Evaluating Narrative-Driven Movie Recommendations on Reddit

Lukas Eberhard
Graz University of Technology
Graz, Austria
lukas.eberhard@tugraz.at

Simon Walk
Detego
Graz, Austria
s.walk@detego.com

Lisa Posch
GESIS & Graz University of Technology
Cologne, Germany
lisa.posch@gesis.org

Denis Helic
Graz University of Technology
Graz, Austria
dhelic@tugraz.at

ABSTRACT

Recommender systems have become omni-present tools that are used by a wide variety of users in everyday life tasks, such as finding products in Web stores or online movie streaming portals. However, in situations where users already have an idea of what they are looking for (e.g., *'The Lord of the Rings', but in space with a dark vibe*), most traditional recommender algorithms struggle to adequately address such a priori defined requirements. Therefore, users have built dedicated discussion boards to ask peers for suggestions, which ideally fulfill the stated requirements. In this paper, we set out to determine the utility of well-established recommender algorithms for calculating recommendations when provided with such a narrative. To that end, we first crowdsource a reference evaluation dataset from human movie suggestions. We use this dataset to evaluate the potential of five recommendation algorithms for incorporating such a narrative into their recommendations. Further, we make the dataset available for other researchers to advance the state of research in the field of narrative-driven recommendations. Finally, we use our evaluation dataset to improve not only our algorithmic recommendations, but also existing empirical recommendations of IMDb. Our findings suggest that the implemented recommender algorithms yield vastly different suggestions than humans when presented with the same a priori requirements. However, with carefully configured post-filtering techniques, we can outperform the baseline by up to 100%. This represents an important first step towards more refined algorithmic narrative-driven recommendations.

ACM Classification Keywords

H.5.m. Information Interfaces and Presentation (e.g. HCI): Miscellaneous; See http://acm.org/about/class/1998/ for the full list of ACM classifiers. This section is required.

CHI'16, May 07–12, 2016, San Jose, CA, USA

© 2016 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ISBN 123-4567-24-567/08/06...$15.00

DOI: http://dx.doi.org/10.475/123_4

Author Keywords

Narrative-driven recommender, Collaborative filtering, Evaluation

INTRODUCTION

The practical applications of recommender systems are manifold. In general, they are tools that help users to find and discover items of interest in large collections, such as books, movies, or people. In a common collaborative filtering scenario, a recommender system makes use of a user's history and predicts new items that user is likely to read, watch, or connect to.

Problem. Often, users already have vague to specific ideas about the desired entities they want to be recommended. More precisely, users often seek recommendations that fit arbitrary criteria, such as movies that evoke certain emotions or have a surprising ending, instead of obtaining suggestions purely based on their (and other users') histories of interactions within a given system. These criteria represent the **narrative** of a recommendation request. Recommendations generated by incorporating such a narrative are referred to as narrative-driven recommendations [8] and also build the foundation for conversation-based recommendation approaches used in chat- and voice-bots. Due to the lack of automated recommender systems that can accurately calculate such recommendations, users have built various discussion boards on the Web to ask peers for suggestions. For example, as of March 2017, there were $190,000$ discussion threads with nearly $25,000$ threads containing requests with a narrative for interesting books on the social cataloging website LibraryThing[1] [8]. Also, there are several subreddits on reddit.com, where users can ask for, for example, video game, movie, or board game suggestions. Requests for movie recommendations can look as follows: *"[...] Movies with the genre 'Crime' [...] like 'Nightcrawler' and 'Prisoners' [...] And it is great if there is any form of plot twists"*[2]. The (free-form) narrative of such requests defines several different elements, such as positively or negatively associated movies (i.e., *Nightcrawler*, *Prisoners*), preferred as well as unwanted genres (i.e., *Crime*), and specific keywords

[1] https://www.librarything.com

[2] https://www.reddit.com/r/MovieSuggestions/comments/3fvycr

that define desired or undesired attributes/keywords of the movie (i.e., *plot twists*) [8].

Approach. In this paper, we systematically analyze the suitability of five standard recommender algorithms for supporting such a narrative in recommender systems. For our evaluation, we compare human suggestions for requests that provide a narrative with purely algorithmic recommendations.

To that end, we first compile an evaluation dataset by collecting and parsing narrative requirements from users of the subreddit r/MovieSuggestions[3]. We extract requirements from the unstructured text of submissions and comments with the help of crowdworkers and make our dataset available online[4] for future research. Next, we implement a recommender framework based on ratings, reviews and textual information of movies available on the Internet Movie Database[5] (IMDb). We calculate recommendations using the following five algorithms for our analysis: item-based collaborative filtering (CF), matrix factorization (MF), a content-based filtering approach based on TF-IDF similarities (TF-IDF), document-level embeddings (Doc2Vec), and a network-based approach (NW). In addition, we extract movie suggestions generated by IMDb, which we use as an empirical baseline (IMDb baseline). We apply post-filtering and re-ranking strategies using metadata from IMDb to refine the computed recommendations. Finally, we evaluate the five recommender approaches by measuring the overlap between their recommendations and the suggestions from users in our evaluation dataset from reddit. Our initial results suggest that traditional recommender algorithms exhibit great potential for improvement when presented with a narrative, as they lack the proper means to include a priori specified requirements in the recommendation process. Further, we demonstrate that we can improve all recommendation approaches (including existing empirical IMDb recommendations) by applying post-filtering and re-ranking strategies using metadata available in the narrative of the initial requests on reddit.

Contributions. With our analyses, we make the following contributions. First, we publish a reference dataset, which enables researchers to conduct independent analyses, advancing the state of research in the context of narrative-driven recommendations. Second, we evaluate the performance of five well-studied recommender approaches on our reddit evaluation dataset, containing a total of 1,480 recommendation requests that provide a narrative. Third, we demonstrate how to improve narrative-driven recommendations by introducing post-filtering and re-ranking techniques and analyze their importance for each of our five implemented recommendation approaches.

RELATED WORK

Traditional Recommender Systems. There exists a vast variety of studies about recommender systems and algorithms (e.g., [3, 4, 5, 8, 10, 11, 14, 15, 18, 20, 21, 22, 27, 34]). However, we still only have limited insights into the quality and suitability of traditional recommender algorithms for calculating narrative-driven recommendations. Typically, traditional research in recommender systems focuses on algorithmic advantages in common scenarios, such as applying users' histories and profiles to compute recommendations [11, 14, 21, 27].

Context-Aware Recommender Systems. To compute recommendations that are well suited to the current needs of a user, context-aware recommender systems use contextual information, such as the time of the day or the current location or interests of the user, besides user profiles and histories [15]. In a context-driven environment, Adomavicius et al. [4] introduced REQUEST, which is a query language for customizing recommendations based on users' personalized recommendation needs. Hariri et al. [15] proposed a query-driven context-aware recommender system that considers user profiles, item representations, and contextual information, such as interests or needs of a user in a specific situation.

A context-aware support vector machine for application in a context-dependent recommender system was proposed by Oku et al. [24]. The authors found that for information recommendation it is important to consider the situations or conditions which influence the users' decisions (e.g., time of day, weather, physical condition).

In the study of Adomavicius et al. [1], the authors presented a multidimensional recommendation model that is based on additional contextual information, such as profiles and aggregation hierarchies. They evaluated their approach on a movie recommender by exploiting contextual information, such as when a movie was seen, where, and with whom. They empirically demonstrated that this contextual information can improve the recommendations.

Basu et al. [5] conducted a study on IMDb data, in which they proposed a recommender approach that exploits both user ratings and content information using collaborative, content, and hybrid features. Lamprecht et al. [18] analyzed how IMDb recommendation networks support alternative information retrieval strategies, such as browsing. The authors showed that current recommendation networks are poorly navigable and require further improvements. This shows potential for providing context-aware recommender systems that involve the current needs of a user without the need of clicking through poorly navigable recommendation networks until finding a more or less fitting movie.

Adomavicius and Tuzhilin [3] argued that relevant contextual information is important when providing recommendations. Such contextual information can be obtained explicitly (i.e., users provide additional information) or implicitly (i.e., system implies the context automatically from the given requirements). To that end, the authors introduced pre- and post-filtering techniques for capturing relevant context during the recommendation process. They used these methods for selecting a relevant set of data and for filtering out irrelevant recommendations or adjusting the ranking of the obtained recommendation list based on a given context. They discussed the notion of context and how it can be modeled, and con-

[3]https://www.reddit.com/r/MovieSuggestions
[4]AnonymizedURL
[5]https://www.imdb.com

ducted an empirical analysis using movie data regarding only the combination of several pre-filters. In this paper, we follow up on their ideas.

In contrast to the study of Panniello et al. [25] that constitutes a first step towards the comparison of pre- and post-filtering using just one contextual variable for each applied dataset, we introduce and combine several post-filters and evaluate their utility in the context of narrative-driven movie recommendations.

Narrative-Driven Recommender Systems. Bogers and Koolen [8] presented a specific context-aware recommendation scenario called narrative-driven recommendation. In such a scenario recommendations are computed based on past transactions of users, and a narrative description of the current needs and interests of users. Narrative-driven recommendations are related to conversational-based recommender systems, where users ask for suggestions in a community and other users then come up with suggestions and possible explanations for their choices [10, 20, 22].

Bogers [7] analyzed the movie discussion threads from the IMDb message boards that contain requests for movies to watch. The author found that content (e.g., movie description), different types of metadata (e.g., genre, language, release year), and searching for a movie by describing its content (e.g., in cases where users forgot the movie title) are important for movie selection practices.

In contrast to previous work, we present the first in-depth analysis and evaluation of recommender algorithms to support narratives for the computation of recommendations.

REDDIT NARRATIVES EVALUATION DATASET

On r/MovieSuggestions, users ask other users for movie suggestions by describing, in natural language, what they are looking for. For example, typical posts include questions such as *“[...] Really dark, slow paced movies with minimal story, but incredible atmosphere, kinda like ‘Drive’ (2011), ‘The Rover’ (2014), or ‘No Country for Old Men’ (2007)? [...]”*[6]. The narrative of this example includes references to three “positively associated” movies (i.e., *Drive*, *The Rover*, *No Country for Old Men*) and several keywords that define the gist of the plot (i.e., *incredible atmosphere, dark, slow paced*, *minimal story*). As these requests are written in free-form text, the amount of information that can be leveraged for calculating recommendations varies. For example, users sometimes include detailed lists and descriptions of movies that they previously did (or did not) enjoy in their requests. Other times, only a single movie is referenced. Further, users frequently provide keywords in the narrative, which should apply to the suggestions (e.g., *“[...] Movies that will make me want to cry [...] like ‘Extremely Loud and Incredibly Close’ ”*[7] with the keyword *cry* and one desired movie, or *“[...] Movies that take place primarily in one room or building. [...] Examples: Exam, Circle, Hateful Eight, Die Hard [...]”*[8] including the keywords *one room or building* and some desired movies). Other users then suggest appropriate movies by writing comments to the original post. Note that recommendations on r/MovieSuggestions are usually generated only considering the information provided in each submission, ignoring previous interactions or requests of users, limiting the amount of available information (see Table 1 for a more detailed characterization of our dataset).

Requests with a Narrative. To compile a dataset suitable for the evaluation of narrative-driven recommendations, we extracted all submissions from r/MovieSuggestions that (i) were posted between August 14, 2011 and August 1, 2017[9], (ii) had received at least ten comments, and (iii) had a score (i.e., the sum of up- and down-votes) greater than zero (3,640 of 23,484 submissions after filtering). Additionally, we extracted all comments to these submissions that had a score greater than zero, which we used as indicator for good recommendations (24,851 of 201,298 comments after filtering). For the compilation of the dataset, we asked crowdworkers to match the movies, genres, actors and other keywords mentioned in the reddit narratives to their corresponding entries on IMDb. The IMDb website provides a wide variety of information about movies and TV shows, such as genres, descriptions, trailers, plot summaries, as well as details about the cast, producers,

[6]https://www.reddit.com/r/MovieSuggestions/comments/3kjrus
[7]https://www.reddit.com/r/MovieSuggestions/comments/11ycep
[8]https://www.reddit.com/r/MovieSuggestions/comments/4va9p8

[9]The dump is available at https://files.pushshift.io/reddit [6]

Table 1: *Reddit Evaluation Dataset Characteristics.* This table lists the statistics of our reference dataset, which we compiled using data from r/MovieSuggestions and crowdworkers on Crowdflower to extract structured data from the unstructured text of the submissions and comments.

#Submissions	1,480
Average Submission Score	11.78
# Movies in Submissions	5,521
# Unique Movies in Submissions	1,908
# Submissions with Desired Movies	1,480
# Submissions with Undesired Movies	75
# Keywords in Submissions	4,492
# Unique Keywords in Submissions	1,878
# Submissions with Desired Keywords	1,198
# Submissions with Undesired Keywords	153
# Genres in Submissions	762
# Unique Genres in Submissions	26
# Submissions with Desired Genres	491
# Submissions with Undesired Genres	61
# Actors in Submissions	100
# Unique Actors in Submissions	79
# Submissions with Desired Actors	75
# Submissions with Undesired Actors	6
#Comments	21,032
Average Comment Score	2.88
# Movie Suggestions in Comments	43,402
# Unique Movie Suggestions in Comments	6,071
Average # Movie Suggestions per Submission	29.33
Average # Movie Suggestions per Comment	2.48

and writers. In February 2017 the publicly available dataset[10] included information about 4.1 million titles and 7.7 million people.

Crowdsourcing Requests and Suggestions. To obtain a structured set of user requests and suggestions, we asked crowdworkers to annotate the unstructured text of the previously extracted submissions and comments from r/MovieSuggestions after filtering (see Table 1 for more details). To that end, we designed four micro tasks on the crowdsourcing platform CrowdFlower (now Figure Eight).[11] First, in the SUBMISSIONS task, we asked crowdworkers to identify all movie titles in each submission. Second, in the SENTIMENT task we asked crowdworkers to specify the sentiment of the user with respect to a movie mentioned in a submission (i.e., positive or negative association to the requested suggestions). We defined *positively associated* movies as movies that users liked or where they stated that they were looking for movies similar to these. Analogously, we defined *negatively associated* movies as movies that users disliked or where they stated that they were not looking for similar movies. Third, in the KEYWORDS task we asked crowdworkers to identify additional information about the user's preferences in each submission's text (i.e., keywords). To extract these keywords, we provided the crowdworkers with a list of keyword types containing, for example, genres, movie settings, and events.[12] We asked the crowdworkers to identify *positively associated keywords* (i.e., keywords which should apply to the recommendations) and *negatively associated keywords* (i.e., keywords which should not apply to the recommendations). Finally, in the COMMENTS task, crowdworkers identified all movie titles in the comments to each submission.

A minimum of three separate crowdworkers worked on each submission in the SUBMISSIONS task. Where there was high disagreement among the workers, we requested judgements from two additional workers. Three workers worked on each movie in the SENTIMENT task and each comment in the COMMENTS task. In the KEYWORDS task, five distinct workers extracted keywords from each submission. We ensured the quality of the crowdworkers' output by requiring an entry-quiz for each task. Additionally, we continuously assessed workers via test questions.

Post-Processing. To obtain a well-curated dataset for the training and evaluation of narrative-driven recommendations, we carried out several manual and semi-automatic post-processing steps.

First, we manually reviewed all submissions from the SUBMISSIONS task and all comments from the COMMENTS task that did not have the crowdworkers' full agreement on movie titles. The crowdworkers fully agreed on the movie titles in $1,205$ submissions and $16,893$ comments, and they disagreed on titles in 457 submissions and $7,958$ comments, which we then manually reviewed. During this step, we also removed submissions and comments without movie titles.

Second, we aggregated the answers from the SENTIMENT and KEYWORDS tasks. In the SENTIMENT task we applied a majority vote whereas in the KEYWORDS task we first split the keyword strings provided by the workers into single keywords. Then, we retained all keywords identified by at least two out of the five workers.

Third, we automatically and unambiguously matched $1,298$ movie titles from the SUBMISSIONS and $5,695$ movie titles from the COMMENTS task to movie titles from IMDb. We then manually reviewed all movie titles that could not be automatically mapped to IMDb. In cases where more than one (or no) movie existed with the exact same movie title, we matched the movie using contextual information of the submission and the comments. In cases where we did not have sufficient information to unambiguously map movies, we removed them from our reference dataset.

Fourth, we automatically identified all common movie genres and actors in the keywords by matching them to the 25 genres and $294,533$ actors available in our IMDb data.

Finally, we removed all movies from the submissions and comments that are not present in our IMDb data. Further, we removed submissions that did not contain any positively associated movie and that did not receive at least ten unique movie suggestions in the comments. After the last preprocessing step, our reference dataset[13] consists of $1,480$ movie-recommendation requests and $43,402$ corresponding suggestions, as noted in Table 1.

EXPERIMENTAL SETUP

Our recommendation framework (see Figure 1) (i) uses one or more movies as input data, (ii) implements five different recommender algorithms to compute a candidate set of recommendations, and (iii) applies several post-filtering and re-ranking strategies, based on metadata from IMDb to calculate a final list of (top ten) recommendations.

To assess the importance of narratives for the calculation of recommendations we further calculate an alternative final recommendation list by applying the structured input (in the form of e.g., actors and keywords) from a given reddit narrative in the post-filtering and re-ranking step.

Finally, we evaluate both lists by comparing them to human suggestions from our reddit evaluation dataset (see Section *Reddit Narratives Evaluation Dataset*).

Hyperparameter Optimization. To analyze if and to what extent traditional recommender approaches can support narratives we aim at making as few assumptions as possible and take a data-driven approach. Thus, we conduct an extensive cross-validation over various configurations of the parameters of the algorithms (see framework components highlighted in

[10]https://www.imdb.com/interfaces

[11]https://www.figure-eight.com

[12]The full list of keyword types included *genres, actors, movie directors, movie characters, movie producers, movie production companies, events or special occasions, movie settings,* and *other movie characteristics*.

[13]Note that on our website AnonymizedURL, we also provide necessary information about the mapping of genres and actors, and an extended version of our dataset without thresholds for the number of suggestions or the number of positively mentioned movies.

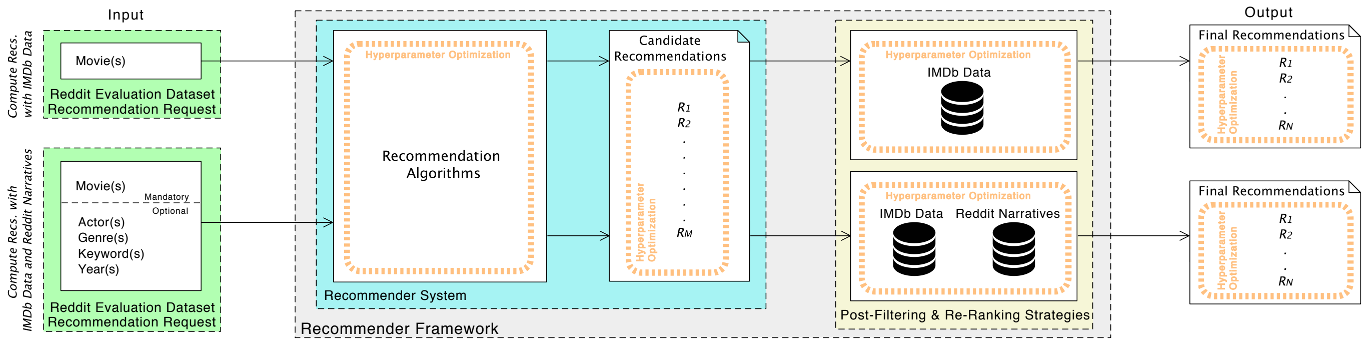

Input
Compute Recs. with IMDb Data
Movie(s)
Reddit Evaluation Dataset Recommendation Request
Compute Recs. with IMDb Data and Reddit Narratives
Movie(s)
Mandatory
Optional
Actor(s)
Genre(s)
Keyword(s)
Year(s)
Reddit Evaluation Dataset Recommendation Request
Hyperparameter Optimization
Recommendation Algorithms
Candidate Recommendations
R_1
R_2
R_M
Hyperparameter Optimization
Recommender System
Hyperparameter Optimization
IMDb Data
Hyperparameter Optimization
IMDb Data
Reddit Narratives
Post-Filtering & Re-Ranking Strategies
Recommender Framework
Output
Final Recommendations
R_1
R_2
R_N
Hyperparameter Optimization
Final Recommendations
R_1
R_2
R_N
Hyperparameter Optimization

Figure 1: *Experimental Setup.* The recommender framework accepts several input parameters (see *Input*), extracted from the narrative of a recommendation request (e.g., reddit submissions). We distinguish between requests that only provide information about desired movies (see *Compute Recs. with IMDb Data*) and requests that include more detailed information from their narratives (see *Compute Recs. with IMDb Data and reddit Narratives*). The input parameters are then fed into the implemented recommender algorithms (see *Recommender System*), which calculate a first list of candidate recommendations. We then apply post-filters (see *Post-Filtering & Re-Ranking Strategies*) based on IMDb Metadata, or IMDb Metadata and reddit narratives, to provide a re-ranked list of recommendations (see *Output*), which better reflects the requirements defined in the narrative of the recommendation request. For all parts that are highlighted in orange (see *Hyperparameter Optimization*), we conduct an extensive grid search over relevant parameter configurations to find the optimal parameter settings.

orange in Figure 1). Specifically, we optimize (i) hyperparameters for the algorithms, such as similarity measures or regularization parameters, (ii) the lengths of the initial and the final recommendation lists, and (iii) hyperparameters of the post-filtering and re-ranking mechanisms, such as overlap measures or functional forms for various scores. We discuss the optimal parameter configurations that we obtain along with introducing a given framework component.

IMDb Movies & Ratings. To implement the recommender algorithms we use data from IMDb. Note that training of recommender algorithms directly on our reddit evaluation dataset is not viable due to the sparsity of data. We leave this option open for future work when more data is available.

In addition to the publicly available IMDb dataset, we collect user reviews and individual ratings for all movies on IMDb. For our experiments, we only consider movies and discard all other types available on IMDb, such as TV series or single TV episodes. To minimize noise and to allow for fair comparisons between the different approaches, we only *keep movies* that have (i) more than 1,000 user ratings, (ii) at least one user review, (iii) a movie description, and (iv) at least one person in the cast. In contrast, we do not *remove users* with small numbers of ratings, as this preprocessing step does not improve our results. We obtain the rating thresholds for movies (1,000) and users (no limit) via grid search.[14] For more details see Table 2. Further, we compute centered ratings [12, 29] by removing user and item bias which improves the overall performance of all implemented recommender approaches.

Recommender Strategies

We generate recommendations by computing similarities between an input movie and all other movies available in our IMDb dataset. Each recommender algorithm determines how and with which data we calculate similarity. As similarity measures we use cosine similarity and an inverse of Euclidean distance and select the best performing measure via cross-validation. In cases where we have more than one input movie we aggregate similarity values. Hence, for each movie in our IMDb data, we add all similarities for all positively associated input movies. Our cross-validation yields better results when

Table 2: *IMDb Dataset Characteristics.* This table describes the features of the dataset that we used for computing the recommendations of our implemented recommender algorithms.

#Movies	11,578
#Ratings	144,021,151
Average #Ratings per Movie	≈ 12,439.21
#Users with Ratings	1,144,136
Average #Ratings per User	≈ 125.88
#Reviews	1,880,837
Average #Reviews per Movie	≈ 162.45
#Users with Reviews	598,247
Average #Reviews per User	≈ 3.14
#Credits	667,279
#People in Cast & Crew	322,881
#Actors	294,533
Average #Actors per Movie	≈ 25.44
Average #Movies per Actor	≈ 2.27
#Genres	32,767
#Unique Genres	25
Average #Genres per Movie	≈ 2.83
#Plot Keywords	1,124,510
#Unique Plot Keywords	89,003
Average #Plot Keywords per Movie	≈ 97.12

[14]We perform the grid search over 0 to 10,000 movie ratings in increments of 500, as well as 0 to 500 user ratings with increments of 10.

we do not subtract negative input movies for the aggregation of similarity values. We call the aggregated similarities *algorithmic score*. Thus, the output of each approach is a ranked list of candidate movies with their corresponding algorithmic scores. We conduct experiments with the following five approaches:

Item-Based Collaborative Filtering. This approach finds similar movies to the movies that a user liked [33]. Thus, we use the IMDb user-ratings vectors of two movies to compute their similarity [33]. The best performing similarity measure for this approach is cosine similarity.

Matrix Factorization. This approach is a well-established method that approximates a ratings matrix with the product of two matrices, one connecting users to factors representing their preferences, and another connecting movies to factors representing their properties [17, 26, 30, 31]. In this paper, we factorize the IMDb user-ratings matrix in a standard manner by minimizing a regularized squared error with a stochastic gradient descent [13]. We then use cosine similarity (determined via hyperparameter optimization) to compute similarity between the obtained movie factors.[15]

Content-Based Filtering with TF-IDF. We use this approach to find similar movies by calculating similarity between movies using their descriptions and user reviews [2]. Hence, we compute the term frequency–inverse document frequency score [32] of terms in the description and user reviews for each movie. To compute the similarity between movies we use normalized TF-IDF vectors and the reciprocal of Euclidean distance (determined via hyperparameter optimization). We receive the best results with unigrams and bigrams, no cut-off threshold for less frequent terms, and with a maximum of 500 features for the TF-IDF vectors.[16]

Document-Level Embeddings with doc2vec. Similar to the TF-IDF approach, we use movie descriptions and reviews as basis for this approach. doc2vec was first proposed by Le and Mikolov [19] and is an enhancement of word2vec [23], extending the learning of embeddings from words to documents. We use doc2vec to generate a document vector for each movie and use these vectors to compute similarities between movies. We obtain the best results with a feature vector dimensionality of 500 and cosine similarity.[17]

Network-Based Recommendations. We use this approach to find movies with similar casts and crews by creating a bipartite graph between movies and people involved in those movies. Specifically, we connect each movie to all cast and crew members including actors, cinematographers, composers, costume designers, directors, editors, producers, production designers, special effect companies, and writers. We calculate similarity between movies by counting common neighbors in the bipartite graph [16].

IMDb Baseline. We collect all movie suggestions on IMDb[18] for each movie in our dataset to determine if and to what extent existing (empirical) recommender systems are suitable to address a narrative. IMDb provides a maximum of twelve recommendations per movie. We use these recommendations for all (desired) input movies in the narrative of each submission. Note that IMDb does not provide any ranks or numerical values quantifying the quality of each recommendation.

Post-Filtering & Re-Ranking

We further refine the algorithmic recommendations by defining a number of post-filtering approaches. This refinement allows us to (i) include additional metadata from IMDb (see Section *Post-Filtering & Re-Ranking with IMDb Data*), and (ii) optionally include reddit narratives (see Section *Post-Filtering & Re-Ranking with Reddit Narratives*). Again, for evaluation of various post-filters we take a data-driven approach and make an extensive cross-validation over various configuration. This allows us to evaluate both importance of the individual post-filters as well as the interactions between different post-filters.

With post-filtering techniques we modify the calculated recommendation list by (i) removing irrelevant recommendations for a given movie, and (ii) re-ranking the obtained list. In general, the more properties (e.g., genres, keywords, actors) the candidate movies have in common with a given input movie, the higher they get ranked. For example, we compute the overlap of genres of all input movies and a candidate movie. With all scores calculated we re-rank the candidate lists by combining algorithmic scores of each candidate recommendation with the corresponding post-filtering scores to compile a final recommendation list. We evaluate the resulting (final) list by comparing it to human suggestions from our reddit evaluation dataset. When limiting our final recommendation list to a total of ten movies to be displayed, we achieve the best results with 500 candidate recommendations.[19]

Post-Filtering & Re-Ranking with IMDb Data

With IMDb metadata we re-rank candidate recommendations with the following scores:

IMDb Popularity & Rating Score. Following the intuition that users are generally more interested in higher and more frequently rated movies, we introduce this score which combines the average IMDb rating (*rating score*) of a candidate movie and the number of ratings received on IMDb (*popularity score*). We experiment with various functional forms for the computation of both the average rating and the number of ratings. Specifically, we calculate logarithmic, square root, quadratic, and cubic scaling and achieve the best results

[15]We have tested different numbers of factors ranging from 100 to 1,000 in steps of 100, learning rates between 0 and 0.1 in steps of 0.001, and regularization parameters from 0 to 0.1 in steps of 0.01. We obtain the best results for MF with 500 factors, a learning rate of 0.002, and 0.02 as regularization parameter.

[16]To obtain this configuration we conducted a grid search experiment over different n-grams [9] (i.e., $n = 1, 2, 3$), several cut-off values for terms with a low document frequency from 0 to 0.1 in increments of 0.001, and different numbers of TF-IDF features ranging from 0 to 1,000 in steps of 100.

[17]To obtain this configuration we conducted a grid search experiment with different similarity measures, and different feature vector sizes ranging from 0 to 1,000 in steps of 100.

[18]For an example see “More Like This” on `https://www.imdb.com/title/tt0076759`

[19]We determine the length for the candidate list with a grid search over the range from 100 to 1,000 movies in steps of 100.

with the following functional form: $\log_2(R_i)\overline{r}_i$, where $\overline{r}_i$ is the average rating, and R_i is the number of ratings of movie i.

IMDb Genre Score. Here, we follow the intuition that users prefer movies of similar genres to the specified movies and calculate the IMDb genre score for each candidate movie. As part of our hyperparameter optimization, we compare several overlap measures, including Jaccard's coefficient, cosine similarity, Sørensen-Dice coefficient, and simple matching coefficient. We achieve the best results with similar scaling and normalizing of the overlap between the genres of the candidate movie and individual positively associated movies from the request so that $S_{\text{iGenre}}(i) = \sum_{j \in I_{\text{pMovie}}} (|G_i \cap G_j|^2 / (|G_i||G_j|))$, where I_{pMovie} is the set of positively associated input movies. The inclusion of negatively associated input movies does not improve our results.

IMDb Year Score. We assume that users want to watch movies from similar time periods unless explicitly stated otherwise. Thus, we introduce the IMDb year score, where candidate movies released closer in time to the input movies receive higher scores. We set this score to 1 for a candidate movie with the smallest difference in release year to one of the input movies. We then linearly scale the year score until we reach 0 for a given maximal difference in release years. We obtain the best results with a release year normalization of 50 years.[20]

IMDb Keyword Score. For our recommender framework, keywords are words or phrases that represent a very specific attribute of a movie. For the IMDb keyword score, we use the plot keywords from IMDb and compute the overlap of all plot keywords of a candidate movie and the plot keywords of the input movies. We conduct a grid search experiment to determine the most suitable overlap measures and whether it is better to consider keywords from negative input movies or not. We obtain the best performance when using Jaccard's coefficient as overlap measure while ignoring plot keywords of negative input movies.

IMDb Predecessor & Successor Filters. We assume that users do not want to receive a list of predecessors or successors of the specified input movies as they are likely familiar with the whole series. Hence, we remove predecessor and successor movies from our recommendation lists. For example, if users ask for movies similar to *The Hunger Games: Catching Fire* we remove *The Hunger Games* and *The Hunger Games: Mockingjay - Part 1 & 2* from our recommendation list.

Combining Scores. To compute the final score for each candidate movie we first normalize all computed scores by their highest values, so that (for each score individually) the movie with the highest score receives the value 1. Second, we configure the post-filters for each recommender algorithm, as they are not equally important across our approaches. To that end, we multiply the scores with weights, reflecting their importance for the re-ranking of the recommendation lists and to analyze differences between approaches. We conduct a grid search experiment over all combinations of weights between 0.0 and 1.0 in steps of 0.2, and select the setup that yields the best results in our experiments. Finally, we sum up all weighted scores to obtain the final score for each movie.

[20]Identified via grid search over the year-range from 20 to 100 in steps of 10.

Post-Filtering & Re-Ranking with Reddit Narratives

For the final step of our evaluation we incorporate metadata, available in the narrative of the initial reddit submission, into our recommendations using additional post-filters. Specifically, we use keywords, genres, actors, and years given in the narrative of the movie suggestion requests in our reddit evaluation dataset. Note that we can calculate post-filtering scores from reddit narratives only if users explicitly provided positively/negatively associated attributes or keywords (e.g., actors or genres) in a recommendation request (see Table 1). With all scores calculated we re-rank the candidate lists (500 candidates) again by combining all IMDb post-filtering scores of each candidate recommendation with the corresponding narrative-based post-filtering scores to compile a final recommendation list (ten recommendations). Again, we evaluate the resulting (final) list by comparing it to human suggestions from our reddit evaluation dataset. To that end, we define and compute the following narrative-based post-filtering scores and evaluate their importance by conducting a grid search experiment:

Narrative-Based Genre Score. If genres are stated in the narrative of a request, we use them to calculate the narrative-based genre score for each candidate movie. We ran the same grid search experiment as we did for the *IMDb Genre Score* and determined that the same overlap metric yields the best results. In contrast to the *IMDb Genre Score*, we remove movies with undesired genres from our recommendation list.

Narrative-Based Year Filter. If users explicitly state year thresholds, we re-rank the recommendation list so that movies outside this range are moved to the end of the list.

Narrative-Based Keyword Score. We exploit keywords in a specific request (e.g., "surprising plot twist") to introduce the narrative-based keyword score. With this score we measure how well the description and the user reviews of a candidate movie reflect the keywords stated in a narrative. We conduct a grid search experiment to determine the most suitable overlap measures for the computation of the narrative-based keyword score. We find that counting the incidences of explicitly stated keywords in the description and all user reviews of the respective candidate movies yields the best results. We aggregate the incidences for positive input keywords and subtract them for negative ones. Finally, we compute the narrative-based keyword score by normalizing over the number of words in the used texts.

Narrative-Based Actor Filter. To reflect the requirement of only recommending movies with one or more specific actors, we introduce the actor filter. We re-rank the list of movie recommendations by counting how many of the positively stated actors appear in the respective movies. Further, we remove all movies with actors that users explicitly specified as undesired.

Combining Scores. To combine all narrative-based post-filtering scores we use the same method as for the IMDb post-filtering scores.

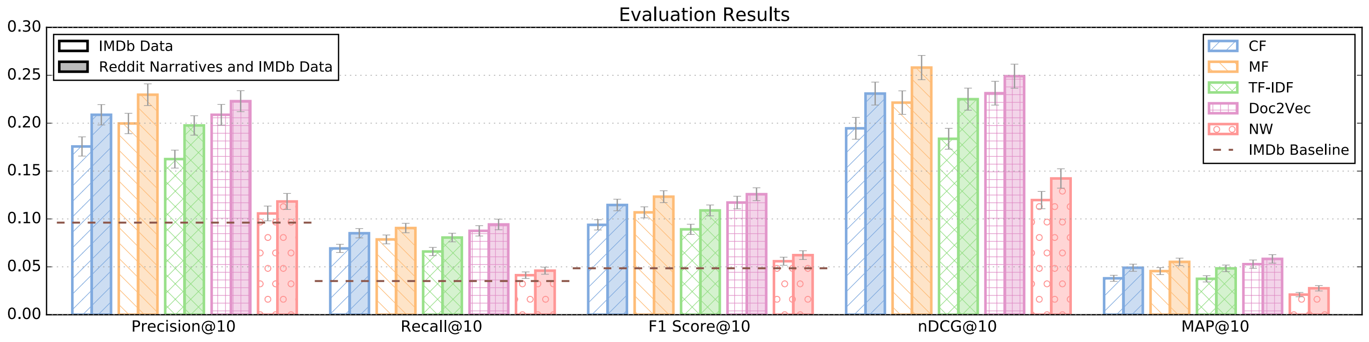


Figure 2: *Results.* This figure depicts the results of our evaluation, comparing our recommendations to the ones of the reddit community in our reddit evaluation dataset. We list the different evaluation metrics on the x-axis, with the corresponding evaluation metric values on the y-axis. The performances of the recommender algorithms with IMDb post-filters are represented by the transparent bars, while the filled bars depict the results for the approaches with additional narrative-based post-filters using reddit data. The grey error bars show the standard deviation of the evaluation metric over all submissions in the test set. All of our approaches outperform the IMDb baseline (dashed horizontal line). We can further improve the results by adding narrative-based post-filters, where Doc2Vec outperforms all other approaches with F1 scores more than twice as good as the IMDb baseline, followed by MF, CF, and TF-IDF, and least improvements for NW.

Evaluation

We evaluate the implemented approaches on our reddit evaluation dataset. Specifically, we use the narrative from each submission to calculate movie recommendations and count the overlap between the movie suggestions of the reddit community, extracted from the replies to corresponding submission (see Section *Reddit Narratives Evaluation Dataset*) and our algorithmic movie recommendations. We calculate precision, recall, F1 score, normalized discounted cumulative gain (nDCG), and mean average precision (MAP) [28, 35]. After training our approaches on the IMDb data, we chronologically split our reddit evaluation dataset into a validation (80%) and a test (20%) set (see Table 3). We use (i) the validation set to conduct all grid search experiments for optimizing hyperparameters for the recommender framework, and (ii) the test set to evaluate the performance of the implemented approaches. We limit our final recommendation lists to ten movies.[21] To allow for a fair comparison we also limit the number of recommendations for our IMDb baseline to ten movies (picked at random, as recommendations are not ranked). First, we evaluate the standard algorithms with post-filters and scores calculated by using IMDb metadata. Second, we measure the performance improvements with the narrative-based post-filters and scores.

Table 3: *Evaluation Protocol.* Basic statistics of the validation set and the test set.

	#Submissions	Timeframe
Validation Set	1,184	08-2011 – 11-2016
Test Set	296	11-2016 – 07-2017
Overall	1,480	08-2011 – 07-2017

RESULTS & DISCUSSION

Post-Filtering & Re-Ranking with IMDb Data

Figure 2 depicts the results of the evaluation of our implemented algorithms for calculating recommendations for a given narrative using our reddit evaluation dataset. The transparent bars represent the means of the evaluation metrics over all submissions in the test set for a given approach using only IMDb-based post-filters, with the error bars showing the standard error. All of our analyzed approaches, while only relying on IMDb-based post-filters, manage to outperform the IMDb baseline (cf. horizontal dashed line in Figure 2). Doc2Vec performs best in all evaluation metrics with an F1 score of 0.117, which is more than twice as good as the IMDb baseline, followed by MF with 0.107, CF with 0.094 and TF-IDF with 0.089, and NW, which performs consistently worst with an F1 score of 0.056, while still outperforming the IMDb baseline.

One possible reason for the moderate performance of NW might be that this approach is fundamentally based on the assumption that users want to see other movies with a similar cast. This inherent restriction appears to impair our results when incorporating the narratives provided by users. However, more research is warranted to further investigate this hypothesis, which we leave open for addressing in future work. MF and CF perform roughly twice as good as NW, possibly due to the larger amount of considered data. They are both based on user ratings and follow similar intuitions (i.e., both approaches favor frequently and highly rated movies), which could explain the similarity in the obtained results. TF-IDF, which is based on the text of movie descriptions and user reviews, performs similar to CF. Doc2Vec performs best of all approaches using the same data, which we attribute to the underlying mechanisms of the approach. Compared to TF-IDF vectors,

[21]This means that recall@10 and F1 score@10 have a mean upper limit of 0.34 and 0.51 respectively, as the average number of movie suggestions from the community per submission is 29.22 in the test set.

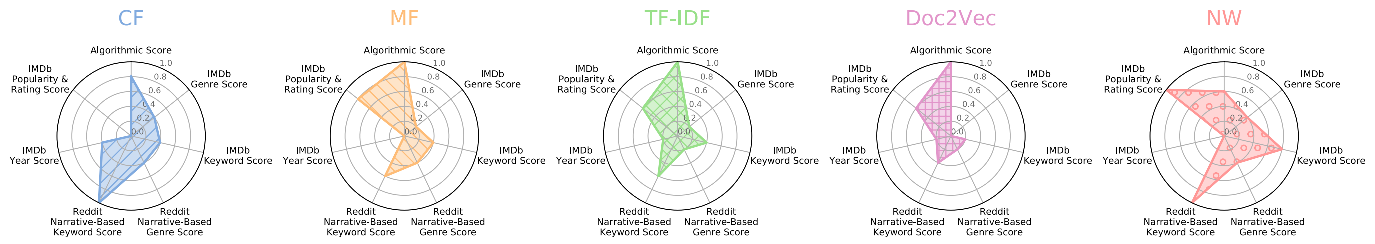


Figure 3: *Score Weights.* Each figure visualizes the score weight configuration of one approach. The algorithmic score and the IMDb popularity and rating scores are important characteristics across most of our approaches. Using narrative-based post-filters, the most important property are the keywords with weights up to 1.0. This also indicates that keywords are important for calculating narrative-driven recommendations.

word embeddings better incorporate latent factors in textual representations, leading to better similarity calculations and, therefore, better recommendations.

Importance of Post-Filters. We present the best-performing IMDb-based post-filter configuration for each approach by depicting the normalized score weight for each post-filter in Figure 3 (obtained by cross-validation), where a higher score signals higher importance of a given post-filter.

In case of CF, we obtain the best-performing configuration with a weight of 0.8 for the algorithmic score, a weight of 0.0 for IMDb popularity and rating influence, and relatively low weights of 0.4 for IMDb genre, keyword and year scores. For MF a higher algorithmic score weight (1.0) and high popularity and rating influence of 0.8 work best, while the year score is completely neglected and the IMDb genre and keyword scores are set to 0.2 and 0.4, respectively. The content-based approaches (TF-IDF and Doc2Vec) exhibit similar best-performing configurations with a 1.0 weight for the algorithmic scores, a 0.6 weight for the IMDb popularity and rating scores and a 0.2 weight for the IMDb year scores. The weights for the IMDb genre and keyword scores range between 0.0 to 0.4. In contrast, NW mainly relies on keywords and popularity and rating influence with weights of 0.6 for the algorithmic score, 1.0 for the IMDb popularity and rating score and 0.8 for the IMDb keyword score. Similar to most other approaches, the influence of IMDb genres and years is quite low.

Findings. Our results reveal that for narrative-driven recommendation scenarios traditional recommender algorithms exhibit only minimal overlaps with human suggestions. Specifically, the algorithmic recommendations using post-filtering with IMDb metadata are computed by calculating similarities between the input movies and the movies from our dataset, while the narrative from reddit is neglected. However, additional information provided by users within their submissions appears to be crucial for the selection of appropriate movie suggestions. Users on reddit parse and consider this information, discerning their recommendations from algorithmic ones.

Post-Filtering & Re-Ranking with Reddit Narratives

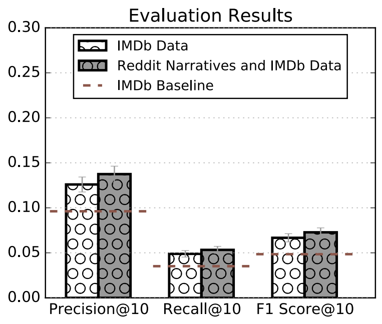
In Figure 2 we also show the results of our experiments with post-filtering and re-ranking of the recommendations using the information from reddit narratives. Due to the fact that we now include narratives we can observe substantial improvements of our results when adding—and carefully configuring—post-filtering techniques (cf. transparent versus color-filled bars in Figure 2). Although not exhausting the potential for improvement, we raise F1 scores of our approaches to be more than twice as high as the IMDb baseline, except for NW. Again, we achieve the best results using Doc2Vec with an F1 score of 0.126, closely followed by MF with 0.123, CF with 0.115 and TF-IDF with 0.109.

Importance of Post-Filters. Although the inclusion of the narrative information improves the recommendations, this additional information needs to be properly configured and strongly depends on the underlying algorithm. For all approaches, the best-performing configuration exhibits higher score weights for keywords extracted from the reddit narratives than for genres. For CF and NW, the narrative-based keyword score is very important, with configuration weights of 1.0, while it is 0.6 for MF and TF-IDF and 0.2 for Doc2Vec. For the narrative-based genre score CF, MF and NW have the same weights of 0.4, while the content-based approaches (TF-IDF and Doc2Vec) exhibit lower score weights of 0.2.

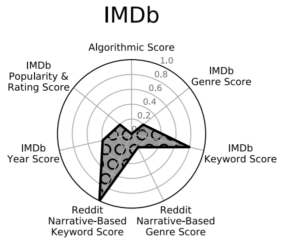
Findings. We find that carefully weighing the different post-filters, particularly in combination with the algorithmic, popularity and rating score, is important to maximize the benefit of the additional information contained in a given narrative.

Further, we find that for all approaches the most important narrative-based post-filter is the keyword score. From this result, we conclude that narrative recommendation requirements, provided in the form of keywords (i.e., the gist of a given text, such as short aspects of the story of a movie), are integral for achieving the best recommendations in our setup. We hypothesize that these keywords provide our post-filters with important information, that specifically helps to filter noise (i.e., unwanted movies) and steer our results towards more fitting movies. However, more research is warranted not only to confirm our hypothesis, but also to determine if additional post-filter or re-ranking strategies exist, for example, based on analyzing characteristics of recommendation requests, which could help to further improve our results.

Besides the narrative-based keyword score, the algorithmic and popularity and rating scores are also important for most of our approaches. This finding also strengthens our intuition that the configuration of algorithmic scores and post-filters is



(a) *Results*



(b) *Score Weights*

Figure 4: *Empirical Recommendations.* This figure depicts the evaluation results and best-performing score weights of the experiment on applying post-filtering techniques on empirical recommendation lists. Subfigure 4a shows the results of our evaluation, comparing the IMDb recommendations to the ones of the reddit community in our reddit evaluation dataset with IMDb post-filters (transparent bars) and with additional narrative-based post-filters using reddit data (colored bars). We list the different evaluation metrics on the *x*-axis, with the corresponding values on the *y*-axis, again. Subfigure 4b visualizes the best-performing score weight configuration. The most important characteristics are keywords from IMDb and from the reddit narratives.

important for the computation of narrative-driven recommendations, and that it is not sufficient to simply apply filters on a given pool of existing recommendations as valuable information is lost and neglected in that process.

Except for NW, the influence of the IMDb genre and keyword scores are similarly low across all approaches. The least important score is the IMDb year score with weights ranging from 0.0 to 0.4. In fact, after manually inspecting our dataset, it appears that movies suggested by humans are more frequently from different years (even decades) than the movies mentioned in the recommendation requests (i.e., reddit submissions).

Applying Post-Filters on Empirical Recommendations

In addition to the datasets presented in this paper, we conduct another experiment to see if our post-filtering strategies can also improve our baseline IMDb recommendations. To that end, we apply all our post-filters on the IMDb baseline. We deploy the same evaluation setup as for our other previous experiments. First, we conduct a grid-search experiment to achieve the best-performing post-filter weights combination. Second, we apply all IMDb post-filters on the IMDb recommendations list and use the top ten recommendations for evaluation. The results, represented by the transparent bars in Figure 4a, reveal that additional IMDb metadata can be used to improve the resulting recommendations. Finally, we add post-filters with metadata from reddit narratives to the IMDb recommendations and further improve our results (see filled bars in Figure 4a), showing that it is possible to refine and improve recommendation algorithms to better support a given narrative using the post-filters presented in this paper.

The most important post-filters for this approach are the keyword scores from the IMDb data as well as from the reddit narratives (see Figure 4b). This further strengthens our finding that keywords provided in narratives are an important factor when re-ranking recommendations. Note that we do not have an algorithmic score for this approach as IMDb does not provide a ranking for their recommendations.

CONCLUSIONS & FUTURE WORK

In this paper, we analyzed and evaluated the potential of a selection of five (MF, CF, TF-IDF, Doc2Vec, NW) recommender algorithms as well as one empirical recommender approach (IMDb) to calculate narrative-driven recommendations. To be able to conduct our analyses, we crowdsourced a dataset from reddit for evaluating narrative-driven recommendations and made this dataset available to other researchers. Moreover, we re-ranked the computed recommendation lists via post-filtering techniques based on specific user requirements from the reference dataset. With our experiments we showed that (i) all implemented recommender approaches struggle to match human-based recommendations and that (ii) the incorporation of the information contained in the narratives (e.g., in the form of post-filters) can substantially improve the performance of recommender algorithms. However, we also showed that our post-filters have to be carefully configured to maximize the benefits of the added information, as the algorithmic score is an important feature across all approaches. Particularly, when applying post-filters on empirical data, we demonstrate that our post-filtering techniques can improve existing approaches, albeit limited due to the lack of an algorithmic score.

For future work, we plan to incorporate the data from reddit narratives in the training phase in the form of, for instance, an additional regularization term. Currently, the recommender algorithms can not be directly trained on the reddit data due to its sparsity but, as our results show, narrative information and the previous human suggestions represent a valuable information that should be leveraged already in the training phase.

Further, we plan on applying our methods to different domains, such as books, board games, or video games, to investigate whether different communities exhibit similar or different recommendation behaviors. Moreover, we will conduct a qualitative evaluation of our recommender framework to study if our suggestions are perceived as useful by the recommendation requesters. We are also dedicated to analyze additional post-filters, informed by characteristics of our reddit evaluation dataset, as well as expanding the arsenal of implemented recommender approaches, such as deep learning and different embedding approaches for the calculation of narrative-driven recommendations. Additionally, we plan on conducting experiments on reddit, by implementing a recommender bot that users can query for recommendations, while providing a narrative. Using this bot, we will be able to evaluate the importance of additional metrics, such as diversity, serendipity or novelty in the context of narrative-driven recommendations.

In this paper we present and publish a reference evaluation dataset, as well as a first analysis of post-filtering and re-ranking strategies for incorporating narratives into recommendations. We strongly believe that our reference evaluation dataset, as well as the presented experiments in this paper will help researchers and practitioners to develop new and improve existing recommendation approaches to better tackle the problem of narrative-driven recommendations, which also represents a fundamental problem in need of novel solutions for the advance of chat and voice bots.

REFERENCES

1. Gediminas Adomavicius, Ramesh Sankaranarayanan, Shahana Sen, and Alexander Tuzhilin. 2005. Incorporating contextual information in recommender systems using a multidimensional approach. *ACM Transactions on Information Systems (TOIS)* 23, 1 (2005), 103–145.
2. Gediminas Adomavicius and Alexander Tuzhilin. 2005. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE transactions on knowledge and data engineering* 17, 6 (2005), 734–749.
3. Gediminas Adomavicius and Alexander Tuzhilin. 2011. *Context-Aware Recommender Systems*. Springer US, Boston, MA, 217–253.
4. Gediminas Adomavicius, Alexander Tuzhilin, and Rong Zheng. 2011. REQUEST: A query language for customizing recommendations. *Information Systems Research* 22, 1 (2011), 99–117.
5. Chumki Basu, Haym Hirsh, William Cohen, and others. 1998. Recommendation as classification: Using social and content-based information in recommendation. In *Aaai/iaai*. 714–720.
6. Jason Michael Baumgartner. 2015. Reddit comment dataset. Website. (July 2015). `https://www.reddit.com/r/datasets/comments/3bxlg7/i_have_every_publicly_available_reddit_comment`.
7. Toine Bogers. 2015. Searching for Movies: An Exploratory Analysis of Movie-related Information Needs. *iConference 2015 Proceedings* (2015).
8. Toine Bogers and Marijn Koolen. 2017. Defining and Supporting Narrative-driven Recommendation. In *Proceedings of the Eleventh ACM Conference on Recommender Systems*. ACM, 238–242.
9. Peter F. Brown, Peter V. deSouza, Robert L. Mercer, Vincent J. Della Pietra, and Jenifer C. Lai. 1992. Class-based N-gram Models of Natural Language. *Comput. Linguist.* 18, 4 (Dec. 1992), 467–479.
10. Konstantina Christakopoulou, Filip Radlinski, and Katja Hofmann. 2016. Towards Conversational Recommender Systems.. In *KDD*. 815–824.
11. Christina Christakou, Spyros Vrettos, and Andreas Stafylopatis. 2007. A hybrid movie recommender system based on neural networks. *International Journal on Artificial Intelligence Tools* 16, 05 (2007), 771–792.
12. Christian Desrosiers and George Karypis. 2011. A comprehensive survey of neighborhood-based recommendation methods. *Recommender systems handbook* (2011), 107–144.
13. Simon Funk. 2006. Netflix update: Try this at home. (2006).
14. Sumit Ghosh, Manisha Mundhe, Karina Hernandez, and Sandip Sen. 1999. Voting for Movies: The Anatomy of a Recommender System. In *Proceedings of the Third Annual Conference on Autonomous Agents (AGENTS '99)*. ACM, New York, NY, USA, 434–435.
15. Negar Hariri, Bamshad Mobasher, and Robin Burke. 2013. Query-driven Context Aware Recommendation. In *Proceedings of the 7th ACM Conference on Recommender Systems (RecSys '13)*. ACM, New York, NY, USA, 9–16.
16. Zan Huang, Xin Li, and Hsinchun Chen. 2005. Link Prediction Approach to Collaborative Filtering. In *Proceedings of the 5th ACM/IEEE-CS Joint Conference on Digital Libraries (JCDL '05)*. ACM, New York, NY, USA, 141–142.
17. Yehuda Koren. 2008. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 426–434.
18. Daniel Lamprecht, Florian Geigl, Tomas Karas, Simon Walk, Denis Helic, and Markus Strohmaier. 2015. Improving Recommender System Navigability Through Diversification: A Case Study of IMDb. In *Proceedings of the 15th International Conference on Knowledge Technologies and Data-driven Business (i-KNOW '15)*. ACM, New York, NY, USA, Article 21, 8 pages.
19. Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *International Conference on Machine Learning*. 1188–1196.
20. Tariq Mahmood and Francesco Ricci. 2009. Improving Recommender Systems with Adaptive Conversational Strategies. In *Proceedings of the 20th ACM Conference on Hypertext and Hypermedia (HT '09)*. ACM, New York, NY, USA, 73–82.
21. Harry Mak, Irena Koprinska, and Josiah Poon. 2003. Intimate: A web-based movie recommender using text categorization. In *Web Intelligence, 2003. WI 2003. Proceedings. IEEE/WIC International Conference on*. IEEE, 602–605.
22. Lorraine McGinty and James Reilly. 2011. On the evolution of critiquing recommenders. In *Recommender Systems Handbook*. Springer, 419–453.
23. Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* (2013).

24. Kenta Oku, Shinsuke Nakajima, Jun Miyazaki, and Shunsuke Uemura. 2006. Context-aware SVM for context-dependent information recommendation. In *Proceedings of the 7th international Conference on Mobile Data Management*. IEEE Computer Society, 109.

25. Umberto Panniello, Alexander Tuzhilin, Michele Gorgoglione, Cosimo Palmisano, and Anto Pedone. 2009. Experimental comparison of pre-vs. post-filtering approaches in context-aware recommender systems. In *Proceedings of the third ACM conference on Recommender systems*. ACM, 265–268.

26. Arkadiusz Paterek. 2007. Improving regularized singular value decomposition for collaborative filtering. In *Proceedings of KDD cup and workshop*, Vol. 2007. 5–8.

27. Patrice Perny and Jean-Daniel Zucker. 2001. Preference-based search and machine learning for collaborative filtering: the "film-conseil" movie recommender system. *Information, Interaction, Intelligence* 1, 1 (2001), 9–48.

28. David Martin Powers. 2011. Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation. (2011).

29. Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. 1994. GroupLens: An Open Architecture for Collaborative Filtering of Netnews. In *Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work (CSCW '94)*. ACM, New York, NY, USA, 175–186.

30. Ruslan Salakhutdinov and Andriy Mnih. 2007. Probabilistic Matrix Factorization.. In *Nips*, Vol. 1. 2–1.

31. Ruslan Salakhutdinov and Andriy Mnih. 2008. Bayesian probabilistic matrix factorization using Markov chain Monte Carlo. In *Proceedings of the 25th international conference on Machine learning*. ACM, 880–887.

32. Gerard Salton and Michael J McGill. 1986. Introduction to modern information retrieval. (1986).

33. Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-based Collaborative Filtering Recommendation Algorithms. In *Proceedings of the 10th International Conference on World Wide Web (WWW '01)*. ACM, New York, NY, USA, 285–295.

34. Paul Seitlinger, Dominik Kowald, Simone Kopeinik, Ilire Hasani-Mavriqi, Tobias Ley, and Elisabeth Lex. 2015. Attention Please! A Hybrid Resource Recommender Mimicking Attention-Interpretation Dynamics. In *Proceedings of the 24th International Conference on World Wide Web (WWW '15 Companion)*. 339–345.

35. Emine Yilmaz, Evangelos Kanoulas, and Javed A. Aslam. 2008. A Simple and Efficient Sampling Method for Estimating AP and NDCG. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '08)*. ACM, New York, NY, USA, 603–610.