RecSys Challenge 2014: an ensemble of binary classifiers and matrix factorization

Róbert Pálovics[1,2] Frederick Ayala-Gómez[1,3] Balázs Csikota[1] Bálint Daróczy[1,3]
Levente Kocsis[1,4] Dominic Spadacene[1] András A. Benczúr[1,3]
[1]Institute for Computer Science and Control, Hungarian Academy of Sciences (MTA SZTAKI)
[2]Technical University Budapest [3]Eötvös University Budapest [4]University of Szeged
{rpalovics, fayala, bcsikota, daroczyb, kocsis, domspad, benczur}@ilab.sztaki.hu

ABSTRACT

In this paper we give our solution to the RecSys Challenge 2014. In our ensemble we use (1) a mix of binary classification methods for predicting nonzero engagement, including logistic regression and SVM; (2) regression methods for directly predicting the engagement, including linear regression and gradient boosted trees; (3) matrix factorization and factorization machines over the user-movie matrix, by using user and movie features as side information. For most of the methods, we use the GraphLab Create implementation. Our current nDCG achieves 0.877.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information filtering; I.2.6 [**Artificial Intelligence**]: Learning

Keywords

Recommender systems, RecSys Challenge; GraphLab Create; Twitter

1. INTRODUCTION

The RecSys Challenge 2014 data set consists of movie ratings automatically tweeted by the IMDb application. In this unusual prediction task, for each user, the top-K tweets based on predicted engagement is requested, where engagement consists of favorites and retweets. Evaluation is based on the quality of the top list produced by the recommender. This so-called top-K recommender task is known to be hard [10]. A recent result on evaluating top-K recommenders is found in [9].

Predicting the number of retweets is a known task: [13] investigates the problem of predicting the popularity of messages as measured by the number of future retweets and [18] finds that performance is dominated by social features, but that tweet features add a substantial boost. In our work we use ideas from these papers for defining user and tweet features.

RecSys'14, October 6–10, 2014, Foster City, Silicon Valley, CA, USA.

Since movie rating tweets are defined over a pair of a user and a movie, we face a dyadic prediction task where recommender and learning-to-rank approaches may also be applicable. In our method we blend recommender methods with side information and direct predictors for the engagement, which are based on a pool of user, movie and tweet features.

Our first class of methods consists of recommenders over the data set considered as a user-movie matrix. The Netflix Prize competition [5] put recommender algorithms through a systematic evaluation on standard data [3]. The final best results blended a very large number of methods whose reproduction is out of the scope of this experiment. Among the basic recommender methods, we use matrix factorization [22, 15].

A twist in the data set over the user-movie matrix is rich side information, both for users and for movies. In addition, some users and movies appear only in the test set and hence there we face the cold start problem [20]. Some methods that use the side information include stochastic gradient descent [17] and the product of user and item kernels [4]. In our experiments we use the factorization machine [19] as a very general toolkit for expressing relations within side information.

We noticed that the most important part is to distinguish between zero and nonzero engagement. We find that the solution of this simplified version of the problem still results in a very high nDCG score of 0.986. In this sense, the task is a mix of a recommender and a binary classification tasks. We use linear models including regression and Support Vector Machines (SVM) [21].

Our final prediction relies on the ensemble of a large number of methods. Classifier ensembles are known to offer a significant improvement in prediction accuracy [23, 8, 7]. Ensembles include changing the instances used for training through techniques such as Bagging [2] and Boosting [12]. In our results, we use gradient boosted trees [26] and we combine classifiers, regressors and recommenders both by averaging and by learning their weigth by linear regression.

In most of our models, we use the GraphLab Create implementation[1] [16].

[1]`http://graphlab.com/products/create/`

Table 2: Users and movies in the training and testing sets.

	Users	Movies
Training set	22,079	13,618
Test set	5,717	4,226
Unique to training set	17,838	10,090
Unique to test set	1,476	698
Apperaing in both	4,241	3,528

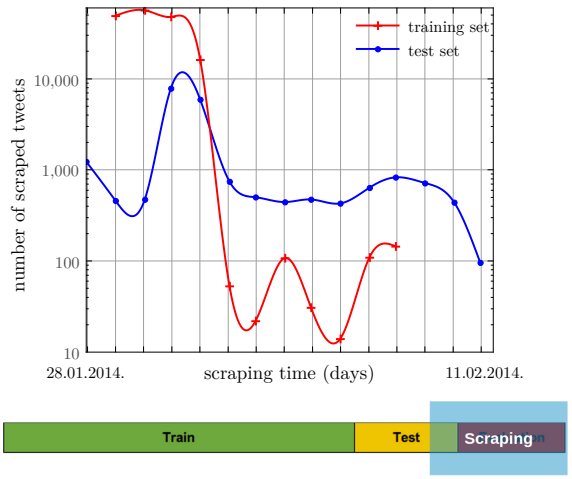

Figure 1: **Top**: Daily crawling frequencies. **Bottom**: The scraping period with respect to the training, testing and evaluation sets.

2. THE RECSYS CHALLENGE 2014 DATA SET

In this section we overview the key properties of the data set that we use for feature generation and modeling. For general information on the number and date of users, movies and tweets we refer to Table 2.

2.1 Training and test set statistics

The training and test sets greatly differ. Table 1 summarizes their properties. While in both data sets, the percentage of tweets with zero engagement is high, the ratio is significantly higher in the training set. Users may belong only to training, only to testing or to both sets (Table 2). We observe large difference between the training, testing and evaluation sets in the time elapsed between the tweet appeared and crawled due to the different timing of the training, testing and evaluation sets, as described in Fig. 1.

2.2 IMDb movie metadata

We collected IMDb related movie features from two different sources. The original MovieTweetings dataset [11] contains release year, movie genre(s) and title for each movie in the dataset. We transformed genres into a binary feature vector. Besides these features, we used the IMDBbPY Python package[2] to crawl two elementary features of the movies in the dataset: their average rating in the system, and the number of ratings they have.

[2]`http://imdbpy.sourceforge.net/`

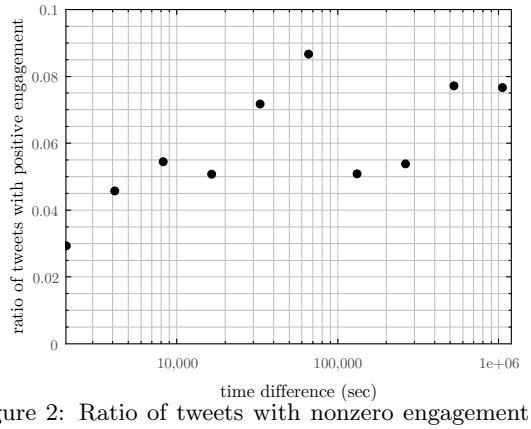

Figure 2: Ratio of tweets with nonzero engagement as the function of the time difference between the creation and scraping timestamps.

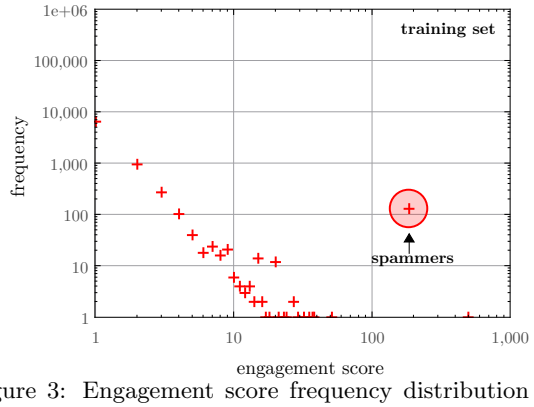

Figure 3: Engagement score frequency distribution of the training set.

2.3 Spam bots

While the frequency of engagement scores appears to follow a power-law relationship, we found within the training set a large number of tweets (130) which had an engagement score of 185, see Fig. 3. They were found to all have been a retweet of a tweet from one of Justin Bieber's friends, Ryan Butler, who also appears to have a large following. Many of the screen names of these users included variations of the name 'Justin Bieber' (e.g. 'thatkidbiebuh', 'BiebsDevotion', 'BelievingJ',...), leading us investigate celebrity posts. In fact the retweet times of such posts can occur within seconds of the post so that we term them 'spam bots'.

One strong indicator of celebrity appears to be the 'verified user' boolean, which is included in every status. In the training data, there are 16 tweets from verified users, all but 3 of which received some form of engagement. In the test data, only one such tweet exists, which too received engagement.

In light of these, we decided to use to remove the Ryan Butler-related tweets from the training data before we trained our models. Compared to the original engagement scores in Fig. 3, after cleansing, the training and testing distribution appears similar in Fig. 4.

Table 1: Training and test set properties.

	Training set	**Test set**
Number of users	22,079	5,717
Number of movies	13,618	4,226
Number of tweets	170,285	21,285
Number of zero engagement scores	162,108 (**95.19%**)	19,727 (**92.68%**)
Earliest creation time	28/02/2013 14:43	8/01/2014 22:06
Latest creation time	8/01/2014 22:06	11/02/2014 15:49
Minimum number of days between creation and scraping	**23**	**0**

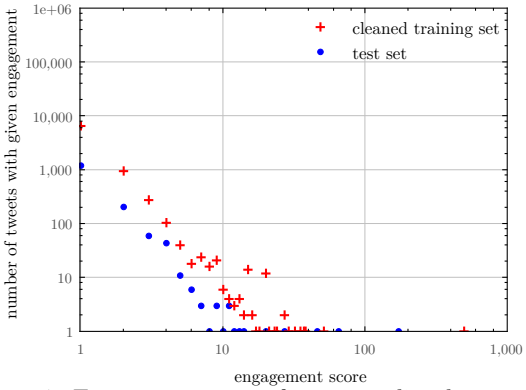

Figure 4: Engagement score frequency distributions in the cleansed training set and the test set.

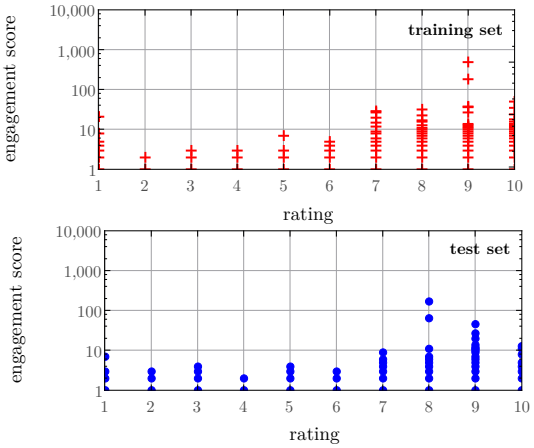

Figure 5: Engagement score as the function of rating for the training set (**top**), and for the test set (**bottom**).

Table 3: Properties of the cleansed training set.

Number of users	21,950
Number of movies	13,618
Number of tweets	170,155
Number of tweets with zero engagement score	162,107 (**95.27%**)

2.4 Extreme ratings

In Fig. 5 a U-shaped relationship can be made out within both datasets. In other words, tweets with extremist ratings generate higher engagement scores in average. In fact in terms of linear regressions based on only one feature, we found that rating performed the best, achieving as high as 0.818 nDCG on the test set. However, we found tweets containing ratings that fell outside of the "standard" range determined by the IMDb tweet template (1-10, inclusive). In the training set, 73 ratings were given outside of this range. The higher "extreme rating" tweets also more often receive engagement than regular tweets.

2.5 Cleansed training set

Table 3 summarizes the properties of our cleansed training dataset. We removed tweets with rating larger than 10 or less than 1. We also excluded the spammers from the dataset.

2.6 Features

Here we list the features we extracted from the original datasets.

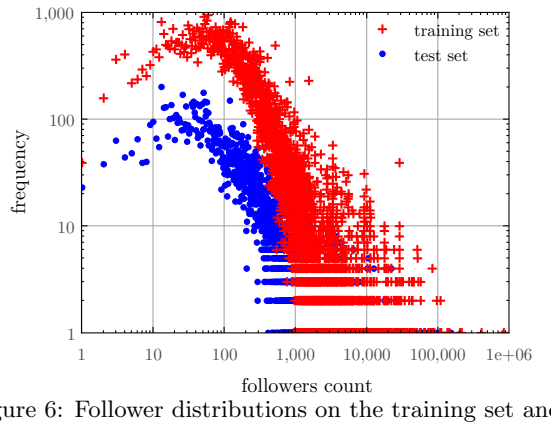

Figure 6: Follower distributions on the training set and the test set.

User features extracted from the Twitter json data: creation time, followers and friends count (see Fig. 6, favourites count, statuses count

Tweet features extracted from the Twitter json data: creation time, scraping time, time difference between creation and scraping, number of hashtags / URLs / mention in the tweet, tweet is retweet, tweet has retweet in the dataset

Table 4: Best nDCG@10 results of the invidual algorithms, separate for resubstitution over the training set and for the test set. In both cases we give the best result not using and using retweet information ("no RT" and "with RT", respectively).

Method		**Training set**		**Test set**	
		no RT	with RT	no RT	with RT
Random prediction	(RN)	≈ 0.61	-	≈ 0.75	-
Retweet based prediction	(RT)	-	0.73038	-	0.80098
Rating based prediction	(RA)	0.69386	0.75670	0.82125	0.84980
Graphlab Create Item Means Model	(IM)	0.74821	0.80729	0.79957	0.84340
Graphlab Create Popularity Model	(POP)	0.75935	0.81684	0.79467	0.83862
SVM	(SVM)	0.63976	0.73112	0.81850	0.86777
Graphlab Create Gradient Boosted Tree	(GCGT)	0.71009	0.79220	0.83336	0.87127
Graphlab Create Linear Regression	(GCLR)	0.67782	0.75374	0.82726	0.86089
Graphlab Create Logistic Regression	(GCGR)	0.67724	0.75270	0.82807	0.86078
Graphlab Create Matrix Factorization	(GCMF)	0.70343	0.77598	0.82444	0.86291
Graphlab Create LibFM based recommender	(GCFM)	0.68005	0.75878	0.82019	0.84139
NDCGboost	(NB)	0.68663	0.77032	0.80532	0.86282

Movie features extracted from the MovieTweetings dataset and our IMDb crawl:
movie average rating, number of ratings on IMDb, binary vector of movie genres from the movieTweetings dataset.

Rating based features: Besides the original user rating, we take the difference of the user's rating and the movie's average rating on IMDb. We also give a score for each tweet based on its extremeness (see 2.4). Moreover, we have created feature vectors from the ratings. We use the average engagement value as the function of the rating (1–10) as shown in Fig. 5.

2.7 Zero and nonzero engagement

In order to simplify the task, we considered separating those tweets that received engagement from those that did not. Assuming that this binary classification task could be perfectly solved, we measured an nDCG@10 of 0.988 over the test set. The fact that the actual engagement count is less important compared to separating zero from nonzero is in part due the shear number of tweets with no engagement (see Table 1), as well as the fact that most of the users have no more than 5 tweets: in the training data, 7,012 of the 22,079 users have 5 or more tweets. Note that of the 7012 users in the training set with more than 5 tweets, only 2285 of them have at least one tweet with nonzero engagement. The advantage of the zero-nonzero task is that binary classifiers are applicable, some of which giving the strongest performance with respect to nDCG, see Table 4.

2.8 Information leakage through retweets

Retweets of movie rating tweets were also fetched by the Twitter API. A retweeted message has by definition nonzero engagement. In addition, the retweet of a message received the engagement score of the original message, possibly as a side effect of how the Twitter API gives information on retweets. Moreover, if we see a retweet in the dataset, if its root tweet is in our dataset, we immediately know that the root tweet's engagement score is higher than 0. We use retweet information in two ways. First, we may include the fact that a retweet exists as a binary feature. Second, since retweets and retweeted messages have nonzero engagement, we increased the predicted score by a large constant for these tweets. For reference, we included the performance of all methods without using the retweet information in any way.

3. THE EVALUATION METRIC

Recommender systems in practice need to rank the best K items for the user. In this top-K recommendation task [10, 9] the goal is not to rate some of the individual items but to provide the best candidates. Despite the fact that only prediction for the top list matters in top-K evaluation, several authors propose models trained for RMSE or AUC with good top-K performance [14, 25] and hence we follow their approach.

The RecSys Challenge 2014 task is evaluated by nDCG, one of the most common measures for top-K recommender evaluation [1]. Note that we train our binary classifiers optimized for RMSE or AUC, both evaluated as a macro measure by globally ranking all tweets. The challenge evaluation, in contrast, defines a user micro average.

4. ELEMENTS OF OUR ENSEMBLE AND EXPERIMENTAL RESULTS

We describe two groups of methods. The first one is based on binary classification and regression in Section 4.2. The details of the features and parameters used in each model are described in Table 5. The second one in Section 4.3 is an online matrix factorization. The performance overview is found in Table 4 both for resubstitution on the training set and for the testing set. We show results not using any retweet based information separate. The columns that use retweet information include the best performing combination of using retweet as a binary feature for training and moving retweets and their root tweets ahead in the ranking. The final blending results are in Table 6.

4.1 Baseline measurements

We defined five nDCG baselines for both the training and testing set to benchmark the result of our models. The first is the random prediction (RN) for the models that does not use the retweet features. This method randomly sorts the tweets for each user and then calculates the nDCG. The second is the retweet based prediction (RT) for the models that uses retweet features. This model gives score 1 to tweets that are retweets or root tweets, and score 0 to all other

Table 5: Main parameters of our algorithms.

Method	Parameters
GCGT	num_trees: 18, step_size: 0.3, max_depth: 4, num_iterations: 18, min_child_weight: 0.1, min_loss_reduction: 0
GCLR	L2_penalty: .10, solver: newton, max_iter: 10
GCGR	L2_penalty: .10, solver: newton, max_iter: 10
GCMF	n_factors: 8 linear_regularization: 0, regularization: 0.0001
GCFM	n_factors: 8 linear_regularization: 0, regularization: 0.0001
SVM with RT Features	C : 0.5, kernel: linear
SVM no RT Features	C : 0.25, kernel: polynomial, degree: 2
NB	num_trees : 20, num_leaves: 8

Table 6: Best blended nDCG@10 results.

Blending Method	Test set	
	no RT features	with RT features
Best combination achieved by grid search	0.83922	0.87713
Average of RA, GCGT, GCGR	0.38973	0.87340
Scikit-learn Linear Regression	0.84027	0.87435

tweets. Our rating based prediction (RA) uses the U-shape based extremeness to predict the ranking of the tweets. We also used the popularity model and the item means model in Graphlab Create as baseline measures. The values of the baselines are shown in Table 4. It turns out that the nDCG changes because of the different properties of each dataset (e.g. engagement frequency, users engaged.).

4.2 Binary classification

In order to apply the binary classification methods, we created a binary feature that expresses if a tweet had engagement or not. We applied linear regression, logistic linear regression, Gradient Boosting Trees [26] and SVM [21].

For both linear regression (GCLR) and logistic linear regression (GCGR), the Newton method and stochastic sradient descent (SGD) solvers were used. However, the Newton method solver led to a better nDCG@10 than SGD. The features were rescaled using the L2-norm to ensure that the features have the same norm. The strongest three features were the rating, the difference of the rating and the movie's average rating in IMDb, and the number of personal mentions in the tweet. Note that one can edit the tweet generated by the IMDb application. If someone includes a personal mention in a tweet, it has higher probability to receive engagement in the future.

In case of the gradient boosted tree algoritm (GCGT) we set the maximum depth of the trees 4, and enabled maximum 18 iterations. Note that this algorithm performed the best on the test set even with and without using the retweet features.

By using support vector machine (SVM) we were able to achieve our second highest nDCG@10. We could observe that normalization, parameters and kernel selection are a very important step. Because of the different scale of the features, we scaled them linearly between zero and one except for the rating and retweet features. Our main reason was to gain advantage of the known relevance of these features.

4.3 Matrix factorization

We used two different matrix factorization techniques that are both implemented in GraphLab Create. The first one is a traditional matrix factorization method [15], while the second one is Steffen Rendle's LibFM algorithm [19]. Both techniques use stochastic gradient descent to optimize for mean-square error on the training set. The difference between the methods is that LibFM uses the side information of the users and items more sophisticatedly. Therefore we used the original matrix factorization technique without any side information, and used LibFM for user and item feature included collaborative filtering prediction. Note that in both cases we used movies instead of tweets as items, as each tweet (excluding the few retweets) is unique in the data.

4.4 Ranking based approach

NDCGboost [24] (NB) is a decision tree boosting algorithm that optimizes the expectation of NDCG over all possible permutations of items. We usend NDCFboost as one algorithm in our ensemble. The models included twenty trees with 8 leaves each.

4.5 Blending

We combined our methods linearly to yield the final prediction. Our methods reach close to the best possible combination weights that we obtained by grid search over the testing set as seen in Table 6. In the simplest method, we average the well performing methods. We also learn the weights by linear regression. Here we used the implementation of scikit-learn[3]

Conclusions

The RecSys Challenge 2014 task for predicting engagement of movie rating tweets has turned out to be a mix of Twitter activity prediction and user-movie top-K recommendation. For the activity prediction component of the task, classification and regression methods have worked well. And for top-k recommendation, we have used dyadic classifiers, variants of recommender methods that may use side information. As different classes of methods model different views

[3]http://scikit-learn.org/

of the data, they have combined well in our final ensemble. Due to the variety of user, movie and tweet side information, data collection and cleansing issues have played an important role. We have collected IMDb movie metadata, removed Twitter spam, and noticed an information leak for retweet information that was probably unintentionally collected for the challenge data set.

Acknowledgments

The publication was supported by the KTIA_AIK_12-1-2013-0037 and the PIAC_13-1-2013-0205 projects. The projects are supported by Hungarian Government, managed by the National Development Agency, and financed by the Research and Technology Innovation Fund. Work conducted at the Institute for Computer Science and Control, Hungarian Academy of Sciences (MTA SZTAKI) was supported in part by the EC FET Open project "New tools and algorithms for directed network analysis" (NADINE No 288956), by the Momentum Grant of the Hungarian Academy of Sciences, and by OTKA NK 105645. Work conducted at the Technical University Budapest has been developed in the framework of the project "Talent care and cultivation in the scientific workshops of BME" project. This project is supported by the grant TÁMOP - 4.2.2.B-10/1–2010-0009. Work conducted at University of Szeged was partially supported by the European Union and the European Social Fund through project FuturICT.hu (grant no.: TAMOP-4.2.2.C-11/1/KONV-2012-0013).

5. REFERENCES

[1] A. Al-Maskari, M. Sanderson, and P. Clough. The relationship between ir effectiveness measures and user satisfaction. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 773–774. ACM, 2007.

[2] E. Bauer and R. Kohavi. An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *Machine learning*, 36(1-2):105–139, 1999.

[3] R. M. Bell and Y. Koren. Lessons from the netflix prize challenge. *ACM SIGKDD Explorations Newsletter*, 9(2):75–79, 2007.

[4] A. Ben-Hur and W. S. Noble. Kernel methods for predicting protein–protein interactions. *Bioinformatics*, 21(suppl 1):i38–i46, 2005.

[5] J. Bennett and S. Lanning. The netflix prize. In *KDD Cup and Workshop in conjunction with KDD 2007*, 2007.

[6] C. Burges, R. Ragno, and Q. Le. Learning to rank with nonsmooth cost functions. *Advances in neural information processing systems*, 19:193, 2007.

[7] R. Caruana, A. Munson, and A. Niculescu-Mizil. Getting the most out of ensemble selection. In *ICDM '06: Proceedings of the Sixth International Conference on Data Mining*, pages 828–833, Washington, DC, USA, 2006. IEEE Computer Society.

[8] R. Caruana, A. Niculescu-Mizil, G. Crew, and A. Ksikes. Ensemble selection from libraries of models. In *ICML '04: Proceedings of the twenty-first international conference on Machine learning*, page 18, New York, NY, USA, 2004. ACM.

[9] P. Cremonesi, Y. Koren, and R. Turrin. Performance of recommender algorithms on top-n recommendation tasks. In *Proceedings of the fourth ACM conference on Recommender systems*, pages 39–46. ACM, 2010.

[10] M. Deshpande and G. Karypis. Item-based top-n recommendation algorithms. *ACM Transactions on Information Systems (TOIS)*, 22(1):143–177, 2004.

[11] S. Dooms, T. De Pessemier, and L. Martens. Movietweetings: a movie rating dataset collected from twitter. In *Workshop on Crowdsourcing and Human Computation for Recommender Systems, CrowdRec at RecSys 2013*, 2013.

[12] Y. Freund, R. E. Schapire, et al. Experiments with a new boosting algorithm. In *ICML*, volume 96, pages 148–156, 1996.

[13] L. Hong, O. Dan, and B. D. Davison. Predicting popular messages in twitter. In *Proceedings of the 20th International Conference Companion on World Wide Web*, WWW '11, pages 57–58, New York, NY, USA, 2011. ACM.

[14] Y. Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 426–434. ACM, 2008.

[15] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

[16] Y. Low, D. Bickson, J. Gonzalez, C. Guestrin, A. Kyrola, and J. M. Hellerstein. Distributed graphlab: a framework for machine learning and data mining in the cloud. *Proceedings of the VLDB Endowment*, 5(8):716–727, 2012.

[17] A. K. Menon and C. Elkan. A log-linear model with latent features for dyadic prediction. In *Data Mining (ICDM), 2010 IEEE 10th International Conference on*, pages 364–373. IEEE, 2010.

[18] S. Petrovic, M. Osborne, and V. Lavrenko. Rt to win! predicting message propagation in twitter. In *ICWSM*, 2011.

[19] S. Rendle, Z. Gantner, C. Freudenthaler, and L. Schmidt-Thieme. Fast context-aware recommendations with factorization machines. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, pages 635–644. ACM, 2011.

[20] A. I. Schein, A. Popescul, L. H. Ungar, and D. M. Pennock. Methods and metrics for cold-start recommendations. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 253–260. ACM, 2002.

[21] J. Shawe-Taylor and N. Cristianini. Kernel methods for pattern analysis. *Journal of the American Statistical Association*, 101:1730–1730, December 2006.

[22] G. Takács, I. Pilászy, B. Németh, and D. Tikk. Investigation of various matrix factorization methods for large recommender systems. In *Proceedings of the 2nd KDD Workshop on Large-Scale Recommender Systems and the Netflix Prize Competition*, pages 1–8. ACM, 2008.

[23] K. Tumer and J. Ghosh. Error correlation and error reduction in ensemble classifiers. *Connection science*, 8(3-4):385–404, 1996.

[24] H. Valizadegan, R. Jin, R. Zhang, and J. Mao. Learning to rank by optimizing ndcg measure. In Y. Bengio, D. Schuurmans, J. D. Lafferty, C. K. I. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems (NIPS) 22*, pages 1883–1891, 2009.

[25] M. Weimer, A. Karatzoglou, and A. Smola. Adaptive collaborative filtering. In *Proceedings of the 2008 ACM conference on Recommender systems*, pages 275–282. ACM New York, NY, USA, 2008.

[26] Z. Zheng, H. Zha, T. Zhang, O. Chapelle, K. Chen, and G. Sun. A general boosting method and its application to learning ranking functions for web search. In *Advances in neural information processing systems*, pages 1697–1704, 2008.