) comparing their divergence from prior semantic category ratings (2(a)); and (ii) comparing their influence of global semantic category taste trends (2(b))

6 SemanticSVD++

In this section we present a brief overview of $SemanticSVD^{++}$ [9], an extension of Koren et al.'s earlier matrix factorisation model: SVD^{++} [4]. The predictive function of the model is shown in full in Eq. 8, we now explain each component.

$$\hat{r}_{ui} = \overbrace{\mu + b_i + b_u}^{\text{Static Biases}} + \overbrace{\alpha_i b_{i,cats(i)} + \alpha_u b_{u,cats(i)}}^{\text{Category Biases}} + \overbrace{\mathbf{q}_i^\intercal \left(\mathbf{p}_u + |R(u)|^{-\frac{1}{2}} \sum_{j \in R(u)} \mathbf{y}_j + |cats(R(u))|^{-\frac{1}{2}} \sum_{c \in cats(R(u))} \mathbf{z}_c \right)}^{\text{Personalisation Component}} \tag{8}$$

6.1 Static Biases

The static biases include the mean rating score (μ) across all ratings within the training segment; the item bias (b_i), and the user bias (b_u). The item bias is the average deviation from the mean bias for the item i within the training segment, while the user bias is the average deviation from the mean bias from the training segment's ratings by user u.

6.2 Item Biases Towards Categories

We model the biases that an item may have given the categories it has been linked to by capturing the proportional change in category ratings - i.e. in general over the provided training portion. Let Q_s be the global taste profile (discrete probability distribution of all categories) in stage s, and k be the number of stages *back* in the training segment from which either a monotonic increase or decrease in the probability of rating category c began from, then the global taste development for c is defined as follows:

$$\delta_c = \frac{1}{4-k} \sum_{s=k}^{4} \frac{Q_{s+1}(c) - Q_s(c)}{Q_s(c)} \tag{9}$$

From this we then calculated the conditional probability of a given category being rated highly by accounting for the change rate of rating preference for the category as follows:

$$Pr(+|c) = \overbrace{Q_5(c)}^{\text{Prior Rating}} + \overbrace{\delta_c Q_5(c)}^{\text{Change Rate}} \tag{10}$$

By averaging this over all categories for the item i we can calculate the evolving item bias from the provided training segment:

$$b_{i,cats(i)} = \frac{1}{|cats(i)|} \sum_{c \in cats(i)} Pr(+|c) \tag{11}$$

6.3 User Biases Towards Categories: Vertex Kernels

To capture the development of a user's preference for a category we derived the average change rate (δ_c^u) over the k lifecycle periods coming before the final lifecycle stage in the training set. The parameter k is the number of stages *back* in the training segment from which either a monotonic increase or decrease in the probability of rating category c highly began from:

$$\delta_c^u = \frac{1}{4-k} \sum_{s=k}^{4} \frac{P_{s+1}^u(c) - P_s^u(c)}{P_s^u(c)} \tag{12}$$

We captured the change in transfer entropy for each user over time and modelled this as a *global influence factor* σ^u, based on measuring the proportional

change in transfer entropy starting from lifecycle period k that produced a monotonic increase or decrease in transfer entropy:

$$\sigma^u = \frac{1}{4-k} \sum_{s=k}^{4} \frac{T_{Q\to P}^{s+1|s} - T_{Q\to P}^{s|s-1}}{T_{Q\to P}^{s|s-1}} \tag{13}$$

By combining the average change rate (δ_c^u) of the user highly rating a given category c with the global influence factor (σ^u), we then derived the conditional probability of a user rating a given category highly as follows, where P_5^u denotes the taste profile of the user observed for the final lifecycle stage (5):

$$Pr(+|c,u) = \overbrace{P_5^u(c)}^{\text{Prior Rating}} + \overbrace{\delta_c^u P_5^u(c)}^{\text{Change Rate}} + \overbrace{\sigma^u Q_5(c)}^{\text{Global Influence}} \tag{14}$$

We then took the average across all categories as the bias of the user given the categories of the item:

$$b_{u,cats(i)} = \frac{1}{|cats(i)|} \sum_{c \in cats(i)} Pr(+|c,u) \tag{15}$$

Although the above summation will quantify the bias for categories linked to item i for which the user has provided a rating beforehand, the bias will ignore any categories for which the user has yet to provide ratings - our so-called *cold-start categories* - a limitation of the approach presented in our prior work [9]. Therefore, to counteract this we used the Category Transfer Function for a given vertex kernel to incorporate the most *similar* categories that the user u has rated before. Let $C \equiv cats(D_{train}^u)$, then we define the bias of the user given the categories of item i as follows:

$$\begin{aligned} b_{u,cats(i)} = &\Big(\beta_k\Big) \overbrace{\frac{1}{|C \cap cats(i)|} \sum_{c \in \{cats(i) \cap C\}} Pr(+|c,u)}^{\text{Prior Rated Categories}} \\ &+ \Big(1-\beta_k\Big) \overbrace{\frac{1}{|f_k(C, cats(i)/C)|} \sum_{c \in f_k(C, cats(i)/C)} Pr(+|c,u)}^{\text{Transferred Categories}} \end{aligned} \tag{16}$$

Here we have β_k-weighted the influence of the transferred categories on the bias in order assess the effects of the transferred categories on recommendation accuracy. In essence, β_k forms one of our hyperparameters that we optimise when tuning the model over the validation set for a given vertex kernel (k). As $\beta_k \in [0,1]$ we can assess its effect: a larger β_k places more emphasis on known information, while a lower β_k places more emphasis on transferred categories by the given kernel (k). As the category biases are, in essence, *static* features we

included two weights, one for each category bias, defined as α_i and α_u for the item biases to categories and the user biases to categories respectively - these weights are then learnt during the training phase of inducing the model.

6.4 Personalisation Component

The personalisation component of the $SemanticSVD^{++}$ model builds on the existing SVD^{++} model by Koren et al. [4] by including four latent factor vectors: $\mathbf{q}_i \in \mathbb{R}^f$ denotes the f latent factors associated with the item i; $\mathbf{p}_u \in \mathbb{R}^f$ denotes the f latent factors associated with the user u; $\mathbf{y}_j \in \mathbb{R}^f$ denotes the f latent factors for item j from the set of rated items by user u: $R(u)$; and we have defined a new vector $\mathbf{z}_c \in \mathbb{R}^f$ which captures the latent factor vector, of f-dimensions, for a given semantic category c. This latter component captures the affinity of semantic categories with latent factors.

6.5 Model Learning

In order to learn our recommendation model (item and user biases, category bias weights, latent factor vectors) we sought to minimise the following objective function (including L2-regularisation of parameters):

$$\min_{b_*,\alpha_*,p_*,q_*} \sum_{(u,i,t,r)\in D} (r_{ui} - \hat{r}_{ui})^2 + \lambda(b_i^2 + b_u^2 + \alpha_i^2 + \alpha_u^2 + ||\mathbf{q}_i||_2^2 + ||\mathbf{p}_u||_2^2)$$

Stochastic Gradient Descent (SGD) [1] was used to learn the parameters by first shuffling the order of the ratings within the training set, and then running through the set of ratings one at a time. For each rating we calculated the predicted rating based on the user and item with the current model parameters, we then updated the model's parameters based on the error: $e_{ui} = r_{ui} - \hat{r}_{ui}$. We stopped the learning procedure once we converged on stable parameter vectors (i.e. the difference in parameters is less than $\epsilon = 10^{-7}$). The update rules for our model are shown in table 3. A single regularisation weight (λ) and learning rate (η) are used for all parameters in the model.

One situation that arises within the data is where the user has no prior rating information for a user within the training segment - i.e. *cold-start users*. In this instance we used the mean rating (μ), the item static bias (b_i) and the category bias to the item (i) given the categories of the item ($b_{i,cats(i)}$): $\hat{r}_{ui}^{cold} = \mu + b_i + \alpha_i b_{i,cats(i)}$. This is an improvement of our prior approach [9] which did not address cold-start users.

7 Experiments

To test the efficacy of our recommendation model we used the existing SVD and SVD^{++} models as baselines, and tested two varieties of $SemanticSVD^{++}$: SVD^{++} with Semantic Biases (Ψ_{SB-SVD}); and $SemanticSVD^{++}$ (Ψ_{S-SVD}),

Table 3. Update rules for each component within the $SemanticSVD^{++}$ model

Model Parameter	Update Rule
Item bias	$b_i \leftarrow b_i + \eta(e_{ui} - \lambda b_i)$
User bias	$b_u \leftarrow b_u + \eta(e_{ui} - \lambda b_u)$
Item category bias weight	$\alpha_i \leftarrow \alpha_i + \eta(e_{ui} - \lambda \alpha_i)$
User category bias weight	$\alpha_u \leftarrow \alpha_u + \eta(e_{ui} - \lambda \alpha_u)$
Item vector	$\mathbf{q}_i \leftarrow \mathbf{q}_i + \eta\big(e_{ui}(\mathbf{p}_u + \lvert R(u)\rvert^{-\frac{1}{2}} \sum_{j \in R(u)} \mathbf{y}_j + \lvert cats(R(u))\rvert^{-\frac{1}{2}} \sum_{c \in cats(R(u))} \mathbf{z}_c) - \lambda \mathbf{q}_i\big)$
User vector	$\mathbf{p}_u \leftarrow \mathbf{p}_u + \eta(e_{ui}\mathbf{q}_i - \lambda \mathbf{p}_u)$
User items vector	$\forall j \in R(u)$: $\mathbf{y}_j \leftarrow \mathbf{y}_j + \eta(e_{ui}\lvert R(u)\rvert^{-\frac{1}{2}}\mathbf{q}_i - \lambda \mathbf{y}_j)$
User categories vector	$\forall c \in cats(R(u))$: $\mathbf{z}_c \leftarrow \mathbf{z}_c + \eta(e_{ui}\lvert cats(R(u))\rvert^{-\frac{1}{2}}\mathbf{q}_i - \lambda \mathbf{z}_c)$

which was the full model that we proposed earlier that includes latent factors for semantic categories. For these latter two models we tested the four vertex kernels and with the use of no kernel - to see how category transfer affected performance. We first performed model tuning, which we explain in more detail below, before then applying the best model, once hyperparameters had been selected. For model tuning, each recommendation model is trained using the training split and applied to the validation split, while for model testing each model is trained using both the training and validation split and applied to the test split. Our aim in both instances is to minimise the Root Mean Square Error (RMSE) over the respective test segment.

7.1 Model Tuning

In order to select the best model for application over the held-out test segment, we tuned the hyperparameters of each of the three models. For SVD and SVD^{++} we had three hyperparameter to tune: the regularisation weight (λ), the learning rate (η) and the number of factors (f). While for SVD^{++} with semantic biases, and $SemanticSVD^{++}$ we have four kernels, each of which requires four hyperparameters to be tuned: the regularisation weight (λ), the learning rate (η), the number of factors (f), and the preference of transferred categories (β_k).[6] We varied these hyperparameters through the following settings, using an exhaustive grid search to find the combination that produced the lowest RMSE: $\lambda = \{10^{-9}, 10^{-8}, \ldots, 10^{0}\}$; $\eta = \{10^{-7}, 10^{-6}, \ldots, 10^{-1}\}$; $\mathbf{f} = \{5, 10, 20, 50, 100\}$; $\beta_k = \{0, 0.1, \ldots, 1\}$. This was performed by searching the hyperparameter space using our parallel processing cluster (11 x AMD Quad Core machines each with 16Gb RAM and 2Tb disk space) - i.e. optimising the model's parameters with SGD given the hyperparameters and reporting the error over the validation split.

[6] We set $n = 1$ for each of the kernels therefore we are only forming feature vectors that are 1-top away from each category.

7.2 Results: Ratings Prediction Error

We now report on the results from forecasting the ratings within the test set based on the optimised models following hyperparmater tuning. Table 4 presents the RMSE values that we achieved. In order to assess for chance effects we performed significance testing using the Mann-Whitney test to assess for differences in location between the SVD^{++} baseline model and each of the proposed models (with different kernels) - after randomly splitting the test segment into 25-folds and macro-averaging the RMSE.[7] We find that for all models we achieved a statistically significant reduction in RMSE over the baseline - with the significance probability levels indicated. The results also indicate that the inclusion of transferred categories reduces prediction error over the use of no vertex kernel, thereby suggesting that the use of prior rating information from related categories boosts performance.

We find that the Cosine kernel performs best over both SVD^{++} with semantic biases, and $SemanticSVD^{++}$, in each case with a higher β_k weighting. Under this weighting scheme, a higher β_k places more emphasis on the item's categories that the user has previously rated, rather than transferring in ratings to cover the unreviewed categories. We find varying levels across the other kernels where, aside from the JS-Divergence kernel, the optimised β_k places more emphasis on using rated semantic categories that the item is aligned to.

Table 4. Root Mean Square Error (RMSE) results with each model's best kernel is highlighted in bold with the p-value of the Mann-Whitney with the baseline marked

Model	Kernel	Tuned Parameters	RMSE
Ψ_{SVD}	-	$\lambda = 0.001$, $\eta = 0.1$, $f = 50$	1.786
Ψ_{SVD++}	-	$\lambda = 0.01$, $\eta = 0.05$, $f = 100$	1.591
$\Psi_{SB-SVD++}$	-	$\lambda = 10^{-5}$, $\eta = 0.05$, $f = 100$	1.590*
	Cosine	$\lambda = 10^{-5}$, $\eta = 0.05$, $f = 20$, $\beta_k = 0.9$	**1.588*****
	Dice	$\lambda = 0.001$, $\eta = 0.05$, $f = 20$, $\beta_k = 0.7$	1.589**
	Squared-Euclidean	$\lambda = 10^{-5}$, $\eta = 0.05$, $f = 20$, $\beta_k = 0.6$	1.589**
	JS-Divergence	$\lambda = 0.01$, $\eta = 0.05$, $f = 50$, $\beta_k = 0.3$	1.590*
$\Psi_{S-SVD++}$	-	$\lambda = 0.001$, $\eta = 0.05$, $f = 20$	1.590*
	Cosine	$\lambda = 0.01$, $\eta = 0.05$, $f = 5$, $\beta_k = 0.8$	**1.588*****
	Dice	$\lambda = 0.001$, $\eta = 0.05$, $f = 20$, $\beta_k = 0.9$	1.590*
	Squared-Euclidean	$\lambda = 0.05$, $\eta = 0.05$, $f = 5$, $\beta_k = 0.7$	1.590*
	JS-Divergence	$\lambda = 10^{-4}$, $\eta = 0.05$, $f = 10$, $\beta_k = 0.8$	1.589**

Significance codes: p-value < 0.001 *** 0.01 ** 0.05 * 0.1 .

8 Discussions and Future Work

The introduction of *semantic level* taste information allows for the evolution of a user's preferences to be captured and used within a recommendation approach.

[7] N.b. all tested models significantly outperformed SVD at $p < 0.001$, so we do not report the different p-values here.

In this paper we have considered vertex kernels that transfer previously rated semantic categories by computing pairwise category similarity using triple-object vectors. One future direction of work will consider how the graph-space can be used, via traversal-based metrics, to compute the similarity between arbitrary pairs of category nodes. For instance, measures such as random walks hitting time and commute time, and the mixing rate of a random walk, measured over a subgraph of the linked data graph would be one future direction of work - forming the subgraph using the n-order egocentric network of the given category nodes.

Within this work we used a recent recommendation dataset derived from Twitter: MovieTweetings. Unlike existing movie recommendation datasets, such as MovieLens and NetFlix, this dataset suffers from a *recency* problem where the use of existing linked data datasets, such as DBPedia, are not timely enough to cover the items within the recommendation dataset - i.e. to provide URIs for those movie items. That said, we chose to use this single dataset as it presented a more recent resource to test our recommendation approach - as opposed to the heavily-subscribed MovieLens and Netflix datasets. Future work will examine the use of additional datasets, such as Freebase, for item to URI alignment that are more timely and could potentially lead to increased coverage of movie items and thus their alignment with semantic web URIs.

The objective function that we considered in this work, when optimising the presented recommendation approach, was the minimisation of the Root Mean Square Error. This objective has been criticised [5] as being unrealistic - i.e. in information filtering tasks limited screen-space renders a ranked list of items more appropriate. Therefore future work will focus on the adaptation of the approach to use a ranked-loss objective. Additionally, the optimisation procedure followed for identifying the best hyperparameters adopted an exhaustive grid-search approach, which is often intractable as the dimensionality of the dataset (i.e. number of items, and number of ratings) increases. Currently being explored is the use of Gaussian Processes in conjunction with Bayesian inference to estimate which portion of the hyperparameter space to examine next. This approach is necessary given the anticipated increased computational complexity that the graph-based kernels, mentioned above, will incur.

9 Conclusions

Recommender systems function by forming taste profiles of users, based on how they have rated items beforehand, and using those profiles to predict how the users will rate items in the future (e.g. movies, songs, products). One approach to forming such profiles is to capture how users have rated the semantic categories of items in the past, where such categories are linked to rated items. This approach is limited however in the presence of *cold-start categories*; semantic categories for which we have no prior rating information. In this paper we proposed a solution to this problem that uses the linked data graph space to identity similar categories that a user had previously rated, and transfer rating information from those categories to cover the unrated ones.. To demonstrate

the efficacy of this solution, we extended our prior $SemanticSVD^{++}$ approach [9] to transfer semantic category ratings using a variety of vertex kernels. This new approach was evaluated using the MovieTweetings dataset, collected from users' movie review Tweets, against the existing SVD and SVD^{++} models. We significantly outperformed these baselines with the use of no kernel, thus using the standard $SemanticSVD^{++}$ approach, while using the the four tested kernel functions improved performance further; significantly outperforming the standard $SemanticSVD^{++}$ approach. Our results indicate that the use of vertex kernels is an effective means to leverage ratings from previously rated semantic categories and thus overcome the *cold-start categories* problem.

References

1. Bottou, L., Bousquet, O.: The tradeoffs of large scale learning. In: NIPS, vol. 4, p. 2 (2007)
2. Di. Noia, T., Mirizzi, R., Ostuni, V.C., Romito, D., Zanker, M.: Linked open data to support content-based recommender systems. In: Proceedings of the 8th International Conference on Semantic Systems, pp. 1–8. ACM (2012)
3. Dooms, S., De Pessemier, T., Martens, L.: Movietweetings: a movie rating dataset collected from twitter. In: Workshop on Crowdsourcing and Human Computation for Recommender Systems, CrowdRec at RecSys, vol. 13 (2013)
4. Koren, Y.: Collaborative filtering with temporal dynamics. Communications of the ACM 53(4), 89–97 (2010)
5. Lee, J., Bengio, S., Kim, S., Lebanon, G., Singer, Y.: Local collaborative ranking. In: Proceedings of the 23rd International Conference on World Wide Web, International World Wide Web Conferences Steering Committee, pp. 85–96 (2014)
6. Julian McAuley and Jure Leskovec. From amateurs to connoisseurs: Modeling the evolution of user expertise through online reviews. In: Proceedings of World Wide Web Conference (2013)
7. Ostuni, V.C., Di Noia, T., Di Sciascio, E., Mirizzi, R.: Top-n recommendations from implicit feedback leveraging linked open data. In: 7th ACM Conference on Recommender Systems, RecSys 2013. ACM (2013)
8. Passant, A.: dbrec — music recommendations using dBpedia. In: Patel-Schneider, P.F., Pan, Y., Hitzler, P., Mika, P., Zhang, L., Pan, J.Z., Horrocks, I., Glimm, B. (eds.) ISWC 2010, Part II. LNCS, vol. 6497, pp. 209–224. Springer, Heidelberg (2010)
9. Rowe, M.: Semanticsvd++: Incorporating semantic taste evolution for predicting ratings. In: Web Intelligence Conference 2014 (2014)
10. Schreiber, T.: Measuring information transfer. Physical Review Letters 85(2), 461 (2000)