Improving Movie Recommendations with Domain-Specific Insights

Rohan Kumar
Department of Computer Science
University of Chicago
Chicago, IL 60637
rohankumar@cs.uchicago.edu

Arjun Rawal
Department of Computer Science
University of Chicago
Chicago, IL 60637
arjunrawal4@cs.uchicago.edu

Abstract

The prediction of user preference is an essential component of modern systems, as better predictions lead to increased customer satisfaction and revenue. However, these predictions are often inaccurate, require large quantities of training data, and have to be run on high performance computers to achieve satisfactory results. In this study, we propose a hybrid prediction approach, where we combine domain-specific information with standard recommendation techniques to achieve more accurate predictions. We propose simple modifications to LMaFit based matrix completion techniques which incorporate genre information, and find that these do not provide any significant improvements. We also propose a modified collaborative filtering technique using a hybrid similarity metric that accounts for users liking the same movie-genres. On MovieLens 1M, our proposed collaborative filtering technique achieves up to 50% smaller prediction error with highly sparse ratings compared to standard collaborative filtering. This result demonstrates that applying relevant domain-specific information to recommendation problems can provide more accurate results, with a minimal effect on training time.

1 Background and Related Work

Movie recommendations provide a perfect test scenario for creating accurate recommendations based on user history. Movie rating data is fairly sparse, as people watch different sets of movies, and have a variety of criteria that they use to evaluate the movies. We first describe the state of the art in matrix completion and collaborative filtering, and then introduce our approach, which combines the domain specific genre information to increase prediction accuracy.

We use work done by Tran et al. [2018] as an inspiration for our approach, as they use the existence of movies that are *co-liked* and *co-disliked* to recommend movies to a user who has seen and liked one of a pair. However, this approach requires a large set of training data, as it fails to provide satisfactory predictions when there are very few overlaps between users. Another approach by Wu et al. [2018] takes into account that some users' recommendations are less reliable than others for predicting future ratings. They analyze the deviation of each user from an average rating, and use that centrality data to create a better matrix factorization. This approach is more accurate than many other methods, but requires many iterations to achieve a proper gradient descent result, and therefore cannot be used in more computation limited circumstances.

Several variations of collaborative filtering have proven successful for recommendation systems. From a survey by Su and Khoshgoftaar [2009], the canonical approaches are model-based and memory-based collaborative filtering, as well as hybrid approaches which build on these. Perhaps the most significant drawback of model-based collaborative filtering is its inability to take advantage of item-similarity (e.g., 2 different comedy movies). Thus, a common strategy to improve the model-

Preprint. Under review.

based approach is to create hybrid approaches which incorporate this additional information about the items being recommended.

Furthermore, collaborative filtering performs worse when the ratings matrices in concern are more sparse (i.e., when there are fewer ratings to train on). Kohrs and Merialdo [1999] present a hierarchical clustering approach which improves on the robustness of collaborative filtering in this case. Towards a similar goal, Gong [2010] presents user-based and item-based clustering strategies which are both improvements over naive the collaborative filtering approach. Inspired by this goal, in this work, we incorporate information about our items (in particular, genres) to improve predictions for sparse ratings.

2 Problem Statement

We explore modifications to the naive collaborative filtering approach within the context of providing movie recommendations. We compare and contrast standard techniques like memory-based collaborative filtering and low-rank matrix completion. Furthermore, we improve on these canonical techniques by using hybrid methods and taking into account additional context about the movie-recommendation problem, such as categorizing movies into genres, and finding a user's top genres.

3 Approach

3.1 Dataset

We train and evaluate our recommendations on the MovieLens 1M dataset [1]. This dataset from Harper and Konstan [2016] consists of a matrix of (u, m) ratings, for 6040 MovieLens users on a set of 3883 movies. Each rating is an integer between 1 and 5, and the movie descriptions include a genre classification from IMDB which can fall into 17 different categories (movies can have multiple genres). Other user metadata and movie features are provided, but not used in this experiment. The sparsity of the ratings matrix is 4.26%.

Notation: For the remainder of this paper, R will denote the true ratings matrix and R' will denote the ratings matrix we train on (more details in Section 4.1). R_{um} denotes user u's rating for movie m.

3.2 Collaborative Filtering

Simple Similarity Metrics: This is the standard model-based collaborative filtering. For n users, we first compute an $n \times n$ similarity matrix S, where S_{ij} represents the similarity between users i and j. Then, in order to compute user u's predicted rating for movie m, we compute:

$$r_{sim}(u, m) = \frac{1}{\sum_{v \in \mathrm{U}} |S_{uv}|} \sum_{v \in \mathrm{U}} S_{uv} \cdot R'_{vm}$$

We experimented with several similarity metrics, including cosine, correlation, hamming, and euclidean. We present results in a later section for cosine and hamming-distance based metrics, since they both perform well but in different ways.

Genre-Only Prediction: This is purely content-based collaborative filtering. We look at a user's top-k genres and see how much they overlap with a movie's genres. More precisely:

$$r_{gen}(u, m) = r_{avg} + \lambda \cdot \mathrm{IOU}(u\text{'s top-}k \text{ genres}, m\text{'s genres})$$

Here, r_{avg} is the average rating in R', and IOU means "intersection-over-union". λ and k are hyper-parameters which can be chosen with care.

Simple Similarity with Genre Bias: In this hybrid approach, we simply add a bias term if a user's top-k genres overlap with the movie's genres, and subtract a bias term otherwise. k and b are hyper-parameters.

$$r_{gen-bias}(u, m) = \begin{cases} r_{sim}(u, m) + b & \text{if genre\_overlap}_k(u, m) \neq \varnothing \\ r_{sim}(u, m) - b & \text{if genre\_overlap}_k(u, m) = \varnothing \end{cases}$$

[1] https://grouplens.org/datasets/movielens/

Simple Similarity with Genre Similarity: Here, we incorporate the genre-similarity of users, as well as their standard movie-similarity. The intent is to capture the fact that a comedy-liker is well-predicted by other comedy-likers, even if she has not watched the same comedy movies as them.

$$G_{uv} = S_{uv} + \lambda \cdot |\{u\text{'s top-}k \text{ genres}\} \cap \{v\text{'s top-}k \text{ genres}\}|$$

$$r_{gen-sim}(u,m) = \frac{1}{\sum_{v \in \mathrm{U}} |G_{uv}|} \sum_{v \in \mathrm{U}} G_{uv} \cdot R'_{vm}$$

λ and k are hyper-parameters.

3.3 Matrix Completion

Averaged Ratings Approach: We first consider the naive approach: we take the average of the nonzero ratings for a movie m_{avg}, and then compare the user's average rating u_{avg}, with the overall average rating U_{avg}, and predict the rating for (u, m) will be the average rating for a movie scaled by a users individual rating average. Effectively, we find the average rating of a movie, and then bias that rating by the user's personal rating average, as some people rate movies higher than others on average.

$$r_{avg-comp}(u,m) = m_{\text{avg}} * \frac{U_{\text{avg}}}{u_{\text{avg}}}$$

LMaFit: We utilize the LMaFit algorithm, developed by Wen et al. [2012]. This algorithm takes in a set of existing indices E and corresponding values, as well as a predicted rank, and then uses a minimization algorithm to solve for the remaining values. This corresponds to solving the problem

$$\text{Find } U \in \mathbb{R}^{m \times n}(k) \text{ s.t. } U_{ij} \approx A_{ij}, (i,j) \in E$$

ISVT: We also consider the Iterative Singular Value Thresholding Algorithm developed by Cai et al. [2010]. This approach, iteratively produces a completed matrix $\hat{X}$ by removing small singular values below a certain threshold in a Singular Value Decomposition, and then uses the truncated singular values to create a new matrix. This process is repeated until the change in the matrix X is below a value ε.

LMaFit with Genre Bias: We consider varying the predicted rank for LMaFit, and then also consider biasing results based on genre preferences. After computing the LMaFit matrix completion, we iterate through our testing data, and bias our results up or down if the set of genres for a movie intersect the set of top k genres for a user is nonempty. We compute the set of top k genres for a user by keeping track of the genres for all movies they rank higher than their average rating over all movies. We use the set intersection of genre sets with the intuition that if a movie falls into a user's top k genres, they are more likely to enjoy watching it. We explored options of weighting the user genre preferences and more complex similarity functions, but found little to no benefit.

$$r_{lma-bias}(u,m) = \begin{cases} r_{lma}(u,m) + b & \text{if genre\_overlap}_k(u,m) \neq \varnothing \\ r_{lma}(u,m) - b & \text{if genre\_overlap}_k(u,m) = \varnothing \end{cases}$$

4 Evaluation

4.1 Methodology

In order to test each of our approaches, we used hold-out sets as follows. Let Ω be the set of indices we have ratings for, i.e., $\Omega = \{(u,m) : R_{um} \text{ is valid}\}$. For a testing fraction $p \in [0,1]$, we choose a uniformly random subset of Ω of size $p \cdot |\Omega|$ — this is our test subset, Γ. We initialize a training matrix R' and for all $(u,m) \in \Omega \setminus \Gamma$, we set $R'_{um} = R_{um}$. In words, we omit the values we are going to test on.

We then run the approach we are testing on the (training) ratings matrix R' to learn and fill in the missing ratings. To measure performance, we compute the Root-Mean-Squared-Error over all

predictions in our test set, Γ, i.e.,

$$\text{error} = \sqrt{\frac{1}{|\Gamma|} \sum_{(u,m)\in\Gamma} ||R'_{um} - R_{um}||_2^2}$$

All of our code can be found on GitHub.[2]. We use a Python3 version of LMaFit which can be found on GitHub.[3]

4.2 Collaborative filtering

4.2.1 Naive Strategies

We evaluate the naive model-based collaborative filtering strategy with two different but well-performing similarity metrics — cosine similarity and hamming-distance similarity. These results are shown in Figure 1. Both strategies give an RMSE of slightly less than 1, which means the predicted ratings are, on average, 1 away from the true ratings. Hamming-similarity seems to perform much better than cosine-similarity in the case of more sparse matrices (i.e., when we train on $\leq 5\%$ of available ratings). These values are commensurate to results seen in the collaborative filtering literature.

4.2.2 Using Genre Information

The Genre-Only Prediction does not perform well, giving RMSE > 1.15, even with optimized hyper-parameters, implying the need for hybrid strategies which account for both model-based similarity (as above) and movie-specific information (genres). For both the Genre-Bias and Genre-Similarity approaches, we found that $k = 3$ gives us the best results; thus, all results shown use $k = 3$. The Genre-Bias approach does not yield improvements over the standard approach, and in fact, seems to increase the error rate with larger bias values (Figures 2 and 3). This hints that adding or subtracting a constant bias term is too naive.

On the other hand, the Genre-Similarity approach provides improvements over the standard approach in the case of sparse matrices, especially in the case of cosine similarity (Figures 4 and 5). When training on 1% of the available ratings, the naive approach gets an RMSE of 2.6 whereas the Genre-Similarity approach gets between 1.3 and 1.4, depending on the scaling factor used. This result further verifies that collaborative filtering does not perform well with a small number of training samples. In these cases, using information about different users liking the same genres (even if not the same movies) improves errors rates by up to 50%.

4.3 Matrix completion

4.3.1 Methods

We evaluate the effectiveness of the various matrix completion approaches using the standard methodology mentioned above. In Figure 6, we show the effectiveness of the ISVT, Avg, and, LMaFit approaches. We see that LMaFit and Avg both perform similarly, with Avg actually outperforming the LMaFit approach at very high sparsity (1% training set). ISVT does very poorly, by inspection, it appears that the values it fills are too close together to be accurate ratings.

4.3.2 LMaFit Ranks

We explore the effect of predicted rank for LMaFit and see that the lower ranks generally perform better, with rank=1 or rank=2 being the best in all cases. We use rank=2 for the other experiments, as it is commonly seen in literature (Yu et al. [2016]) and performs very similarly to the optimal based on our experiments. We present the rank comparisons in Figure 7.

[2]https://github.com/rohankumar42/movie_recommendations

[3]https://github.com/mcrovella/mining-low-dim-network-data/blob/master/lmafit.py

4.3.3 Genre Bias

We evaluate the benefit of adding genre bias to the LMaFit based predictions, and find, surprisingly, that they do not improve prediction accuracy. We use a standard set intersection to determine an overlap between movie and user genre preferences, and only look at the user's top 3 movies. This parameter was fixed after exploring values in the range [1,8]. We add the bias value to our predictions if the movie and user genre's overlap, and subtract the bias value otherwise. We plot the effect of various levels in Figure 8. In our experiments, we also considered the effect and benefit of normalizing the ratings before and after the addition of bias, and found no benefit. Additionally, we considered other approaches to including bias, such as only using positive bias, or using multiplicative bias instead of a constant offset. However, in all of these cases, we found that the predictions did not improve significantly.

As evident from Figure 8, we see that the benefit of adding bias decreases both as the bias increases, and as p increases. By inspection, we see that when LMaFit is run with very little data, LMaFit completes the matrix with very small values, and so predictions are very inaccurate. Consequently, the bias values do little to fix the inaccuracies in the predictions. In all cases, we find the the LMaFit predictions seem to off by similar amounts on both movies in the user's genre preference and on movies not in the user's preferences. One explanation for this is the fact that users are unlikely to watch many movies not in their preference set, so such adjustments are not very beneficial to overall error reduction. Furthermore, because LMaFit, unlike collaborative filtering, does not focus on user similarity, perhaps some of the underlying genre structure is already present in the completed matrix, and increasing the bias just creates inaccuracies.

5 Conclusion

In this work, we implement and evaluate several standard approaches to both collaborative filtering and matrix completion, with the goal of producing accurate movie recommendations. However, we notice that these approaches lack insight about the nature of the data, specifically genre preference. We incorporate the genre information about both movies and users into our collaborative filtering and matrix completion predictions with varying success. Although the genre bias does not seem to provide a benefit on matrix completion, it improves collaborative filtering predictions on sparse data by up to 50%. These results demonstrate that in certain cases, using domain specific information to create a hybrid approach can vastly outperform traditional collaborative filtering approaches.

Acknowledgments

We thank Tapan Srivastava and Andrew Wells for their helpful insight and moral support in the research process.

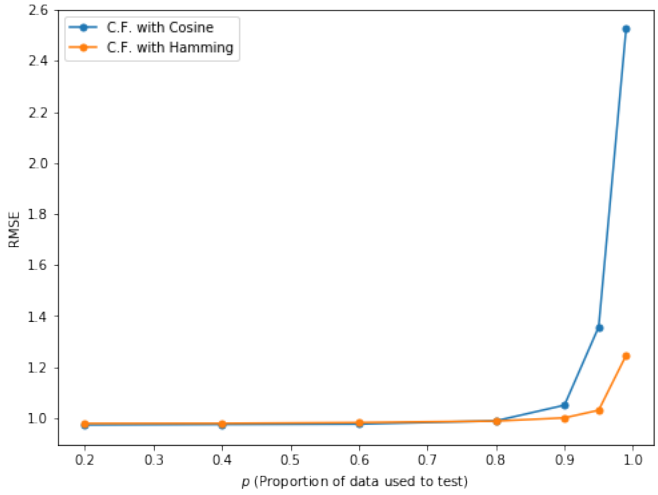

Figure 1: Naive collaborative filtering with cosine and hamming similarity.

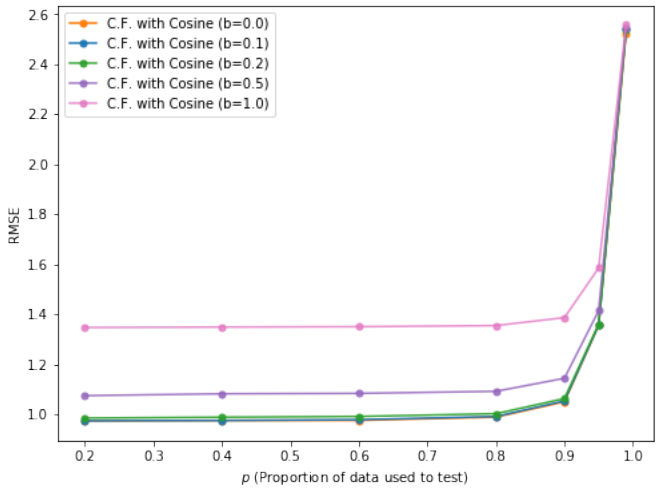

Figure 2: Collaborative filtering (cosine) with genre-bias.

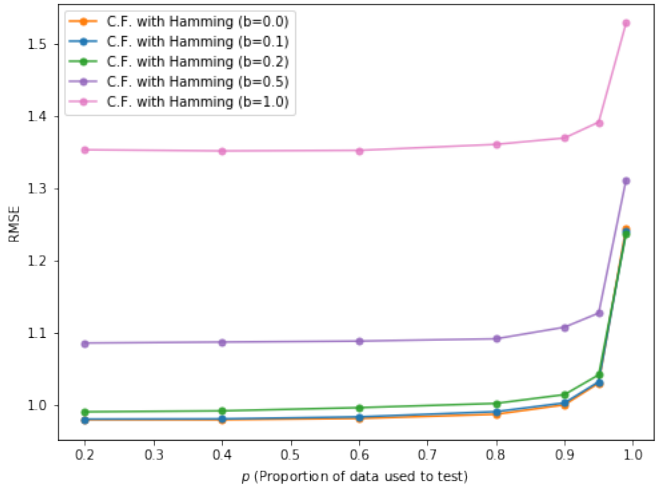

Figure 3: Collaborative filtering (hamming) with genre-bias.

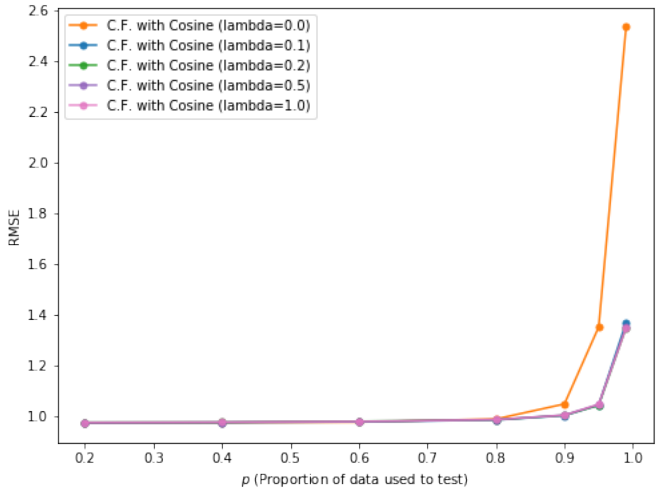

Figure 4: Collaborative filtering (cosine) with additional genre-similarity.

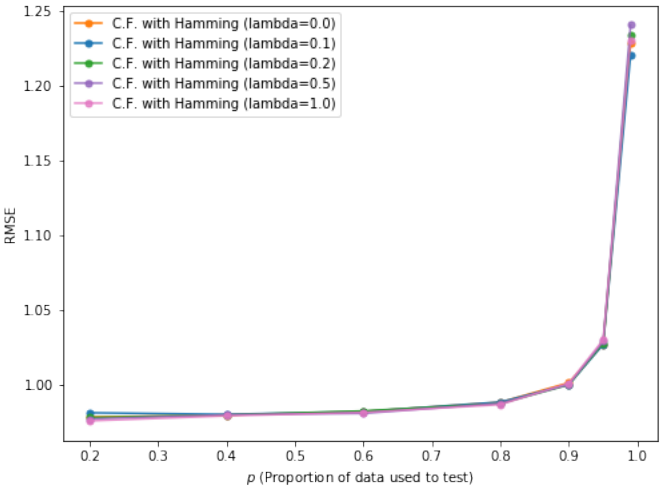

Figure 5: Collaborative filtering (hamming) with additional genre-similarity.

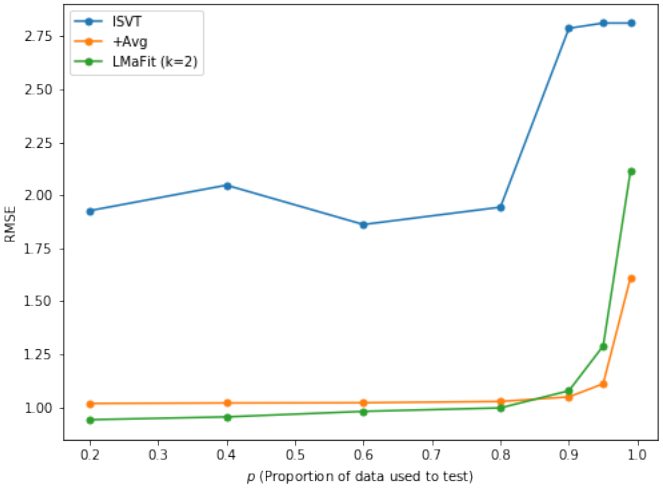

Figure 6: Comparison of matrix completion methods.

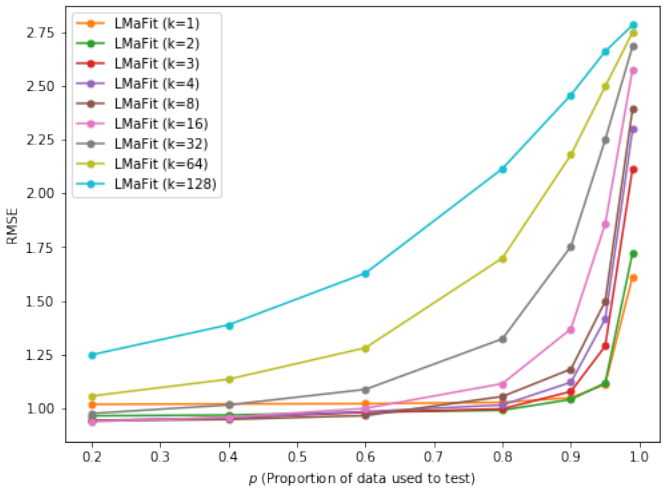

Figure 7: Comparison of LMaFit accuracy with varying rank.

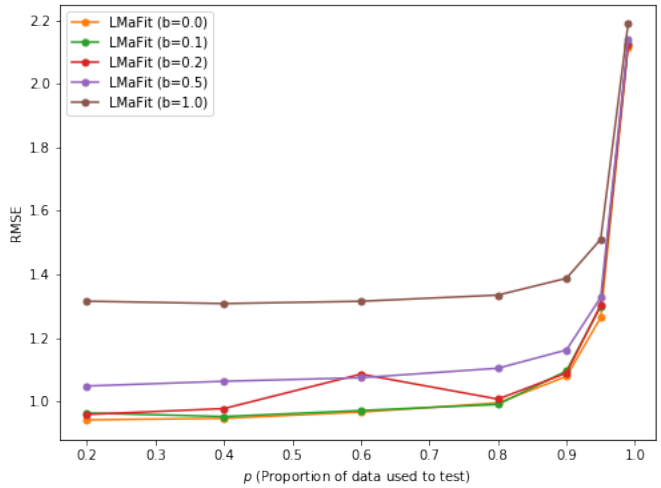

Figure 8: Comparison of LMaFit with genre bias.

References

J.-F. Cai, E. J. Candès, and Z. Shen. A singular value thresholding algorithm for matrix completion. *SIAM Journal on optimization*, 20(4):1956–1982, 2010.

S. Gong. A collaborative filtering recommendation algorithm based on user clustering and item clustering. *JSW*, 5(7):745–752, 2010.

F. M. Harper and J. A. Konstan. The movielens datasets: History and context. *Acm transactions on interactive intelligent systems (tiis)*, 5(4):19, 2016.

A. Kohrs and B. Merialdo. Clustering for collaborative filtering applications. *Intelligent Image Processing, Data Analysis & Information Retrieval*, 3:199, 1999.

X. Su and T. M. Khoshgoftaar. A survey of collaborative filtering techniques. *Advances in artificial intelligence*, 2009, 2009.

T. Tran, K. Lee, Y. Liao, and D. Lee. Regularizing matrix factorization with user and item embeddings for recommendation. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pages 687–696. ACM, 2018.

Z. Wen, W. Yin, and Y. Zhang. Solving a low-rank factorization model for matrix completion by a nonlinear successive over-relaxation algorithm. *Mathematical Programming Computation*, 4(4):333–361, Dec 2012. ISSN 1867-2957. doi: 10.1007/s12532-012-0044-1. URL https://doi.org/10.1007/s12532-012-0044-1.

Z. Wu, H. Tian, X. Zhu, and S. Wang. Optimization matrix factorization recommendation algorithm based on rating centrality. *CoRR*, abs/1806.07678, 2018. URL http://arxiv.org/abs/1806.07678.

X. Yu, W. Bian, and D. Tao. Scalable completion of nonnegative matrices with the separable structure. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, pages 2279–2285. AAAI Press, 2016. URL http://dl.acm.org/citation.cfm?id=3016100.3016217.