Metrical Representation of Readers and Articles in a Digital Newspaper

Jorge Díez
Artificial Intelligence Center
University of Oviedo
Gijón, Spain
jdiez@uniovi.es

David Martínez-Rego
Lab. for Research and
Development in AI (LIDIA)
Univ. da Coruña, Spain
dmartinez@udc.es

Amparo Alonso-Betanzos
Lab. for Research and
Development in AI (LIDIA)
Univ. da Coruña, Spain
ciamparo@udc.es

Oscar Luaces
Artificial Intelligence Center
University of Oviedo
Gijón, Spain
oluaces@uniovi.es

Antonio Bahamonde
Artificial Intelligence Center
University of Oviedo
Gijón, Spain
abahamonde@uniovi.es

ABSTRACT

Personalized recommendation of news in digital journals have to deal with important peculiarities. A majority of users (readers) are anonymous, and frequently news are volatile, they have an extremely short duration while other items arise. In this paper, we learn a mapping of users and items into a common Euclidean space where the similarities can be computed in a linear geometric context. The location of readers in the map are refined as they read more articles, and at the same time news can be inserted or removed easily. The metric properties of readers and news will pave the way for a solid base to offer recommendations for readers not only adjusted to their tastes, but with a certain degree of diversity or serendipity. Additionally, clusters of readers with similar interests or tastes could be discovered and exploited for marketing purposes. This mapping is learned using a scalable factorization algorithm that aims at optimizing the accuracy of the personalized recommendations. The paper includes an experimental study done with real word data.

CCS Concepts

•Information systems → Retrieval models and ranking; •Human-centered computing → Collaborative filtering; •Computing methodologies → Factorization methods;

Keywords

Metric Embedding; Matrix factorization; Learning to rank

1. INTRODUCTION

News recommenders aim to improve the reading experience of users and increase engagement. In [2] the authors underscore two very important characteristics of digital news: a) the set of items is continually changing, and b) there is a large set of users that most of the times are anonymous. Therefore, readers are only described by their reading trajectory, that typically is quite short.

In this paper, we propose to build a map into a Euclidean space, $\mathbb{R}^k$ for an integer k, to locate both readers and news according to reading affinities. The idea is to obtain a metric representation where we may use geometric tools to draw personalized recommendations and clusters of readers or news. In other words, the map is going to translate similarity into Euclidean distances or inner products.

The map is learned using a matrix factorization algorithm [3] to optimize personalized recommendations for readers given their reading trajectories. The purpose is to suggest for each reader an ordered list of news that other readers with similar trajectories have seen in the past.

The approach presented is a pure collaborative filter. This is the case, for instance, of the recommender described in [2] for users of Google News. And from a formal point of view, the learning task faced in this paper can be seen as a time series classification where the target class is a set of news, the sequels suggested to be read. In fact, if we would get rid of the order of these news, the learning task could be seen as a multilabel classification.

In any case, we take into account that although many suggestions may be made to specific readers, it is widely recognized that attention span can be damaged in numerous ways. For this reason, our objective function was designed in such a way as to incentivize accuracy regarding the top-ranked suggestions.

Other aspect of the approach presented here is the time required to update the model. This is important given the volatility of news. But since the proposed algorithm, an adaptation of [6], is a Stochastic Gradient Descent (SGD) optimization, it is suitable for demanding real-time scenarios. Moreover, the learning algorithm has to determine the location of readers and news, and then both can be updated easily. The situation of readers is refined as they read more articles, and at the same time news can be inserted or removed without affecting others.

RecSys '16, Boston, Massachusetts, USA
© 2016 ACM. ISBN 123-4567-24-567/08/06...$15.00
DOI: 10.475/123_4

In the next sections the formal framework and learning algorithm are described. Finally, we report some experiments carried out with a real world data from *El País*[1], Spain's most popular, and probably the most influential, newspaper in the global Spanish-speaking community.

2. FORMAL FRAMEWORK

The purpose is to define an *utility* function; that is, a function to estimate the *affinity* of readers and news. The recommendations for each reader will be given by the values of this utility. Then, we first discuss how to represent news and readers.

Let $\mathcal{N}$ be a set of digital news. Since we will not use any information about their contents, in this paper, each article $\boldsymbol{a} \in \mathcal{N}$ is going to be represented by a binary codification vector

$$\boldsymbol{a} \in \mathbb{R}^{|\mathcal{N}|}.$$

All components are zero except the one with index $\boldsymbol{a}$ that is 1. To avoid the introduction of new symbols, we use $\boldsymbol{a}$ both for the article and its vectorial representation. It is worth mentioning that the incorporation of contents in the representation of news is straightforward, see [4]),

Thus, a reader $\boldsymbol{r}$, given that we could only rely on the trajectory of the news read in one session, will be represented somehow by a sequence of vectors

$$\boldsymbol{r} \equiv [\boldsymbol{r}^0, \boldsymbol{r}^1, \boldsymbol{r}^2, \ldots], \tag{1}$$

where $\boldsymbol{r}^0$ is the last article read, $\boldsymbol{r}^1$ is the second last, and so on. Then, the dataset should be formed by pairs of sequences of news: the reading trajectory of the reader, and a sequel of news read after the last one:

$$\boldsymbol{d} = ([\boldsymbol{r}^0, \boldsymbol{r}^1, \boldsymbol{r}^2, \ldots], [\boldsymbol{a}^0, \boldsymbol{a}^1, \boldsymbol{a}^2, \ldots]). \tag{2}$$

A straightforward approach to tackle the recommendation task consists in recommending an ordered list of the most frequently accessed news from the article just read. In other words, we may use *1-grams* to model the sequence of articles read. This method uses the utility function

$$f_{1\mathrm{g}}(\boldsymbol{r}, \boldsymbol{a}) = f_{1\mathrm{g}}([\boldsymbol{r}^0, \boldsymbol{r}^1, \boldsymbol{r}^2, \ldots], \boldsymbol{a}) = \Pr(\boldsymbol{a}|\boldsymbol{r}^0). \tag{3}$$

However, f_{1-g} is blind to previous readings. We could try to avoid this flaw with an *n-gram* utility function, but then to make recommendations to a reader, we need to wait until he/she reads n news. In any case, if we use a probability distribution of the Linear Exponential Family, the ranking of news for a reader is the same than the one achieved with the general utility function

$$f(\boldsymbol{r}, \boldsymbol{a}) = \sum_{i,j} \alpha_{ij} r_i a_j. \tag{4}$$

This utility function is just a weighted sum of the products of the components of readers and news. Depending of how many news we include in the representation of a reader, we have different *n-gram* implementations.

On the other hand, to try to grasp the whole trajectory of a reader $\boldsymbol{r}$ at once, we may codify the sequence by the sum of the codification of all articles included on it

$$\boldsymbol{r} = \sum_{i \geq 0} \boldsymbol{r}^i \in \mathbb{R}^{|\mathcal{N}|}. \tag{5}$$

The weakness of this representation, *whole-t* in the rest of the paper, is that we ignore the ordering of the articles read by the user. All permutations of the same set of articles will give rise to the same representation.

Note that in all cases, the expression (4) is the utility function. But this expression may have too much parameters to learn, thus we search for a couple of matrices such that

$$f(\boldsymbol{r}, \boldsymbol{a}) = \boldsymbol{r}^{\boldsymbol{T}} \boldsymbol{W}^{\boldsymbol{T}} \boldsymbol{A} \boldsymbol{a} = \langle \boldsymbol{W}\boldsymbol{r}, \boldsymbol{A}\boldsymbol{a} \rangle. \tag{6}$$

Therefore, the parameters to be learned have a clear geometrical interpretation. They are mappings (embedding) from the spaces of readers and news into a common Euclidean space, $\mathbb{R}^k$ for some integer k. If $|R|$ is the dimension of the space to represent readers, the mappings are given by

$$\begin{aligned} \mathbb{R}^{|R|} &\to \mathbb{R}^k, \quad \boldsymbol{r} \mapsto \boldsymbol{W}\boldsymbol{r} \\ \mathbb{R}^{|\mathcal{N}|} &\to \mathbb{R}^k, \quad \boldsymbol{a} \mapsto \boldsymbol{A}\boldsymbol{a}. \end{aligned} \tag{7}$$

Then, the utility function can be seen in terms of a distance from an hyperplane,

$$\begin{aligned} f(\boldsymbol{r}, \boldsymbol{a}) &= \|\boldsymbol{W}\boldsymbol{r}\| \|\boldsymbol{A}\boldsymbol{a}\| \cos(\boldsymbol{W}\boldsymbol{r}, \boldsymbol{A}\boldsymbol{a}) \\ &= \|\boldsymbol{W}\boldsymbol{r}\| \, \mathrm{d}(\mathrm{hyper}(\boldsymbol{W}\boldsymbol{r}), \boldsymbol{A}\boldsymbol{a}). \end{aligned} \tag{8}$$

Thus, for a reader $\boldsymbol{r}$, the news are ordered by their distance to the hyperplane in $\mathbb{R}^k$ perpendicular to $\boldsymbol{W}\boldsymbol{r}$.

Hence, to make recommendations and sound representations of readers and news, we need to learn matrices $\boldsymbol{W}$ and $\boldsymbol{A}$ such that the determined ordering of news being coherent with the one observed as actual interest of readers. In the next subsection we present the loss function that we will optimize by means of Stochastic Gradient Descent (SGD) of obtain those matrices.

2.1 Loss function

We are interested in the top ranked news given by the utility function f. We are going to suggest only 5 news that hopefully the reader would be interested in reading. To measure the performance of these suggestions we check the percentage of those news that were actually read; that is, we compute the *Precision* with the 5 top ranked news; in symbols, $P_{@5}$. Therefore, $P_{@5}$ would be the perfect loss function. But unfortunately, this is not a maneuverable function to find optimal parameters. Then, following [6], we present a loss function somehow equivalent to $P_{@5}$ with mathematical properties that allows a smooth search for the optimal values of the parameters.

To introduce the loss function let $\boldsymbol{d}$ (2) an element of the dataset. The essence of a loss function for this learning task is to penalize values of the parameters such that the score of news in $[\boldsymbol{a}^0, \boldsymbol{a}^1, \boldsymbol{a}^2, \ldots]$ is lower than scores of those news not appearing neither in the past nor in the future reading records of the reader, $\boldsymbol{b} \notin \{\boldsymbol{r}^0, \boldsymbol{r}^1, \boldsymbol{r}^2, \ldots, \boldsymbol{a}^0, \boldsymbol{a}^1, \boldsymbol{a}^2, \ldots\}$. For ease of reference we call *positive* those news $(\boldsymbol{a}^i)$ in the future reading trajectory of the reader, and *negative* those like $\boldsymbol{b}$ that have not been read ever.

The loss function uses a so-called *maximum margin* approach, and so the aim is to find out parameters such that

$$f(\boldsymbol{r}, \boldsymbol{a}^i) \geq 1 + f(\boldsymbol{r}, \boldsymbol{b}) \tag{9}$$

for all positive news $(\boldsymbol{a}^i)$ and negative news $(\boldsymbol{b})$.

To introduce formally the loss function, we need to define the set of *violating news*. Let $\boldsymbol{a}^i$ be any positive article for

[1] http://www.elpais.com

a reader $\boldsymbol{r}$, then define

$$\text{vio}(\boldsymbol{d}, \boldsymbol{a}^i) = \tag{10}$$
$$\{\boldsymbol{b} \notin \{\boldsymbol{r}^0, \boldsymbol{r}^1, \ldots\} \cup \{\boldsymbol{a}^0, \boldsymbol{a}^1, \ldots\} : 1 + f(\boldsymbol{r},\boldsymbol{b}) > f(\boldsymbol{r},\boldsymbol{a}^i)\}.$$

Since we want to push the score of $\boldsymbol{a}^i$ as high as possible, our aim is to reduce the size of violating news. We define then the error due to the positive article $\boldsymbol{a}^i$ by an amount that depends on the number of violating news and the average violation of the margin of the articles in $vio(\boldsymbol{d}, \boldsymbol{a}^i)$

$$err(f, \boldsymbol{d}, \boldsymbol{a}^i) = \tag{11}$$
$$\sum_{b \in \text{vio}(\boldsymbol{d},\boldsymbol{a}^i)} L\Big(\big|\,\text{vio}(\boldsymbol{d}, \boldsymbol{a}^i)\big|\Big) \max\{0, 1 - f(\boldsymbol{r}, \boldsymbol{a}^i) + f(\boldsymbol{r}, \boldsymbol{b})\} \Pr(\boldsymbol{b}|\boldsymbol{d}).$$

L is a function that transforms the size of the set of violating news into a loss and it weights up the gravity of the violation: it is desirable that $\boldsymbol{a}^i$ is ranked on the top 5 positions. In general, for an integer t the goal is to optimize the $P_{@t}$. For this purpose, following [6], we define

$$L(t) = \sum_{j=1}^{t} \frac{1}{j}. \tag{12}$$

The problem with this approach is that it is not possible to determine the number of violating news without computing the scores of all negative articles, which is unfeasible in practice. In order to implement this loss function, in [6], the authors rely on a sampling that simplifies the calculations, the *Weighted Approximate-Rank Pairwise (WARP)*, and makes them affordable in datasets with large volumes of data. So, we sample news uniformly with replacement from the set of all news (excluding those already read) until we find a violating one. If N is the number of trials in the sampling, we approximate the size of the violating set by

$$\big|\,\text{vio}(\boldsymbol{d}, \boldsymbol{a}^i)\big| \cong \left\lfloor \frac{|\mathcal{N}| - 1 - |\boldsymbol{d}|}{N} \right\rfloor. \tag{13}$$

2.2 Algorithm

To find the parameters, $\boldsymbol{W}$ and $\boldsymbol{A}$, of the utility function (6), we use an adaptation of an algorithm presented in [6] (called WSABIE). Algorithm 1 details the optimization devised to solve the news recommender learning task. The algorithm is an SGD implementation with a regularization term: the sum of the Frobenius norms of matrices $\boldsymbol{W}$ and $\boldsymbol{A}$. Formally, the optimization problem to be solved is

$$\underset{\boldsymbol{W},\boldsymbol{A}}{\text{argmin}}\; err_{WARP}(\boldsymbol{W}, \boldsymbol{A}) + \text{reg}(\boldsymbol{W}, \boldsymbol{A}). \tag{14}$$

These matrices are initialized with random components drawn from a Gaussian distribution with 0 mean and standard deviation $1/\sqrt{k}$, where k is the dimension of the Euclidean common space. To update the matrices $\boldsymbol{W}$ and $\boldsymbol{A}$ (only the necessary columns) we need some derivatives that ease the gradient step:

$$\begin{aligned} \boldsymbol{W} &\leftarrow \boldsymbol{W} - \gamma\left(L \cdot (\boldsymbol{A}\boldsymbol{b} - \boldsymbol{A}\boldsymbol{a}^i)\boldsymbol{r}^{\mathrm{T}} + 2\nu \cdot \boldsymbol{W}\right) \\ \boldsymbol{A}\boldsymbol{a}^i &\leftarrow \boldsymbol{A}\boldsymbol{a}^i + \gamma(L \cdot \boldsymbol{W}\boldsymbol{r} - 2\nu \cdot \boldsymbol{A}\boldsymbol{a}^i) \\ \boldsymbol{A}\boldsymbol{b} &\leftarrow \boldsymbol{A}\boldsymbol{b} - \gamma\left(L \cdot \boldsymbol{W}\boldsymbol{r} + 2\nu \cdot \boldsymbol{A}\boldsymbol{b}\right). \end{aligned} \tag{15}$$

Notice that we only need to update two columns of matrix $\boldsymbol{A}$: the positive $\boldsymbol{a}^i$ and the negative $\boldsymbol{b}$. In all cases, the term L (12) is the estimation of WARP.

Algorithm 1 SGD algorithm to minimize the WARP error with a regularization term

```
Input: All pairs $\boldsymbol{d} = \big(\boldsymbol{r}, [\boldsymbol{a}^0, \boldsymbol{a}^1, \boldsymbol{a}^2, \ldots]\big)$, in dataset;
Input: $k > 0$, $\gamma > 0$ and $\nu > 0$
assign random values to the parameters of function $f$;
{Gaussians with mean 0 and standard deviation $1/\sqrt{k}$}
repeat
  Pick a random $\boldsymbol{d} = (\boldsymbol{r}, [\boldsymbol{a}^0, \boldsymbol{a}^1, \boldsymbol{a}^2, \ldots])$ and a positive label $\boldsymbol{a}^i$
  from $[\boldsymbol{a}^0, \boldsymbol{a}^1, \boldsymbol{a}^2, \ldots]$;
  $f(\boldsymbol{r}, \boldsymbol{a}^i) = \langle \boldsymbol{W}\boldsymbol{r}, \boldsymbol{A}\boldsymbol{a}^i \rangle$;
  Set $N = 0$;
  repeat
    Pick a random negative article $\boldsymbol{b}$ for $\boldsymbol{d}$; {not read in past or
    future trajectories}
    Set $N = N + 1$;
  until $f(\boldsymbol{r}, \boldsymbol{b}) > f(\boldsymbol{r}, \boldsymbol{a}^i) - 1$ or $N \geq |\mathcal{N}| - 1 - |\boldsymbol{d}|$
  if $f(\boldsymbol{r}, \boldsymbol{b}) > f(\boldsymbol{r}, \boldsymbol{a}^i) - 1$ then
    Make a gradient step to minimize: {see (15)}
    $L\Big(\Big\lfloor \frac{|\mathcal{N}|-1-|\boldsymbol{d}|}{N} \Big\rfloor\Big) \max\big\{0,1-f(\boldsymbol{r},\boldsymbol{a}^i)+f(\boldsymbol{r},\boldsymbol{b})\big\} + \nu \cdot \text{reg}(\boldsymbol{W},\boldsymbol{A})$
    Project weights to enforce constraints
    Update learning rate $\gamma$
  end if
until stop criterion
return $\boldsymbol{W}, \boldsymbol{A}$
```

3. EXPERIMENTAL RESULTS

The dataset used in this paper comes from the access logs to the web page of the digital version of a newspaper one single day. Each web page access is associated in the log file to its URL and a user identifier, allowing us to construct the trajectory of read news of each user (2). For training purposes we used only trajectories with at least two pieces of news (to apply 2-gram method), and at least one article in the future reading list.

Starting at 00:00 every half hour the reading data of 4 consecutive hours were collected in a set of data used for training, the readings of the next hour were used as test. We collected in this way 39 train/test pairs. The last test time finished at 24:00.

After training, we asked the three recommenders to be tested (*1-gram*, *2-gram* and *whole-t* (5)) to suggest 5 news that should be interesting for each reader with known reading trajectory including more that 5 news read. We checked the percentage of those 5 news that were actually read; in other words we computed *Precision* with the 5 top ranked news ($P_{@5}$). Therefore, in test sets we only used readers who read at least 7 news: 2 for past trajectory and the rest for future readings.

To fix the parameters of Algorithm 1, we made a grid search using the training set of records from 9:00 to 13:00 that was split in two halves with similar distribution of trajectories of the same length; first half was used for training with a selection of parameters, and second half was used for testing purposes. The parameters that yield best scores were selected for the experiments.

Table 1 shows the scores achieved by the recommendation methods. In 24 (respectively 26) out of 39 datasets, *2-gram* reaches better scores than *1-gram* (respectively *whole-t*). There are important differences in some scores, but the significance of these results is not the same since the test sets range from only 41 instances to 1846. However, differences are quite small if we compute the average of precisions weighted by the number of test examples. In fact, they are 42.47% for *1-gram*, 43.00% for *2-gram*, and 42.18% for the *whole-t* method. Thus, note that the scores obtained with *whole-t* are only slightly worse than the best ones.

Table 1: $P_{@5}$ obtained using 4 hours as training set and the next hour for test. The first column registers the beginning of the training period

	#Instances					
Begin	Train	Test	#News	*1-gram*	*2-gram*	*whole-t*
0:00	1013	41	172	38.54	**40.00**	33.17
0:30	834	50	166	34.40	**35.20**	30.00
1:00	694	67	159	31.34	**38.21**	27.76
1:30	579	90	148	37.78	**39.33**	36.67
2:00	506	92	134	39.13	**50.87**	45.43
2:30	499	195	146	**47.28**	44.31	35.08
3:00	503	391	148	42.51	**44.25**	40.92
3:30	625	641	160	45.74	**48.49**	44.65
4:00	884	967	171	47.28	**50.03**	42.92
4:30	1357	1352	183	45.13	**45.95**	44.81
5:00	2101	1450	188	**45.32**	45.05	43.09
5:30	3185	1784	195	**45.00**	44.37	43.81
6:00	4306	1846	207	**43.43**	43.38	42.72
6:30	5707	1774	219	**44.04**	44.01	43.28
7:00	7021	1695	237	44.86	44.94	**45.46**
7:30	8353	1562	258	**46.12**	46.02	45.77
8:00	9311	1452	261	**45.26**	44.37	44.35
8:30	9979	1233	270	**46.08**	45.24	45.19
9:00	10140	1033	274	41.72	42.15	**43.89**
9:30	10047	834	273	42.33	41.49	**42.40**
10:00	9622	631	274	39.81	40.06	**40.95**
10:30	8746	489	274	**35.95**	35.30	35.87
11:00	7616	472	269	37.16	37.25	**37.84**
11:30	6433	445	262	34.20	33.93	**35.15**
12:00	5558	476	266	**36.13**	34.96	36.05
12:30	4687	501	262	39.68	37.84	**40.20**
13:00	3955	445	257	37.21	38.79	**39.51**
13:30	3468	460	255	38.13	**40.09**	36.96
14:00	3127	585	253	39.66	**43.21**	40.48
14:30	3058	651	253	39.88	**45.53**	42.30
15:00	3339	572	253	40.21	**43.74**	41.15
15:30	3535	476	258	40.84	**42.06**	41.85
16:00	3618	410	259	38.49	**40.59**	39.12
16:30	3616	390	266	36.15	**37.85**	37.49
17:00	3575	359	264	**38.77**	38.72	38.66
17:30	3432	331	265	34.80	36.80	**36.92**
18:00	3266	321	273	36.32	37.26	**38.32**
18:30	2954	308	276	33.12	**34.94**	33.18
19:00	2679	302	279	36.75	36.62	**39.87**

To close the section let us report time consumption of the Algorithm 1. In the experiments we used a computer with an Intel i7, 2.2 GHz quad-core with 8 GB of RAM memory, but we used only a single core in these experiments to avoid multi-threading optimizations in matrix operations. Average training time in seconds: 65 (*1-gram*), 225 (*2-gram*), 314 (*whole-t*). In none of the training experiments, the time for the whole-t method (the slowest one) was over 400 seconds. Therefore, we can obtain a new model every few minutes. This is important since volatility is high in our target scenario: new articles appear and old articles disappear while people are reading news online.

4. CONCLUSIONS AND FUTURE WORK

We have presented 3 methods to suggest news to readers with similar accuracy. The 1 and 2 *grams* can be seen as solid baseline given the scarce information available. The codification used by *whole-t*, on the other hand, provides Euclidean representations of readers attending to the *whole* set of news read. Thus, the behavior of a reader $\boldsymbol{r}$ can be represented by this method (5) to be mapped to $\boldsymbol{W}\boldsymbol{r} \in \mathbb{R}^k$ and then obtain a reasonably accuracy in the recommendations for $\boldsymbol{r}$. Additionally, the mapping may be used by visualization tools, or to cluster readers in $\mathbb{R}^k$ according to their reading taste. These clusters may be useful, for instance, for marketing purposes. If we had registered readers we may also record their preferences using a single $\mathbb{R}^k$ point, and then offer them every new day a *personalized newspaper*.

From the point of view of news, each article $\boldsymbol{a}$ is also mapped into $\mathbb{R}^k$ by $\boldsymbol{A}\boldsymbol{a}$ (using the matrix $\boldsymbol{A}$ learned by the *whole-t* method). This map can provide suggestions with customizable degrees of diversity or *serendipity* [5, 7]. Notice that in this approach, once we have a Euclidean representation, a trip from user's location crossing the map of news provides the list of suggestions. In this way, the level of diversity in recommendations (*serendipity*) is proportional to the average dissimilarity of news and reader locations in the Euclidean representation.

We think that a future work on this idea would improve the reading experience searching for a trade-off between accuracy and diversity [1]. We postulate that both accuracy and diversity are two sides of the same coin when it comes to recommendation systems. Diversity should be grounded in a solid representation of users and items built aiming to optimize accuracy, a mapping procedure based on learning an accurate recommender system with a factorization method.

Acknowledgments

This work was funded by grants TIN2015-65069-C2-1-R and TIN2015-65069-C2-2-R from Ministerio de Economía y Competitividad. David Martínez-Rego acknowledges support from the Xunta de Galicia under postdoctoral grant code POS-A/2013/196. We also would like to thank El País for providing us with the dataset used in this paper.

5. REFERENCES

[1] S. Chen, J. Moore, D. Turnbull, and T. Joachims. Playlist Prediction Via Metric Embedding. In *Proc. of the* 18^{th} *ACM SIGKDD*, pages 714–722, 2012.

[2] A. S. Das, M. Datar, A. Garg, and S. Rajaram. Google News Personalization: Scalable Online Collaborative Filtering. In *Proc. of the 16th WWW*, pages 271–280. ACM, 2007.

[3] Y. Koren, R. Bell, and C. Volinsky. Matrix Factorization Techniques for Recommender Systems. *Computer*, 42(8):30 –37, aug. 2009.

[4] O. Luaces, J. Díez, T. Joachims, and A. Bahamonde. Mapping preferences into euclidean space. *Expert Systems with Applications*, 42(22):8588–8596, 2015.

[5] S. Vargas, L. Baltrunas, A. Karatzoglou, and P. Castells. Coverage, redundancy and size-awareness in genre diversity for recommender systems. In *Proceedings of the 8th ACM RecSys*, pages 209–216, 2014.

[6] J. Weston, S. Bengio, and N. Usunier. Large Scale Image Annotation: Learning to Rank with Joint Word-Image Embeddings. *Machine Learning Journal*, 81(1):21–35, 2010.

[7] C.-N. Ziegler, S. M. McNee, J. A. Konstan, and G. Lausen. Improving recommendation lists through topic diversification. In *Proc. of the 14th WWW*, pages 22–32. ACM, 2005.