Improving Collaborative Metric Learning with Efficient Negative Sampling

Viet-Anh Tran, Romain Hennequin, Jimena Royo-Letelier, Manuel Moussallam
Deezer Research & Development, Paris, France
research@deezer.com

ABSTRACT

Distance metric learning based on triplet loss has been applied with success in a wide range of applications such as face recognition, image retrieval, speaker change detection and recently recommendation with the Collaborative Metric Learning (CML) model. However, as we show in this article, CML requires large batches to work reasonably well because of a too simplistic uniform negative sampling strategy for selecting triplets. Due to memory limitations, this makes it difficult to scale in high-dimensional scenarios. To alleviate this problem, we propose here a 2-stage negative sampling strategy which finds triplets that are highly informative for learning. Our strategy allows CML to work effectively in terms of accuracy and popularity bias, even when the batch size is an order of magnitude smaller than what would be needed with the default uniform sampling. We demonstrate the suitability of the proposed strategy for recommendation and exhibit consistent positive results across various datasets.

CCS CONCEPTS

• **Information systems → Recommender systems**; • **Computing methodologies → Metric learning**;

KEYWORDS

Recommender Systems, Collaborative Filtering, Triplet Loss, Metric Learning

ACM Reference Format:
Viet-Anh Tran, Romain Hennequin, Jimena Royo-Letelier, Manuel Moussallam. 2019. Improving Collaborative Metric Learning with Efficient Negative Sampling. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '19), July 21–25, 2019, Paris, France.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3331184.3331337

1 INTRODUCTION

Distance metric learning aims at representing data points in a space where proximity accounts for similarity. A recent popular approach in face recognition [14], image retrieval [17] or speaker change detection [2] formalizes this problem as a triplet loss optimization task, namely minimizing: $L = max(D(a,p) - D(a,n) + \alpha, 0)$ where $D(a,p)$ is the distance between *intra-class* (same label) samples (anchor and positive), $D(a,n)$ is the distance between *inter-class* (different labels) samples (anchor and negative) and $\alpha > 0$ is a margin constant. The main idea is to enforce inter-class pairs to be far away from intra-class pairs at least by a margin α. This favors clustering of same class samples. As pointed out in [6, 19], minimizing L is not easy as the number of possible triplets grows cubically with the number of identities. Furthermore, a naive uniform sampling strategy would select trivial triplets for which the gradient of L is negligible. As a result, learning may be slow and stuck in a local minima [21]. To address this problem, some works proposed to select only *hard* samples ($D(a,p) > D(a,n)$) for training [15, 16]. Hard samples mining, however, selects triplets with noisy (high variance) gradients of L. Models may then struggle to effectively push inter-class pairs apart, and end up in a collapsed state [14, 21]. A relaxed alternative is to mine only *semi-hard* samples [14]: triplets in which the negative is not necessarily closer to the anchor than the positive, but which still produce a strictly positive loss. This strategy improves the robustness of training by avoiding overfitting outliers in the training set [4]. It typically converges quickly in the first iterations, but eventually runs out of informative samples and stops making progress. In [21] authors attributed this phenomenon to the concentration of the gradient's variance of L for semi-hard samples to a small region. To address this issue, they proposed to select negative samples based on their distances to anchors. They demonstrated that this strategy results in the variance of the gradient of L being spread in a larger range, and thus consistently produces informative triplets [21].

Its ability to deal with large-scale catalogs and data sparsity [19] makes the triplet loss model suitable for recommendation tasks. It has indeed been recently proposed as the CML model [7], reaching competitive results with traditional Matrix Factorization (MF) methods [13, 22]. CML assumes that users and items can be placed in a joint low dimensional metric-space. Recommendations are then easily done based on their proximity measured by their Euclidean distance. CML can achieve competitive accuracy [7] but we show in this paper that it requires large batches to do so, because of it's simplistic uniform negative sampling strategy. Owing to memory limitations, this makes CML unable to scale in high-dimensional scenarios, *e.g.*, when building a hybrid multimedia recommender system that learns jointly from interaction data and high-dimensional item contents such as audio spectrograms [9]. For that reason, following the idea in [21], we replace the default uniform sampling by a 2-stage strategy, which finds triplets that are consistently informative for learning. This enables CML to be competitive with uniform sampling, even with small batches, both in terms of accuracy and popularity bias.

SIGIR '19, July 21–25, 2019, Paris, France

ACM ISBN 978-1-4503-6172-9/19/07…$15.00
https://doi.org/10.1145/3331184.3331337

Our contributions are threefold: (1) We study the influence of batch size on the CML's performance. (2) We propose a 2-stage negative sampling that makes CML efficient with small batches. (3) We demonstrate the suitability of our sampling strategy on three real-world datasets, for the Top-N recommendation task, in terms of accuracy and popularity bias. We note especially a significant improvement over standard CML on music recommendation. We also provide code to reproduce our results[1].

2 PRELIMINARIES

2.1 Problem Formulation

Consider a dataset with N users, M items and the binary interaction $M \times N$ matrix R, where R_{ij} indicates the only positive implicit feedback (*e.g.*, clicks, listens, view histories logs etc.) between the i-th user and the j-th item. We use $\mathbf{S} = \{(i,j) \mid R_{ij} = 1\}$ to denote the set of user-item pairs where there exists implicit interactions. The considered task is to predict the items/users that are likely to interact together.

2.2 Collaborative Metric Learning

CML [7] learns a joint metric space of users and items to encode $\mathbf{S}$. The idea is to learn a metric that pulls the positive pairs in $\mathbf{S}$ closer while pushing the negative pairs (pairs not in $\mathbf{S}$) relatively further apart compared to the positive ones, based on the following loss:

$$L^{\text{triplet}} = \sum_{(i,j) \in B} w_{ij} [D^2(\mathbf{u}_i, \mathbf{v}_j) - \min_{k \in N_{ij}} D^2(\mathbf{u}_i, \mathbf{v}_k) + \alpha]_+ + \lambda_c L_c$$
$$\text{s.t. } \forall p \leq M, q \leq N : ||\mathbf{u}_p||_2 \leq 1, ||\mathbf{v}_q||_2 \leq 1 \tag{1}$$

where $\mathbf{u}_i$, $\mathbf{v}_j$ are, respectively, user and item latent vectors in $\mathbb{R}^d$, $B \subset S$ is the set of positive pairs in the considered mini-batch, $N_{ij} \subset \{k | (i,k) \notin S\}$ is a set of negative samples per triplet, $\alpha > 0$ is a margin constant, D is the Euclidean distance and w_{ij} is a weight based on the number of negatives in N_{ij} falling inside the α-ball to penalize items at a lower rank [20], $[.]_+ = max(., 0)$, L_c is regularization term (weighted by the hyper parameter λ_c) used to de-correlate the dimensions in the learned metric [7]. The recommendation for an user is then made by finding the k nearest items around her/him in the latent space.

In this work, we set w_{ij} to 1 for fair comparison between different sampling strategies. Furthermore, we do not use L_c for all models because we have inferior results for the uniform sampling with this regularization (with the code provided by authors on github[2]). Additionally, all user and item vectors are normalized to the unit sphere: $\forall p \leq M, q \leq N : ||\mathbf{u}_p||_2 = 1, ||\mathbf{v}_q||_2 = 1$ (by adding a L_2-normalization step after the user/item embedding layer) instead of being bound within the unit ball.

3 SAMPLING STRATEGY

3.1 Spread-out Regularization

In [23], the authors argued that in order to fully exploit the expressive power of the embedding, latent vectors should be sufficiently "spread-out" over the space. Intuitively, two randomly sampled non-matching vectors are "spread-out" if they are orthogonal with high probability. To this end, they proved that if $\mathbf{p}_1, \mathbf{p}_2$ are two vectors independently and uniformly sampled from the unit sphere in $\mathbb{R}^d$, the probability density of $\mathbf{p}_1^T \mathbf{p}_2$ satisfies

$$p(\mathbf{p}_1^T \mathbf{p}_2 = s) = \begin{cases} \frac{(1-s^2)^{\frac{d-1}{2}-1}}{\text{Beta}(\frac{d-1}{2}, \frac{1}{2})} & \text{if } -1 \leq s \leq 1 \\ 0 & \text{otherwise} \end{cases}$$

where Beta(a, b) is the beta distribution function. From this distribution, they further found that $\mathbb{E}\left[\mathbf{p}_1^T \mathbf{p}_2\right] = 0$ and $\mathbb{E}\left[(\mathbf{p}_1^T \mathbf{p}_2)^2\right] = \frac{1}{d}$, and proposed the Global Orthogonal Regularization (GOR) to enforce the spread of latent vectors. The application of GOR for CML is thus:

$$L = L^{\text{triplet}} + \lambda_g L^{\text{GOR}} \tag{2}$$

$$L^{\text{GOR}} = \Big(\frac{1}{Q} \sum_{(i,j) \in B} \sum_{k \in N_{ij}} \mathbf{v}_j^T \mathbf{v}_k\Big)^2 + \Big[\frac{1}{Q} \sum_{(i,j) \in B} \sum_{k \in N_{ij}} (\mathbf{v}_j^T \mathbf{v}_k)^2 - \frac{1}{d}\Big]_+ \tag{3}$$

where λ_g is an hyperparameter, $Q = |B| \times |N_{ij}|$ and d is the dimension of the latent space.

3.2 2-stage negative sampling

To construct a batch, we first randomly sample pairs in S as in [7] to get the anchor users and the positive items. Our strategy aims at replacing the uniform sampling for the set N_{ij} negative items in a triplet by a 2-stage setting as described below.

In the first stage, we sample C negative candidates from all items in the dataset based on their frequencies as proposed in the popular Word2Vec algorithm in natural language processing [11] and its application for the recommendation task [1, 3, 12]:

$$\Pr(j) = \frac{f(j)^\beta}{\sum_{j'} f(j')^\beta}, \tag{4}$$

where $f(j)$ is the interaction frequency of item j and the parameter β plays a role in sharpening or smoothing the distribution. A positive β leads to a sampling that favors popular items, a β equal to 0 leads to items being sampled uniformly, while a negative β makes unpopular items being more likely sampled. In this work, we use a positive β to favor popular items as negative samples. The motivation is that due to the popularity bias in interaction data [18], popular items tend to be close together. A challenge is thus to push non-matching popular items farther away in the latent space. Spreading popular items apart could then help to reduce the popularity bias often witnessed in recommendation.

In the second stage, we select informative negative items from the C previous candidates in a similar manner as in [21]. Given the latent vector of a positive item $\mathbf{v}_j$, we sample a negative item index n, with corresponding latent factor $\mathbf{v}_n$ as follows:

$$\Pr(n|\mathbf{v}_j) \propto \begin{cases} \frac{1}{p(\mathbf{v}_j^T \mathbf{v}_n = s)}, & 0 \leq s \leq 1 \\ 0, & \text{otherwise} \end{cases}$$

[1] Code available at: https://github.com/deezer/sigir2019-2stagesampling
[2] Original CML https://github.com/changun/CollMetric

This strategy has two objectives: first, the choice of this probability function offers triplets with a larger range of gradient's variance than what would be obtained with semi-hard triplet sampling [21]. Second, it puts high probability on items n that produce high positive value for $\mathbf{v}_j^T\mathbf{v}_n$, hence inducing positive values for L^{triplet} and large values for L^{GOR}. Indeed, it's obvious that with positive $\mathbf{v}_j^T\mathbf{v}_n$, L^{GOR} increases as $\mathbf{v}_j^T\mathbf{v}_n$ gets higher. At the same time, for each positive-negative pair $(\mathbf{v}_j, \mathbf{v}_n)$, we have $||\mathbf{v}_j - \mathbf{v}_n||_2^2 = ||\mathbf{v}_j||_2^2 + ||\mathbf{v}_n||_2^2 - 2\mathbf{v}_j^T\mathbf{v}_n = 2 - 2\mathbf{v}_j^T\mathbf{v}_n$, so the greater the value of $\mathbf{v}_j^T\mathbf{v}_n$ is, the closer the positive-negative points are. This leads to a smaller difference between $D^2(\mathbf{u}_i, \mathbf{v}_j)$ and $D^2(\mathbf{u}_i, \mathbf{v}_n)$, making L^{triplet} more likely to be positive. It thus induces higher loss values compared to the uniform sampling case, and hopefully results in gradients more suitable for training.

4 EXPERIMENTS

4.1 Experimental Settings

4.1.1 Datasets. We experiment with three datasets covering different domains: namely movie, book and music recommendations.

Amazon movies [5]: The *Amazon* dataset is the consumption records with reviews from *Amazon.com*. We use the user-movie rating from *the movies and tv category 5-core*. The data is binarized by keeping only ratings greater than 4 as implicit feedback. Users with less than 20 positive interactions are filtered out.

Book crossing [24]: The dataset contains book ratings which scale from 0 to 10 with the higher score indicating preference. Again, explicit ratings are binarized by keeping values of five or higher as implicit feedback. Only users with more than 10 interactions are then kept.

Echonest [10]: The EchoNest Taste Profile dataset contains user playcounts for songs of the Million Song Dataset (MSD). After deduplicating songs, playcount data is binarized by considering values of five or higher as implicit feedback. Finally, only users with more than 20 interactions and items with which at least 5 users interacted.

The characteristics of these three datasets after filtering are summarized in Table 1.

4.1.2 Evaluation Methodology. We divide user interactions into 4-fold for cross-validation where three folds are used to train the model and the remaining fold is used for testing. Based on the ranked preference scores, we adopt Mean Average Precision (MAP) and Normalized Discounted Cumulative Gain (NDCG) to measure whether ground-truth items are present on the ranked list of preferences truncated at 50 items and their positions. In addition, we calculate the Mean of Median Rank (MMR) of recommended items to assess the popularity bias of the model.

4.1.3 Parameters setting. The parameter C should be chosen in order to retain a sufficient number of candidates while limiting the amount of computations occurring in the second stage. We set it to 2000 and leave its optimization to future work. Besides that, the latent dimension d is set to 128 and the margin α to 1. For the other parameters, the 4-fold cross-validation mentioned above is used to choose the best values using grid-search. Adam optimizer [8] is used for all models. The learning rate is 0.0001, the parameter β for the first stage is 1.0 for Amazon movies and Echonest and 0.8 for Book crossing. Finally, λ_g is 0.01 when the number of negatives is 1 and 2 and to 0.001 as the number of negatives is 5.

Table 1: Statistics of the datasets

Dataset	User#	Item#	Rating#	Density
Amazon Movies	11181	94661	620059	0.058%
Book Crossing	3593	127339	240020	0.052%
Echonest	31521	159063	1405671	0.028%

4.2 Comparison Results

4.2.1 Uniform sampling. Performance of CML with uniform sampling [7] is summarized in Table 2 (*Uni* sub-table). We discuss results for the Amazon movies dataset as the same trend can be observed on the two others. We see that the performance of CML in terms of MAP and NDCG heavily decreases when using small batches, especially when $|N_{ij}| = 1$. For example, when the batch size is an order of magnitude smaller (256 vs 4096), MAP relatively decreases by 19% (2.26 → 1.82) and NDCG by 14% (7.55 → 6.47). This drop supports the idea that the number of informative triplets is low in small batches with the uniform sampling setup. With more negatives per triplet ($|N_{ij}| = 5$), this decrease is alleviated, about 7% relative drop against 19% for MAP (2.48 → 2.31) and 5% relative drop against 14% for NDCG (8.06 → 7.68). Additionally, another issue of CML is being prone to a strong popularity bias (MMR). As shown in Table 2 this bias increases with the batch size: e.g., from 256 to 4096, with 1 negative per triplet, MMR raises relatively by 29% (86.4 → 111.8).

4.2.2 Popularity-based sampling. To confirm our intuition on the necessity of pushing non-matching popular items farther away (as discussed in the Section 3.2), we study the popularity-based negative sampling method of Equation (4). Table 2 (*Pop* sub-table) reveals a high impact of this strategy on the performance of CML in terms of MAP & NDCG. Specifically, with smaller batch size (256 vs 4096), the MAP with popularity-based sampling already surpasses the best result of the uniform sampling, by 3.6% for movies (2.26 → 2.57), 8.7% for books (1.15 → 1.25) and 40% for music (5.71 → 8.0) respectively. As expected, the recommendations are less biased towards popular items: MMR decreases by 80% on Amazon movies (86.4 → 17.2), 90.8% on Book crossing (44.5 → 4.1) and 77% on Echonest (146.7 → 33.8).

4.2.3 2-stage sampling. While popular-based negative sampling is efficient with small batches, the reported NDCG of this strategy is slightly worse than what can be obtained by uniform sampling on large batches (except for the Echonest). In book recommendation, the gap is quite significant with a 6.3% decrease (4.13 → 3.87). To further improve the performances of the CML model on small batches, we add on top of the popularity strategy a second stage based on dot product weighted sampling as described in Section 3.2. This enables CML with small batches to have a competitive NDCG w.r.t the best result using the uniform sampling strategy. In detail, with 16 times smaller batch size, 2-stage sampling yields the same NDCG for book recommendation and reaches a 2.1% increase for movie recommendation (8.06 → 8.23), 33.6% increase for music recommendation (14.5 → 19.37). Meanwhile MAP is remarkably

Table 2: CML's performance with different sampling strategies, number of negatives per triplet and batch sizes. The format is *mean* ± *std* obtained from 4 runs on cross-validation splits. The italic bold face shows the best values for uniform strategy while bold face shows the best overall values

Sam	$\lvert N_{ij} \rvert$	Batch	Amazon Movies			Book Crossing			Echonest		
			MAP(%)	NDCG(%)	MMR	MAP(%)	NDCG(%)	MMR	MAP(%)	NDCG(%)	MMR
Uni	1	4096	2.26 ± 0.06	7.55 ± 0.12	111.8 ± 0.1	1.12 ± 0.07	3.98 ± 0.11	55.2 ± 0.4	4.88 ± 0.02	12.78 ± 0.01	241.8 ± 1.4
		1024	2.13 ± 0.08	7.19 ± 0.16	101.7 ± 1.5	1.08 ± 0.08	3.86 ± 0.14	53.2 ± 0.4	4.41 ± 0.01	11.70 ± 0.04	196.5 ± 0.6
		256	1.82 ± 0.04	6.47 ± 0.10	***86.4 ± 2.4***	0.93 ± 0.04	3.50 ± 0.06	***44.5 ± 1.2***	3.66 ± 0.03	9.90 ± 0.12	***146.7 ± 2.2***
	2	4096	2.34 ± 0.06	7.72 ± 0.10	114.5 ± 0.2	1.15 ± 0.08	4.06 ± 0.12	54.9 ± 0.9	5.16 ± 0.04	13.39 ± 0.05	265.5 ± 1.1
		1024	2.23 ± 0.04	7.49 ± 0.10	109.5 ± 0.2	1.13 ± 0.06	3.97 ± 0.13	54.5 ± 1.2	4.85 ± 0.08	12.60 ± 0.09	231.7 ± 3.0
		256	2.04 ± 0.07	6.98 ± 0.12	96.0 ± 1.9	1.04 ± 0.10	3.74 ± 0.17	49.5 ± 0.8	4.18 ± 0.06	11.14 ± 0.09	175.0 ± 5.3
	5	4096	***2.48 ± 0.05***	***8.06 ± 0.13***	118.5 ± 0.4	***1.15 ± 0.08***	***4.13 ± 0.14***	53.7 ± 0.8	***5.71 ± 0.04***	***14.50 ± 0.09***	315.5 ± 1.4
		1024	2.45 ± 0.06	8.01 ± 0.13	115.2 ± 0.6	1.15 ± 0.08	4.07 ± 0.09	55.2 ± 0.6	5.56 ± 0.05	14.17 ± 0.04	287.5 ± 2.5
		256	2.31 ± 0.02	7.68 ± 0.04	107.2 ± 0.2	1.12 ± 0.09	3.94 ± 0.16	53.5 ± 1.0	5.11 ± 0.01	13.10 ± 0.04	229.1 ± 1.0
Pop	1	256	2.12 ± 0.02	7.14 ± 0.01	26.2 ± 0.3	1.04 ± 0.07	3.54 ± 0.06	8.3 ± 0.04	6.48 ± 0.11	15.55 ± 0.17	54.3 ± 1.6
	2		2.43 ± 0.04	7.83 ± 0.12	24.3 ± 0.02	1.22 ± 0.07	3.87 ± 0.14	6.9 ± 0.13	7.26 ± 0.06	17.0 ± 0.03	46.1 ± 0.2
	5		2.57 ± 0.07	7.89 ± 0.06	**17.2 ± 0.04**	1.25 ± 0.04	3.75 ± 0.06	**4.1 ± 0.01**	8.0 ± 0.01	18.44 ± 0.08	**33.8 ± 0.3**
2st	1	256	2.32 ± 0.06	7.78 ± 0.14	32.9 ± 0.1	1.26 ± 0.08	**4.13 ± 0.13**	10.1 ± 0.1	6.99 ± 0.03	16.69 ± 0.05	96.1 ± 0.2
	2		2.57 ± 0.03	8.14 ± 0.04	27.9 ± 0.1	1.3 ± 0.08	4.12 ± 0.11	7.6 ± 0.1	7.91 ± 0.14	18.32 ± 0.13	78.9 ± 0.2
	5		**2.73 ± 0.03**	**8.23 ± 0.02**	19.6 ± 0.1	**1.38 ± 0.09**	4.12 ± 0.15	4.5 ± 0.01	**8.70 ± 0.07**	**19.37 ± 0.06**	48.4 ± 0.8

enhanced for all datasets, 10% (2.48 ⟶ 2.73), 20% (1.15 ⟶ 1.38) and 52% (5.71 ⟶ 8.7) for movie, book and music recommendation respectively. Note that 2-stage sampling makes the MMR slightly higher than that of the popularity-based strategy, but it is still significantly lower than the one with uniform sampling.

5 CONCLUSIONS

We proposed a 2-stage sampling strategy that enables the CML model to perform efficiently with batch size an order of magnitude smaller than what would be needed with the default uniform sampling. At its heart, a set of samples is first selected based on their popularity. Then, informative ones are drawn from this set based on their inner product weights with anchors. Experiments demonstrate positive results across various datasets, especially for music recommendation for which the proposed approach increased very significantly the performance of the system. In future work, we will leverage this sampling strategy to jointly learn from multimedia content and collaborative data where huge batches are prohibitive due to memory limitations.

REFERENCES

[1] Oren Barkan and Noam Koenigstein. 2016. Item2vec: neural item embedding for collaborative filtering. In *IEEE 26th International Workshop on Machine Learning for Signal Processing (MLSP)*. IEEE, 1–6.

[2] Hervé Bredin. 2017. Tristounet: triplet loss for speaker turn embedding. In *Procedding of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE.

[3] Hugo Caselles-Dupré, Florian Lesaint, and Jimena Royo-Letelier. 2017. Word2Vec applied to Recommendation: Hyperparameters Matter. In *Proceedings of the 11th ACM Conference on Recommender Systems (RecSys)*. ACM.

[4] Ben Harwood, Vijay Kumar B G, Gustavo Carneiro, Ian Reid, and Tom Drummond. 2017. Smart mining for deep metric learning. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. IEEE.

[5] Ruining He and Julian McAuley. 2016. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *Proceedings of the 25th International Conference on World Wide Web (WWW)*.

[6] Alexander Hermans, Beyer Lucas, and Leibe Bastian. 2017. In defense of the triplet loss for person re-identification. http://arxiv.org/abs/1703.07737 arXiv preprint arXiv:1703.07737.

[7] Cheng-Kang Hsieh, Longqi Yang, Yin Cui, Tsung-Yi Lin, Serge J. Belongie, and Deborah Estrin. 2017. Collaborative metric learning. In *Proceedings of the 26th International Conference on World Wide Web (WWW)*.

[8] Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. http://arxiv.org/abs/1412.6980

[9] Jongpil Lee, Kyungyun Lee, and Jiyoung Park. 2018. Deep Content-User Embedding Model for Music Recommendation. http://arxiv.org/abs/1807.06786

[10] Brian McFee, Thierry Bertin-Mahieux, Daniel P.W. Ellis, and Gert R.G. Lanckriet. 2012. The million song dataset challenge. In *Proceedings of the 21st International Conference on World Wide Web (WWW)*.

[11] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems (NIPS)*. 3111–3119.

[12] Cataldo Musto, Giovanni Semeraro, Marco De Gemmis, and Pasquale Lops. 2015. Word Embedding Techniques for Content-based Recommender Systems: An Empirical Evaluation.. In *Proceedings of the 9th ACM Conference on Recommender Systems (RecSys)*. ACM.

[13] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence (AUAI)*.

[14] Florian Schroff, Kalenichenko Dmitry, and Philbin James. 2015. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE.

[15] Abhinav Shrivastava, Abhinav Gupta, and Ross Girshick. 2016. Training region-based object detectors with online hard example mining. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE.

[16] Edgar Simo-Serra, Eduard Trulls, Luis Ferraz, Iasonas Kokkinos, Pascal Fua, and Francesc Moreno-Noguer. 2015. Discriminative learning of deep convolutional feature point descriptors. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. IEEE.

[17] Hyun Oh Song, Yu Xiang, Stefanie Jegelka, and Silvio Savarese. 2016. Deep Metric Learning via Lifted Structured Feature Embedding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE.

[18] Harald Steck. 2011. Item popularity and recommendation accuracy. In *Proceedings of the 5th ACM Conference on Recommender Systems (RecSys)*. ACM, 125–132.

[19] Chong Wang, Xue Zhang, and Xipeng Lan. 2017. How to train triplet networks with 100k identities?. In *ICCV Workshops*.

[20] Jason Weston, Samy Bengio, and Nicolas Usunier. 2010. Large scale image annotation: learning to rank with joint word-image embeddings. *Machine Learning* 81.1 (2010), 21–35.

[21] Chao-Yuan Wu, R. Manmatha, Alexander J. Smola, and Philipp Krahenbuhl. 2017. Sampling Matters in Deep Embedding Learning. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. IEEE, 2859–2867.

[22] Hu Yifan, Yehuda Koren, and Chris Volinsky. 2008. Collaborative filtering for implicit feedback datasets. In *Procedding of IEEE International Conference on Acoustics Data Mining (ICDM)*. IEEE.

[23] Xu Zhang, Felix X. Yu, Sanjiv Kumar, and Shih-Fu Chang. 2017. Learning Spread-out Local Feature Descriptors. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. IEEE.

[24] Cai-Nicolas Ziegler, Sean M. McNee, Joseph A. Konstan, and Georg Lausen. 2005. Improving recommendation lists through topic diversification. In *Proceedings of the 14th International Conference on World Wide Web (WWW)*.