Journal of Data Intelligence, Vol. 1, No. 4 (2020) 442–467


# A COLLABORATIVE FILTERING BASED APPROACH TO CLASSIFY MOVIE GENRES USING USER RATINGS [a]

RAJI GHAWI[b]
*Bavarian School of Public Policy*
*Technical University of Munich*
*Munich, Germany*
*raji.ghawi@tum.de*

JÜRGEN PFEFFER
*Bavarian School of Public Policy*
*Technical University of Munich*
*Munich, Germany*
*juergen.pfeffer@tum.de*

In this paper, we present an approach for classifying movie genres based on user-ratings. Our approach is based on collaborative filtering (CF), a common technique used in recommendation systems, where the similarity between movies based on user-ratings, is used to predict the genres of movies. The results of conducted experiments show that our genres classification approach outperforms many existing approaches, by achieving an F1-score of 0.70, and a hit-rate of 94%. We also construct a multilayer network of movies, with genres as layers. We apply agglomerative clustering on the layers of this network to obtain a comprehensible taxonomy of genres which groups together similar genres using the similarity of their movies in terms of user preferences.

*Keywords*: Movie Genre, Classification, Collaborative Filtering, Multi-label Classification, KNN, Multilayer Networks, Taxonomy.

## 1. Introduction

The task of labeling movies according to their corresponding genre is a challenging classification problem. Movies may belong to multiple genres at the same time, making movie genre assignment a typical multi-label classification problem, which is per se much more challenging than standard single-label classification.

The ability to automatically classify movies by genre has many benefits [35], as it would enable 1) indexing multimedia databases to help search for particular types of film, 2) automatically identifying movies for consumers through user preference modeling, 3) facilitating automatic movie content filtering and summarization.

In literature, various automatic genre classification methods have been proposed for both movies [17, 2, 10, 9] and general video data [5, 21, 16, 29]. Some of the proposed methods rely on textual features of movies such as synopsis, or plot summaries. Other methods rely on audio features ([5, 21, 18, 23]), video features ([29, 2, 30, 35]), and low-level visual features ([5, 2, 17, 9]).

[a] This paper is an extension of an already published conference paper [6].
[b] Hochschule fr Politik, Richard-Wagner-Strae 1, 80333 Munich, Germany

In this paper, instead of using textual, or audio-visual information about movies, we use a completely different type of information, namely, the user ratings of the movies. Our classification approach is based on Collaborative Filtering (CF), which is a technique used by recommendation systems. The CF technique predicts and recommends items (information, products or services) that the user might like. Recommender systems are tools which should help users to overcome information overload by selecting the most interesting information based on their preferences [14]. These systems attempt to predict a user's interest towards an item, such as movies to watch, books to read, or products to buy. Recommender systems are popular in both commercial and research settings, and they are applied in a variety of applications such as movies, music, books, social connections and venues. In particular, movie recommender systems (such as Netflix) provide the customers with personal recommendations of movies they might like. Existing CF algorithms employed in movie recommendation systems predict the unknown rating of a given user for a movie using only the ratings (i.e., preferences) of other like-minded users who have seen the movie. [12]

In this paper, we use the principle of Collaborative Filtering in order to classify the genres of movies. The idea behind classification of movie genres is to use the similarity between movies based on user ratings, to predict the genres of movies. The assumption is the following: a user may like/prefer movies of certain genres, and hence give them high ratings. Thus, movies that belong to the same genres will be rated by the set of users who like/prefer this genres. In other words, movies of the same genres would be similar to each other in terms of common users, and (correlation of) ratings. Therefore, we will classify movie genres based on the genres of their similar movies.

In order to efficiently find similar movies of a target movie, we construct, as a preprocessing step, a network whose nodes are movies, and whose edges connect similar movies (i.e., movies that have a similarity score above certain threshold). Thus, during genres classification, the problem of finding similar movies of a target movie is reduced to a straightforward retrieval of the neighbors of that target movie within the network. We explore several types of neighborhood, including: 1) all neighbors (AN), 2) k-nearest neighbors (kNN), and 3) all neighbors weighted by the number of common neighbors (WCN). We also explore different types of similarity measures of movies, including: 1) Pearson correlation coefficient of user ratings, and 2) Jaccard index of the sets of common users (users who rate the same movies). We also consider the product of those two measures as a third similarity measure.

Since the genres classification is a multi-label classification problem, we need to predict a set of genres for each target movie. Based on the chosen type of neighborhood and similarity measure, we calculate a confidence score for all candidate genres. Then, to determine the relevant genres based on their confidence score, a thresholding strategy is needed. In this paper, we adopt a well known thresholding strategy called Rank Cut (RCut), that outputs for each target instance, a certain number of labels/genres with the highest confidence score. However, we propose a modified version of this strategy that we call Parametric Adaptive Rank Cut (PARC). In the PARC technique, the number of selected labels is dynamic and corresponds to the average number of labels within the neighborhood of the target movie, instead of the static average number of labels in the training set.

Moreover, the network of movies which we construct is actually a multilayer network, as we consider each genre as a layer comprising the movies that belong to that genre. Thus,

we can measure the similarity of layers, using different measures, and apply agglomerative clustering on the layers of this network. As a result, we obtain a comprehensible taxonomy of genres which groups together similar genres using the similarity of their movies in terms of user preferences.

The main contributions of this paper are:

- We present an approach for movie genres classification based on user ratings.
- Our approach is based on a movie-similarity network, where nodes are movies and edges are weighted with different similarity scores.
- We use and experiment with various types of neighborhoods: AN, kNN, and WCN.
- We analyze and experiment with various similarity measures, including Pearson correlation of ratings, and Jaccard index of common users.
- We propose and use a variation of RCut thresholding technique for multi-label classification, called: Parametric Adaptive Rank Cut.
- The experimental results show that our genres classification approach outperforms many existing approaches, by achieving an F1-score of 0.70.
- We apply agglomerative clustering on the layers (genres) of the network of movies, and constuct a taxonomy of genres.

The paper is organized as follows. Section 2 reviews existing related work, and Section 3 introduces some necessary preliminaries. Section 4 is dedicated to dataset, analysis, and preprocessing. Section 5 is the core part of the paper that presents our approach for genres classification and its various related aspects. Section 6 presents the experimental part including, experiments design and results. Section 7 is devoted to the the multilayer networks of movies, and the taxonomy of genres constructed using it. Section 8 concludes the paper.

## 2. Related Works

Over the years, researchers have proposed various automatic genre classification methods for movies. Proposed methods are mainly based on audio-visual features, or textual features of movies.

### 2.1. *Methods that use audio-visual features*

Rasheed and Shah proposed in [18] a method to classify movies on the basis of audio-visual cues present in the previews. Inspired by cinematic principles, Rasheed et al. [17] uses four computable video feature: average shot length, color variance, motion content and lighting key, to classify movies into four broad categories: Comedies, Action, Dramas, or Horror films. Similarly, Huang et al. [10, 9] use low-level features and visual features to classify the film genres. Brezeale and Cook [2] use closed captions for classifying movies by genre and learning user preferences, using a support vector machine as the classifier.

Roach and Mason [21] propose a classification approach of video genre using audio features. Sageder et al., [23] propose an unsupervised method for the selection of features in the context

of audio-based video genre classification. Huang and Wang [11] propose a hybrid approach that combines both low-level visual features and audio information.

Wehrmann et al., [30, 31] propose a classification method that encapsulates an ultra-deep ConvNet with residual connections. This approach extracts temporal information from image-based features prior to performing the mapping of trailers to genres.

[35] presents a method for movie genre categorization of movie trailers, based on scene categorization. The approach decomposes each trailer into a collection of keyframes through shot boundary analysis. From these keyframes, scene detectors and descriptors are used to extract features, which are then used for shot categorization via unsupervised learning, and consequently to represent trailers using a bag-of-visual-words (bovw) model with shot classes as vocabularies. Then, the genre classification task is approached by mapping bovw temporally structured trailer features to four high-level movie genres: action, comedy, drama or horror films.

### 2.2. *Methods that use textual features*

Ho [32] investigated different methods to classify movies' genres based on synopsis: one-vs-all Support Vector Machines (SVM), multi-label K-nearest neighbor (KNN), parametric mixture model (PMM) and neural network. All these methods use the term frequency inverse document frequency of the words as features. The dataset used for experiment is relatively small with only 16,000 movie titles for both the train and test sets. In addition, the experiment is limited to only predicting only 10 most popular genres, including action, adventure, comedy, crime, documentary, drama, family, romance, short films, and thrillers. Overall, SVM achieves the highest F1 score of 0.55.

Hoang [8] explores several Machine Learning methods to predict movie genres based on plot summaries. Naive Bayes, Word2Vec + XGBoost and Recurrent Neural Networks are used for text classification, while K-binary transformation, rank method and probabilistic classification with learned probability threshold are employed for the multi-label problem involved in the genre tagging task.

Blackstock and Spitz [1] attempt to classify movie scripts by building a logistic regression model using NLP-related features extracted from the scripts such as the ratio of descriptive words to nominals or the ratio of dialogues frames to non-dialogue frames. For each movie scripts, the model, based on extracted features, estimates the probability that the movie belong to each genre and takes the k best scores to be its predicted genres, where k is a hyper-parameter. The experiment is done on a small dataset with only 399 scripts and the best subset of features achieves an F1 score of 0.56.

### 2.3. *Methods that use other features*

To the best of our knowledge, Makita and Lenskiy [14] is the only work that uses user ratings to predict movie genres. They propose a Naive Bayes model to predict movie genres based on user ratings of the movie. They applied Naïve Bayes classifier, where a multinomial event model is used to estimate a likelihood of a movie given genre, and the Bayes rule to evaluate the posterior probability of a genre given a movie. However, they use a weak evaluation metric, the Prediction Rate (aka, Hit Rate) which is the percentage of movies that the model predicts correctly at least one of the true labels. That is, the prediction is considered successful

if the predicted genre matches the true one, or one of the true ones if a movie simultaneously was assigned to a number of genres. Makita's approach achieves a 70% hit-rate, whereas our approach achieves a 94% hit-rate, as we will see.

## 3. Preliminary

Let $U$ be a set of users, and $M$ set of items (movies in our case). Utility function is a function: $r : U \times M \to R$, where $R$ is a set of ratings, such that, 0–5 stars or real number in $[0, 1]$. Thus, $r(u, m)$ denotes the rating given by user $u$ to movie $m$.

Let $U_m \subseteq U$ denote the set of users who rated movie $m$. Let $\overline{r(m)}$ denote the average of ratings given to movie $m$:

$$\overline{r(m)} = \frac{1}{|U_m|} \sum_{u \in U_m} r(u, m)$$

We define two similarity measures of movies based on user ratings: (1) Pearson correlation coefficient, and (2) Jaccard index.

The first similarity measure is the Pearson correlation coefficient of the ratings given to two movies $i, j \in M$ by same users:

$$\rho(m, n) = \frac{\sum_{u \in U_{m,n}} [r(u, m) - \overline{r(m)}].[r(u, n) - \overline{r(n)}]}{\sqrt{\sum_{u \in U_{m,n}} [r(u, m) - \overline{r(m)}]^2 . \sum_{u \in U_{m,n}} [r(u, n) - \overline{r(n)}]^2}} \tag{1}$$

where $U_{m,n} = U_m \cap U_n$ is the set of users who rated both movies $m$ and $n$.

The second ratings-based movie-similarity measure is the Jaccard index of common users, which is the fraction of users who rated both $m$ and $n$ to all users who rated $m$ and/or $n$:

$$\phi(m, n) = \frac{|U_m \cap U_n|}{|U_m \cup U_n|} \tag{2}$$

Let $\mathcal{G}$ be a set of genres labels. Movie-genres function is a function: $G : M \to 2^{\mathcal{G}}$, that associates each movie $m \in M$ with one or more genres $G(m) = \{g_1, \cdots, g_{l_m}\} \subseteq \mathcal{G}$.

Inversely, genres-movies function is a function $f : \mathcal{G} \to 2^M$ that associates each genres $g \in \mathcal{G}$ with the set of movies that belong to this genres: $f(g) = \{m \mid m \in M, g \in G(m)\}$

The genres classification problem is the problem of predicting for an input movie $m \in M$, a set of genres labels $G(m) \in 2^{\mathcal{G}}$.

For analysis purposes, we define a genres-based similarity for pairs of movies as:

$$\gamma(m, n) = \frac{2|G(m) \cap G(n)|}{|G(m)| + |G(n)|} \tag{3}$$

where $G(m)$ and $G(n)$ are the sets of genres labels of movies $m$ and $n$, respectively. We only use this similarity during data preprocessing in order to properly prune the movies network to increase its utility.

## 4. Analysis

### 4.1. *Data*

We used MovieLens 20M movie ratings dataset [7] [c]. The dataset includes, among others, two tables: *ratings* and *movies*.

In *ratings* table, each entry represents one rating of one movie by one user, and has the following format: $(userId, movieId, rating, timestamp)$. This table comprises 20,000,263 ratings applied to 26,744 movies by 138,493 users. Ratings are made on a 5-star scale, with half-star increments (0.5 stars – 5.0 stars).

In *movies* table, each entry represents one movie, and has the following format: $(movieId, title, genres)$. This table comprises 27,278 movies. Movie titles include the year of release. Genres are a pipe-separated list, and are selected from the list of genres shown in Table 1. Crossed out genres are the rare ones (that appear in less than 5% of the movies), and they are pruned as we will see in Section ).

Table 1. List of genres

| | | | |
|---|---|---|---|
| Action | Adventure | ~~Animation~~ | Children |
| Comedy | Crime | Documentary | Drama |
| Fantasy | ~~Film-Noir~~ | Horror | ~~IMAX~~ |
| ~~Musical~~ | Mystery | Romance | Sci-Fi |
| Thriller | ~~War~~ | ~~Western~~ | ~~(no genres listed)~~ |

### 4.2. *Preprocessing*

#### 4.2.1. *Pruning movies with few ratings*

The first preprocessing step is to remove movies having less than 20 ratings. In the original dataset each user has at least 20 ratings; while each movie can have 1 rating. To make the data more balanced, the first preprocessing step is to prune the *ratings* table, such that it contains only the movies that have at least 20 ratings ($|U_m| < 20$). This corresponds to 67,174 entries in *ratings* table. By removing these entries, the resulting table still contain 19,933,089 entries, which is about 99.6% of original ratings. However, the number of movies is reduced to 13,132 which is about 49.1% of original movies. Table 2 shows the difference in rating per movie distribution before and after this pruning step. The number of users stays the same: $138,493$; whereas the average rating changes slightly from 3.525 to 3.527.

Table 2. Ratings per movie distribution, before and after pruning movies w. $< 20$ ratings

| | # movies | # ratings per movie | | | | | |
|---|---|---|---|---|---|---|---|
| | | avg | min | 25% | 50% | 75% | max |
| before | **26,744** | 747.8 | **1** | 3 | 18 | 205 | 67,310 |
| after | **13,132** | 1,517.9 | **20** | 63 | 215 | 931 | 67,310 |

[c] https://grouplens.org/datasets/movielens/20m/

#### 4.2.2. *Movie-movie similarity network*

The next step is to construct a movie-movie similarity network. For each pair of movies, we calculate the two ratings-based similarity measures: Pearson correlation coefficient $\rho$ (Equation. 1), and Jaccard index of common users $\phi$ (Equation. 2). We discard movie pairs that have zero or less correlation coefficient, i.e., we retain only movie pairs that have positive correlation: $\rho(m, n) > 0$. The result is a large undirected network whose nodes are movies, and the edges are equipped with two weights: correlation $\rho$, and Jaccard index $\phi$. This network consists of 13,132 nodes and 48,954,323 edges (density = 0.2839).

#### 4.2.3. *Pruning rare genres*

As mentioned above, the movies in the dataset belong to a set of 20 genres. However, the distribution of movies over genres is not balanced. Moreover, some genres are rare and only few movies belong to them. To enhance the classification results, we removed any genres that appear in less than 5% of the movies. That is, we remove any genres $g$ with $|f(g)|/|M| < 0.05$.

That way, we removed 7 genres, and retain 13 genres whose distribution is depicted in Figure 1. This causes the removal of 157 movies. With 12,975 remaining movies (98.8%), the movie-similarity network now contains 47,902,947 edges, which is about 97.85% of edges (density = 0.2846).

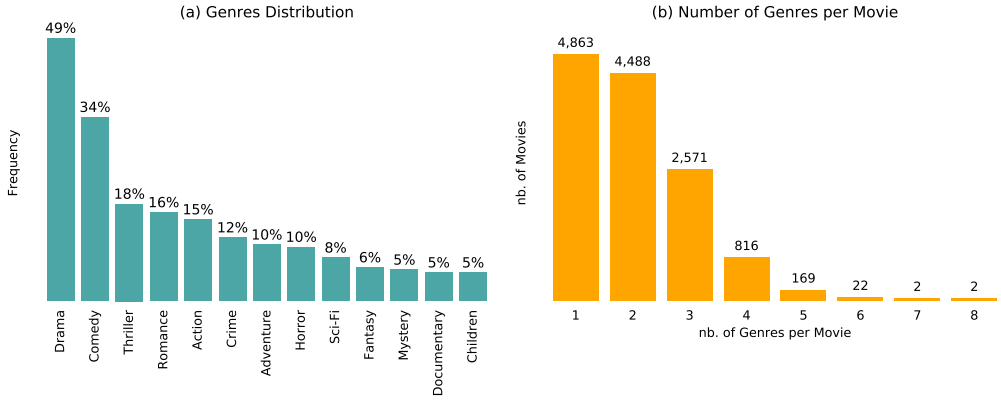



Fig. 1. Genres distribution

### 4.3. *Analysis*

Having two measures for movie similarity: correlation $\rho$ and Jaccard index $\phi$, we need to figure out the predictability power of these measures and/or their combinations with respect to predicting the movie genres. Thus, we can decide which measure or combination of measures we would use to better classify movie genres. For this purpose, we calculated the genres-based similarity for pairs of movies in our network, using Equation 3.

Having this genres-based movie similarity measure, we then find, for the whole network, the correlation of this measure with our movie similarity measures $\rho$ and $\phi$, and two combinations: their product $\rho.\phi$, and their harmonic mean: $hm(\rho, \phi) = \frac{2\rho\phi}{\rho+\phi}$. The results are shown in Table 3. We observe that the product $\rho.\phi$ has the highest correlation with $\gamma$, therefore, we consider

it as our predictor of choice. That is, we will use this measure in the classification of movie genres, as we will see in the next section.

Table 3. Correlation of movie similarity measures

| ? | $\rho$ | $\phi$ | $\rho.\phi$ | $hm(\rho, \phi)$ |
|---|---|---|---|---|
| $corr(\gamma, ?)$ | 0.0098 | 0.1052 | **0.1230** | **0.1225** |

The movie similarity network is very large, hence to make best use of it we need to prune it for two reasons:

- The large size of edges makes the computations practically unfeasible.
- Edges with low similarity values hinder the accuracy of classification.

Therefore, we need to prune the network to only retain edges that have a similarity score above certain threshold. But which measure to use for this cut, $\rho$ or $\phi$? and which threshold we should use? To answer those questions, we tried several thresholds for $\rho$ and for $\phi$. For each cut, we calculate the number of edges $E'$ and nodes $N'$ of the pruned network, and their ratios to the original network: $E'/E$ and $N'/N$. Our aim is to (i) keep (almost) all the nodes, while reducing the number of edges; and most importantly to (ii) increase the predictability power, defined as the correlation of genres-based movie similarity to our best predictor $\rho.\phi$: $corr(\gamma, \rho.\phi)$. The results of these several cuts are shown in Table 4.

Table 4. Network pruning to increase utility

| $\rho >?$ | $E'$ | % | $N'$ | % | – | $corr(\gamma, \rho.\phi)$ |
|---|---|---|---|---|---|---|
| 0.1 | 42,377,706 | 88.5 | 12,975 | 100 | 0 | 0.1255 |
| 0.2 | 34,212,821 | 71.4 | 12,975 | 100 | 0 | 0.1328 |
| 0.3 | 26,249,002 | 54.8 | 12,975 | 100 | 0 | 0.1406 |
| 0.4 | 19,937,240 | 41.6 | 12,975 | 100 | 0 | 0.1441 |
| 0.5 | 14,889,883 | 31.1 | 12,975 | 100 | 0 | 0.1436 |
| 0.6 | 11,498,906 | 24.0 | 12,975 | 100 | 0 | 0.1391 |
| 0.7 | 8,851,877 | 18.5 | 12,975 | 100 | 0 | 0.1323 |
| 0.8 | 6,756,940 | 14.1 | 12,975 | 100 | 0 | 0.1220 |
| 0.9 | 4,800,681 | 10.0 | 12,955 | 99.8 | 20 | 0.1073 |
| $\phi >?$ | $E'$ | % | $N'$ | % | – | $corr(\gamma, \rho.\phi)$ |
| 0.01 | 28,773,658 | 60.1 | 12,975 | 100 | 0 | 0.1206 |
| 0.02 | 16,445,054 | 34.3 | 12,975 | 100 | 0 | 0.1288 |
| 0.03 | 9,806,798 | 20.5 | 12,973 | 100 | 2 | 0.1434 |
| 0.04 | 6,105,477 | 12.7 | 12,965 | 99.9 | 10 | 0.1606 |
| **0.05** | **3,959,515** | **8.3** | **12,933** | **99.7** | **42** | **0.1780** |
| 0.06 | 2,652,842 | 5.5 | 12,819 | 98.8 | 156 | 0.1940 |
| 0.07 | 1,817,721 | 3.8 | 12,560 | 96.8 | 415 | 0.2103 |
| 0.08 | 1,269,675 | 2.7 | 12,106 | 93.3 | 869 | 0.2252 |
| 0.09 | 902,164 | 1.9 | 11,552 | 89.0 | 1,423 | 0.2399 |
| 0.10 | 647,856 | 1.4 | 10,806 | 83.3 | 2,169 | 0.2535 |

We can observe how increasing the threshold leads to decrease in the number of edges and eventually decrease the number of nodes. We also observe that when we cut using $\phi$,

increasing the threshold always leads to an increase in the predictability power (correlation with $\gamma$). For instance, with $\phi > 0.1$, we could achieve $> 0.25$ utility, but we will lose too many nodes (movies); in this case, more than two thousand movies.

As a reasonable choice, we decided to cut using $\phi > 0.05$. Thus, the result network contains about 4 million edges (about 8.3% of original edges), and we can achieve 0.178 utility. However, we lose 42 nodes, which means that we still retain $> 99\%$ of original nodes.

## 5. Classification

The idea behind the classification of movie genres is to use the similarity between movies based on user ratings, to predict the genres of movies. The assumption is the following: a user may like/prefer movies of certain genres, and hence give them high ratings. Thus, movies that belong to the same genres will be rated by the set of users who like/prefer this genres. In other words, movies of the same genres would be similar to each other in terms of user ratings (correlation $\rho$), and in terms of common users (Jaccard index $\phi$). Therefore, we will classify movie genres based on the genres of their similar movies, that is, the movies that are neighbors to the target movie within the similarity network. However, two questions still need to be answered:

1. what exactly this neighborhood should be?
2. once we determine the neighborhood, how to predict the genres of target movie based on the genres of neighbor movies?

### 5.1. *Defining Neighborhood*

To answer the first question, we will use four different, but similar, definitions of neighborhood. Each definition would give us a different classification approach.

Let $\Gamma(m)$ be the set of neighbor nodes of $m$, and let $\Gamma(m, n)$ be the set of common neighbor nodes of $m$ and $n$. Let $S_{m,n}$ denote the similarity between the nodes $m$ and $n$, that is, $S_{m,n} = \rho(m, n).\phi(m, n)$. As shown in Figure 5, for a target node (movie), the neighborhood can be one of the following three sets:

1. **AN**: all neighbors of the target node.

$$N_{\text{AN}}(m) = \{\langle n, S_{m,n}\rangle \mid n \in \Gamma(m)\}$$

2. **kNN**: k-nearest neighbors, a subset of (predefined) size $k$ of nearest (most similar) nodes to the target node.

$$N_{\text{kNN}}(m) = \{sort(N_{\text{AN}}(m))\}_{i=1}^{k}$$

3. **WCN**: all neighbors weighted by the number of common neighbors of the target node.

$$N_{\text{WCN}}(m) = \{\langle n, S_{m,n}.(p + q\ .\ |\Gamma(m, n)|)\rangle \mid n \in \Gamma(m)\}$$

where $p$ and $q$ are two control parameters.

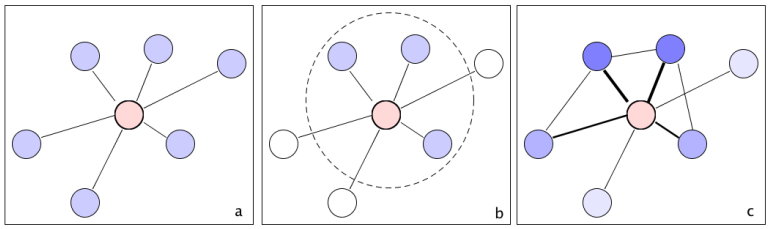



Table 5. Neighborhoods: (a) AN, (b) kNN, (c) WCN.

### 5.2. *Predicting Genres*

The second question concerns the prediction method of genres of a target movie based on the genres of its neighbors. Since each movie could have more than one genres, the problem is a multi-label classification problem.

For each movie $m \in I$, we calculate a confidence score for each candidate genres $g \in \mathcal{G}$. Then, we use a thresholding technique to determine whether a genres is relevant to the target movie; that is, to obtain a bipartition of the labels into relevant and irrelevant with respect to the target movie.

The confidence score of a movie $m \in I$ with respect to a genres $g \in \mathcal{G}$ is given by:

$$s_{m,g} = score(m \mid g) = \frac{1}{|N_*(m)|} \sum_{n \in N_*(m)} \gamma(n,g).S_{m,n}$$

where $N_*(m)$ is a chosen neighborhood of $m$, $S_{m,n}$ is the similarity of $m$ and $n$, and $\gamma(n,g)$ indicates whether $g$ is genres of $n$; that is, $\gamma(n,g) = 1$ if $g \in G(n)$, 0 otherwise.

When the neighborhood is WCN, we also incorporate the number of common neighbors as a factor. We use two free parameters $p$ and $q$, to control the participation of number of common neighbors $|N(m,n)|$ in the movie-genres score, as follows:

### 5.3. *Thresholding Strategies*

Once we obtain confidence scores of genres per test movie, we need a strategy to determine relevant genres.

Thresholding method is a general solution to multi-labeled classification for a large group of classification algorithms which yield a confidence score for each candidate class. Yang [33] examines three common thresholding strategies: RCut (rank-based thresholding), SCut (score-based local optimization), and PCut (proportion-based assignments). Ioannou et al., [13] provide an interesting comparative study of six thresholding methods, including the three methods above, and: OneThreshold (OT) [28, 27, 19, 20] MetaLabeler [26], and Threshold-Prediction (TP) [4].

Among these different thresholding strategies, we adopt Rank-Cut (RCut) strategy. RCut [33] is one of the simplest approaches for obtaining a bipartition from a scores vector. For each new instance, it outputs the $t$ labels with the highest score; i.e., top $t$ labels. The parameter $t$ can be either specified by the user or automatically tuned using a validation set or $k$-fold cross-validation [33]. In the former case, a logical value is the average number of labels in the training set [26], also called *label cardinality*.

However, we modify this strategy as follows. Instead of considering a single parameter $t$ for all test instances, we use an adaptive parameter $t_m$ that differs from one instance to another. That is, instead of taking the average number of labels in the training set, we take the average number of labels in the neighborhood ($N(m)$) of instance $m$. Moreover, we multiply this average by a constant parameter $\alpha$. We call this technique *Parametric Adaptive Rank Cut* (PARC). Specifically, for an instance $m$, we take the top $t_m$ labels with the highest confidence score, where $t_m$ is given by:

$$t_m = \lfloor\ \alpha.\overline{G_{N(m)}}\ \rceil \tag{4}$$

where $\alpha \geq 1$ is a constant parameter, $\overline{G_{N(m)}}$ is the average number of genres for movies in the neighborhood of $m$:

$$\overline{G_{N(m)}} = \frac{1}{|N(m)|} \sum_{n \in N(m)} |G(n)|$$

and $\lfloor . \rceil$ denotes rounding to the nearest integer.

The idea of dynamic thresholding given a new unlabeled instance is actually not that new. MetaLabeler [26], and ThresholdPrediction (TP) [4] both use a (instance-based) dynamic thresholding. However, TP is a score-based strategy rather than rank-based. MetaLabler require an additional learning step to learn a model for dynamically predicting the size $t$ of the set of relevant labels. Moreover, Spyromitros et al. [25] propose BRkNN, an adaptation of the kNN algorithm for multilabel classification that uses Binary Relevance transformation. One of their extensions to this method, called BRkNN-b, uses the average size of the label sets of the $k$ nearest neighbors as a rank cut. PARC technique that we use is very similar to this extension, however, we use an additional parameter $\alpha$ to further tune the rank cut. Actually, BRkNN-b is equivalent to PARC for $\alpha = 1$.

### 5.4. *Evaluation Metrics*

In traditional classification such as multi-class problems, there exists a set of standard evaluation metrics, including: accuracy, precision, recall, F-measure, and ROC area defined for single-label multi-class classification problems [24]. However, in multi-label classification, predictions for an instance is a set of labels and, therefore, the prediction can be fully correct, partially correct (with different levels of correctness) or fully incorrect. None of these existing evaluation metrics capture such notion in their original form. This makes evaluation of a multi-label classifier more challenging than evaluation of a single label classifier.

To capture the notion of partially correct, one strategy is to evaluate the average difference between the predicted labels and the actual labels for each test instance, and then average over all instances in the test set. This approach is called *instance based* evaluations. Seemingly, one could define a *label based* evaluation where each label is evaluated first and then averaged over all labels. Since we are not interested in evaluating genres labels individually, we opt to use *instance based* evaluations, namely: recall, precision, and F-measure.

Let $D \subset M$ denote the test dataset, and let $m \in D$ be a test instance, then $G(m)$ denotes the true genres labels of $m$, and $Q(m)$ denotes the predicted genres labels.

The recall of instance $m$ is the fraction of correctly predicted labels for $m$ to all true labels of $m$. The overall *Recall* is the average per-instance recall over the entire test-set:

$$R = \frac{1}{|D|} \sum_{m \in D} R(m) = \frac{1}{|D|} \sum_{m \in D} \frac{|G(m) \cap Q(m)|}{|G(m)|} \tag{5}$$

The precision of instance $m$ is the fraction of correctly predicted labels to all predicted labels: The overall *Precision* is the average per-instance precision over the entire test-set:

$$P = \frac{1}{|D|} \sum_{m \in D} P(m) = \frac{1}{|D|} \sum_{m \in D} \frac{|G(m) \cap Q(m)|}{|Q(m)|} \tag{6}$$

The *F-measure* (also called the F1 score) is the harmonic mean of recall and precision:

$$F = \frac{2\ P.R}{P + R} \tag{7}$$

We have mentioned earlier that some related work, such as [14], uses a simple measure called the *Prediction Rate* (aka, Hit Rate) as an evaluation metric. This metric is *weak* comparing to recall, precision and F-measure, as it considers a prediction successful if the predicted label matches at least one correct label. In other words, Hit-Rate considers any partially correct predication as fully correct.

$$HR = \begin{cases} 1 & \text{if } |T \cap P| > 0 \\ 0 & \text{otherwise} \end{cases}$$

## 6. Experiments

### 6.1. *Experiments Design*

#### 6.1.1. *Training-Test Split.*

We split the set of movies (12,933 movies) into training-test datasets. A random sample of 20% of movies is chosen as a test set (contains 2,586 movies), whereas the remaining movies (10,347) are used as training set. The movie similarity network covers both training and test movies. Thus, for classifying a test instance, when we take its neighborhood, we only consider the neighbors that belong to the training set, and discard those that belong to test set, if any.

#### 6.1.2. *Neighborhoods.*

We use three types of neighborhoods as discussed earlier: (i) AN: all neighbors, (ii) kNN: k nearest neighbors, (iii) WCN: weighted common neighbors. For kNN neighborhood, we use different values for $k$, ranging from 5 to 50, with 5 increments. For WCN neighborhood, we use $p = 1$ and $q = 0.5$ as these values perform better than others according to our pilot experiments.

We have also experimented with a fourth type of neighborhood, which is SN: the strong neighbors. This is the set of neighbors that have a common neighbors with the target movie. $N_{\text{SN}}(m) = \{\langle n, S_{m,n}\rangle \mid n \in \Gamma(m), |\Gamma(m,n)| > 0\}$. However, the behavior of this neighborhood type is very close to the behavior of the first type AN; therefore, we do not include it in the results.

6.1.3. *Similarity Measures.*

As a baseline, we use the two similarity measures described in Section , namely: Pearson correlation coefficient of ratings $\rho$, and Jacard index of common users $\phi$. Moreover, we use a third similarity measure which is the product of Pearson correlation $\rho$ and Jacard index $\phi$. Thus, in our experiments we will compare the performance of those three similarity measures:

$$\begin{aligned} sim_1(m,n) &= \rho(m,n) \\ sim_2(m,n) &= \phi(m,n) \\ sim_3(m,n) &= \rho(m,n).\phi(m,n) \end{aligned} \tag{8}$$

We have also experimented with a fourth measure, which is the harmonic mean of $\rho$ and $\phi$: $sim_4 = \frac{2\rho\phi}{\rho+\phi}$. However, the behavior of this measure is very close to the behavior of the third measure (the product $\rho.\phi$); therefore, we do not include it in the results.

6.1.4. *Thresholding Strategy.*

We use Parametric Adaptive Rank Cut (PARC) thresholding technique, with different values of parameter $\alpha$, ranging from 1 to 1.5 with 0.1 increments.

We have also experimented with other threshoding strategies, such as classic RCut, and OneThreshold (OT). However, we found that PARC technique gives better performance than the other strategies, therefore, we report about PARC technique only in our results.

### 6.2. *Results*

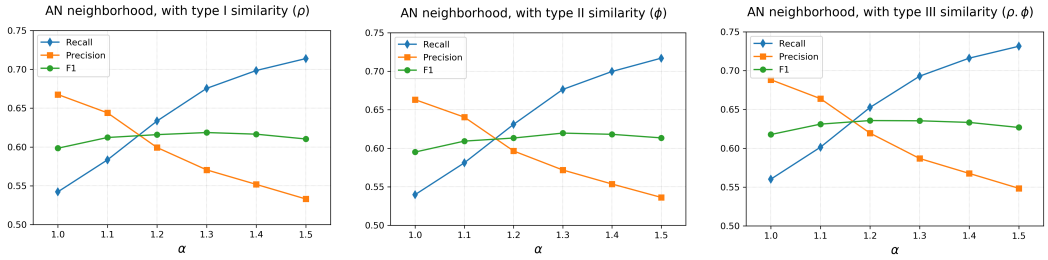
In this section, we present the results of our experiments, where we address each type of neighborhood separately (AN, kNN, and WCN). For each type of neighborhood, we compare the classification performance over the different similarity measures, and different values of parameters ($\alpha$, and $k$). Later, we compare the performance over the three types of neighborhoods.

6.2.1. *Neighborhood AN*

In this type, the neighborhood of a target movie comprises all of its neighbor movies as by the movie similarity network. Three similarity measures are used to compute the confidence scores: $sim_1 : \rho$, $sim_2 : \phi$, and $sim_3 : \rho.\phi$.

Figure 2 shows, for each similarity measure, the classification performance in terms of recall, precision and F-measure, over different values of $\alpha$, the parameter of PARC thresholding technique.

Our main observation is regarding the impact of $\alpha$ parameter on the recall and precision. We clearly observe that increasing $\alpha$ value will increase the recall and decrease the precision, regardless of the similarity type. To explain this behavior, it is sufficient to notice that increasing $\alpha$ implies an increase of the adaptive rank cut parameter $t_m$ as per Equation 4; and consequently an increase in the size of the set of predicted genres labels $Q(m)$, as more genres labels are considered relevant. Hence, with a larger set $Q(m)$, the set of correctly predicted genres $G(m) \cap Q(m)$ would eventually become larger, as well as the numerator of Equation 5, which means a higher recall. Moreover, this implies an increase of the denominator of Equation 6, which means a lower precision.




(a) Similarity 1: $\rho$ (b) Similarity 2: $\phi$ (c) Similarity 3: $\rho.\phi$

Fig. 2. Neighborhood AN

The second observation is that F-measure has almost a stable behavior with respect to $\alpha$. This is a direct consequence of the contrary behavior of increasing recall and decreasing precision. However, we observe that the F-measure slightly increases near the intermediate values of $\alpha$, e.g., 1.2 and 1.3, and then decreases. Hence, the performance with intermediate values of $\alpha$ is better than low values (e.g., 1), or high values (e.g., 1.5).

Table 6 and Figure 3 depict the classification performance of the three similarity measures in terms of F-measure, over the different values of $\alpha$. First, we observe that the third type of similarity $sim_3 : \rho.\phi$ has a better performance than the other baseline similarity types, for any value of $\alpha$. The best performance can be achieved by the third type of similarity at $\alpha = 1.2$, where the F-measure is 0.6357. Moreover, we observe that the first two similarity types have similar performances. However, $sim_1$ is better than $sim_2$ for $\alpha \leq 1.2$, while $sim_2$ is better for $\alpha \geq 1.3$.

Table 6. AN neighborhood, F-measure

| $\alpha$ | 1.0 | 1.1 | 1.2 | 1.3 | 1.4 | 1.5 |
|---|---|---|---|---|---|---|
| $sim_1$ | 0.5984 | 0.6122 | 0.6159 | 0.6186 | 0.6165 | 0.6103 |
| $sim_2$ | 0.5951 | 0.6093 | 0.6133 | 0.6197 | 0.6181 | 0.6135 |
| $sim_3$ | 0.6177 | 0.6311 | **0.6357** | 0.6354 | 0.6333 | 0.6268 |

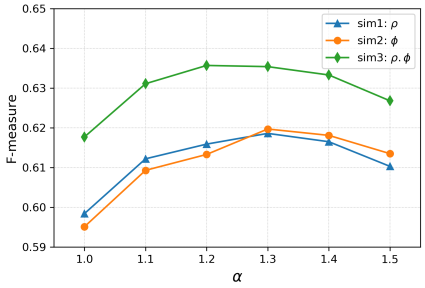



Fig. 3. Neighborhood AN, comparing similarity types

### 6.2.2. *Neighborhood kNN*

In this type, the neighborhood of a target movie comprises the top $k$ nearest neighbor movies

(with highest similarity scores) as by the movie similarity network. Here also, we consider the three similarity measures to compute the confidence scores: $sim_1 : \rho$, $sim_2 : \phi$, and $sim_3 : \rho.\phi$. For parameter $k$, we range the value from 5 to 50, with increments of 5.

Figure 4 shows the classification performance of the three similarity types, in terms of F-measure, for different values of $\alpha$, over the range of $k$ values. Inversely, Figure 5 shows the F-measure of the three similarity types for different values of $k$, over the range of $\alpha$ values [d].

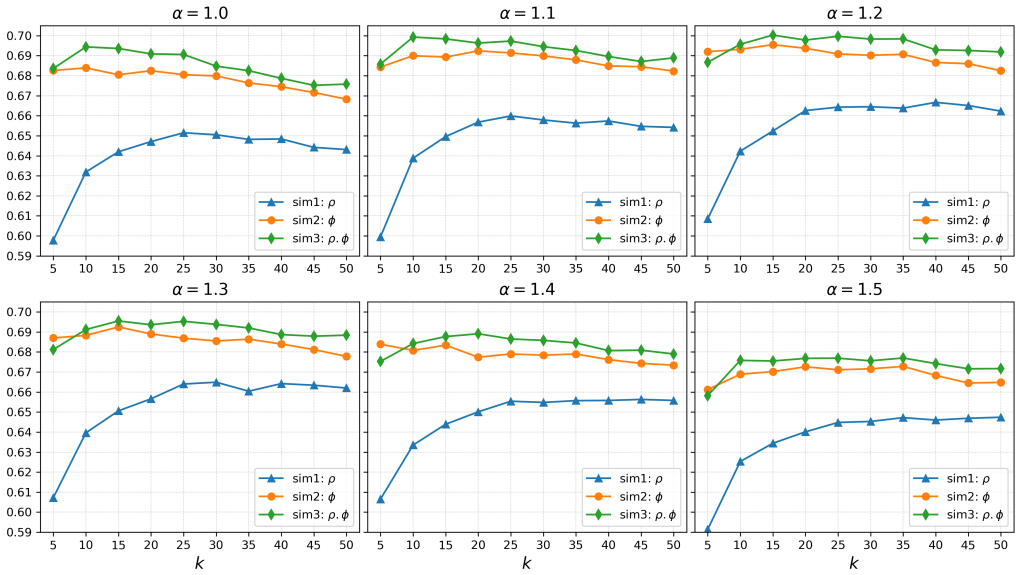



Fig. 4. kNN, performance over $k$ for different $\alpha$ values

We observe that, for almost all values of $\alpha$ and $k$, the third similarity type $sim_3$ has the best performance, followed by $sim_2$ which is slightly weaker than $sim_3$ but much better than $sim_1$ which has the lowest performance. The only exception is when $k = 5$ and $\alpha \geq 1.2$; in this case, $sim_2$ has a better performance than $sim_3$.

We also observe that $\alpha = 1.2$ gives the best performance for all similarity types regardless of $k$ value. Therefore, we show in Table 7 the values of F-measure for the three similarity types and all $k$ values. We find that $sim_2$ and $sim_3$ have their best performance at $k = 15$; while $sim_1$ has its best at $k = 40$. The highest F-measure value is about 0.70 and it is achieved by $sim_3$ at $k = 15$.

[d]Charts for $k \geq 35$ are omitted for sake of brevity

Table 7. kNN, F-measure over $k$ for $\alpha = 1.2$

| $k$ | $sim_1$ | $sim_2$ | $sim_3$ |
|---|---|---|---|
| 5 | 0.6085 | 0.6920 | 0.6867 |
| 10 | 0.6423 | 0.6932 | 0.6957 |
| 15 | 0.6524 | **0.6955** | **0.7003** |
| 20 | 0.6626 | 0.6937 | 0.6978 |
| 25 | 0.6643 | 0.6909 | 0.6997 |
| 30 | 0.6645 | 0.6902 | 0.6983 |
| 35 | 0.6638 | 0.6907 | 0.6984 |
| 40 | **0.6667** | 0.6866 | 0.6929 |
| 45 | 0.6651 | 0.6860 | 0.6926 |
| 50 | 0.6623 | 0.6825 | 0.6919 |

Moreover, we observe that increasing $k$ up to certain limit enhances the performance, but after such a limit, larger $k$ values start to give lower performance.

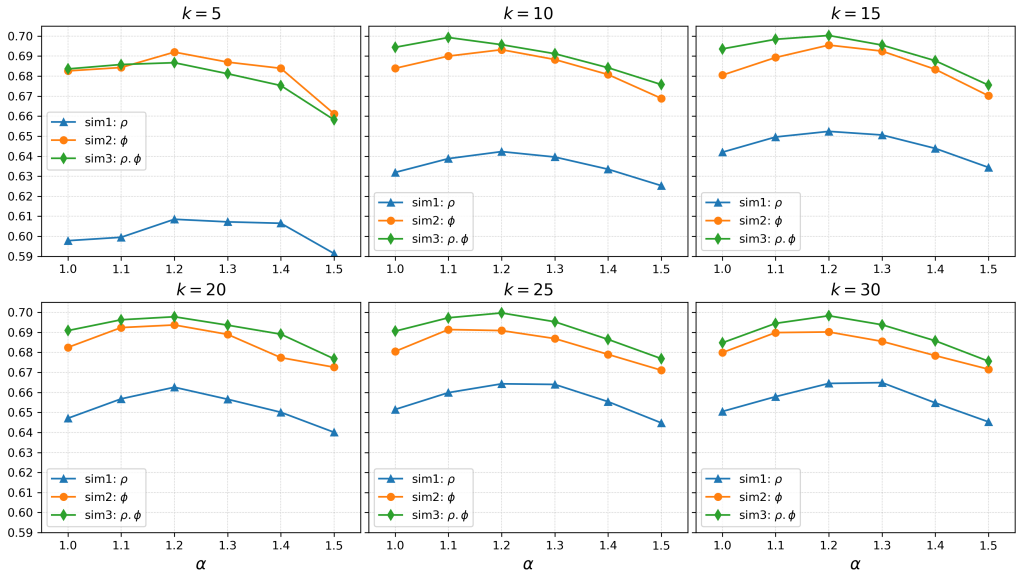



Fig. 5. kNN, performance over $\alpha$ for different $k$ values

Table 8 depicts the F-measure values for the three similarity types for $k = 15$ and $k = 40$, for all values of $\alpha$.

Table 8. kNN, F-measure over $\alpha$ for $k = 15$ and $k = 40$

| | $k = 15$ | | | $k = 40$ | | |
|---|---|---|---|---|---|---|
| $\alpha$ | $sim_1$ | $sim_2$ | $sim_3$ | $sim_1$ | $sim_2$ | $sim_3$ |
| 1 | 0.6420 | 0.6805 | 0.6936 | 0.6484 | 0.6745 | 0.6787 |
| 1.1 | 0.6496 | 0.6893 | 0.6984 | 0.6574 | 0.6849 | 0.6896 |
| 1.2 | 0.6524 | 0.6955 | **0.7003** | 0.6667 | 0.6866 | 0.6929 |
| 1.3 | 0.6506 | 0.6925 | 0.6955 | 0.6642 | 0.6840 | 0.6887 |
| 1.4 | 0.6439 | 0.6834 | 0.6877 | 0.6558 | 0.6762 | 0.6807 |
| 1.5 | 0.6344 | 0.6702 | 0.6755 | 0.6460 | 0.6683 | 0.6742 |

Figures 6 and 7 show the behavior of recall, precision, and F-measure for $sim_3$. Figure 6 shows the performance over $\alpha$ for different $k$ values; whereas Figure 6 shows the performance over $k$ for different $\alpha$ values. We observe that these performance metrics, at a given $\alpha$, change slightly over $k$. However, for a given $k$, they change drastically over $\alpha$.

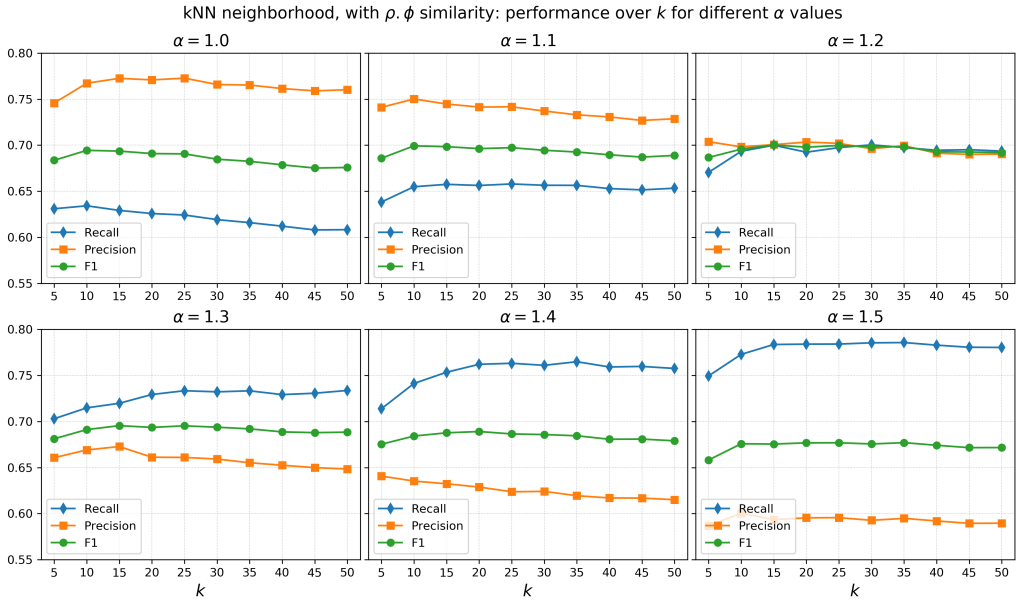



Fig. 6. kNN, $sim_3$ performance over $\alpha$ for different $k$ values

Similar to what we saw in AN neighborhood, increasing $\alpha$ leads to increase the recall, and decrease the precision. For instance, when $\alpha = 1$, the precision is much higher than the recall. When $\alpha$ increases, the precision decreases and the recall increases until they become very close at $\alpha = 1.2$. Finally, when $\alpha = 1.5$, the recall becomes much higher than the precision.

### 6.2.3. *Neighborhood WCN*

In this type, the neighborhood of a target movie comprises all of its neighbor movies, but each neighbor is weighted by the number of common neighbors with the target node. Formally, the similarity between a movie $m$ and one of its neighbors $n \in \Gamma(m)$, is modified by multiplying it with $p + q|\Gamma(m,n)|$, hence the similarity $S_{m,n}$ becomes $S_{m,n}(p + q|\Gamma(m,n)|)$, where $\Gamma(m,n)$ is the number of common neighbors of $m$ and $n$, and $p$ and $q$ are two parameters. The original similarity $S_{m,n}$ is one of the three types discussed earlier: $\rho$, $\phi$, or $\rho.\phi$. In pilot experiments, we found that $p = 1$ and $q = 0.5$ give the best performance comparing to other values.

Figure 8 shows, for each similarity measure, the classification performance in terms of recall, precision and F-measure, over different values of $\alpha$.

Here again we observe the impact of $\alpha$ parameter on the recall and precision. That is, increasing $\alpha$ value will increase the recall and decrease the precision, for all similarity types.

The second observation is that F-measure has almost a stable behavior with respect to $\alpha$. This is a direct consequence of the contrary behavior of increasing recall and decreasing precision. However, we observe that the F-measure slightly increases near the intermediate

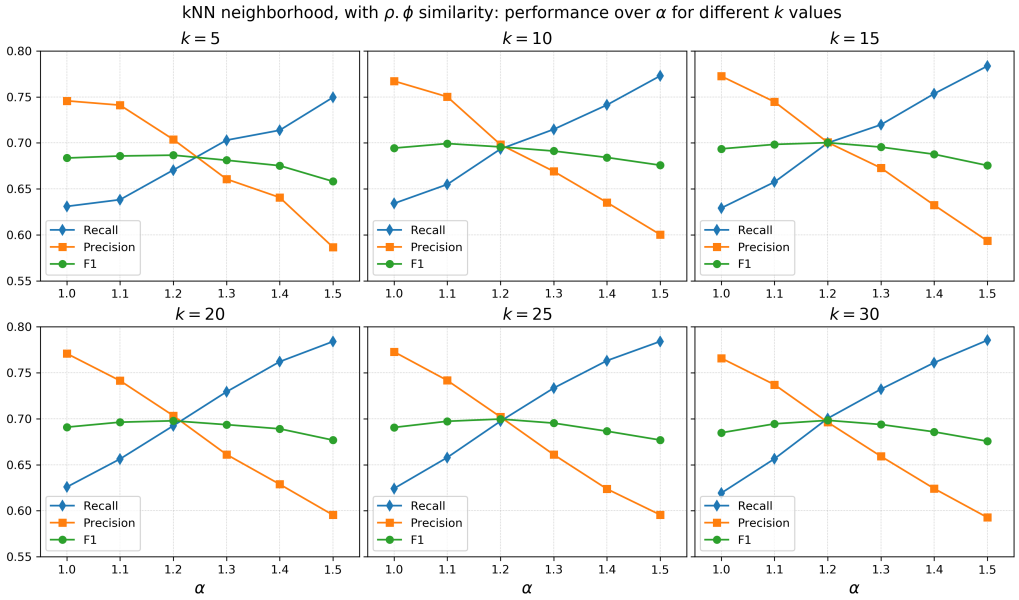
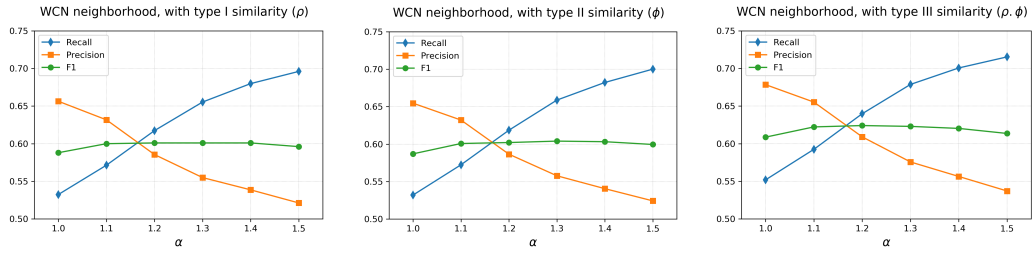



Fig. 7. kNN, $sim_3$ performance over $k$ for different $\alpha$ values




(a) $sim_1 : \rho$ (b) $sim_2 : \phi$ (c) $sim_3 : \rho.\phi$

Fig. 8. Neighborhood WCN

values of $\alpha$, e.g., 1.2 and 1.3, and then decreases. Hence, the performance with intermediate values of $\alpha$ is better than low values (e.g., 1), or high values (e.g., 1.5).

Table 9 and Figure 9 depict the performance in terms of F-measure of the three similarity types over the different values of $\alpha$. Again, we observe that the third type of similarity $sim_3 : \rho.\phi$ has the best performance that can be achieved at $\alpha = 1.2$, where the F-measure is 0.6241. We also observe that the first two similarity types have similar performances. However, $sim_1$ is better than $sim_2$ for $\alpha \leq 1.1$.

### 6.3. *Comparing Neighborhoods*

As a comparison of the three types of neighborhoods: AN, kNN, and WCN, Figure 10 shows the performance, in terms of F-measure, of those types for the three similarity types. For parameter $k$ of kNN, we selected the two values that give the best performance, namely 15 and 40. Table 10 depicts the values of F-measure at two values of $\alpha$: 1.2 and 1.3.

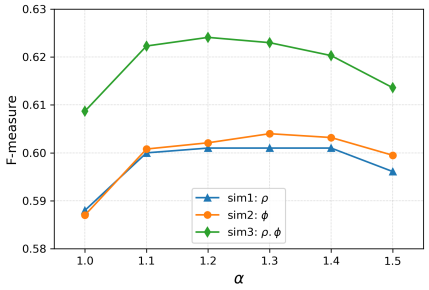



Fig. 9. Neighborhood WCN

Table 9. WCN neighborhood, F-measure

| $\alpha$ | 1.0 | 1.1 | 1.2 | 1.3 | 1.4 | 1.5 |
|---|---|---|---|---|---|---|
| $sim_1$ | 0.5880 | 0.6000 | 0.6010 | 0.6010 | 0.6010 | 0.5961 |
| $sim_2$ | 0.5870 | 0.6008 | 0.6021 | 0.6040 | 0.6032 | 0.5995 |
| $sim_3$ | 0.6087 | 0.6223 | **0.6241** | 0.6230 | 0.6203 | 0.6136 |

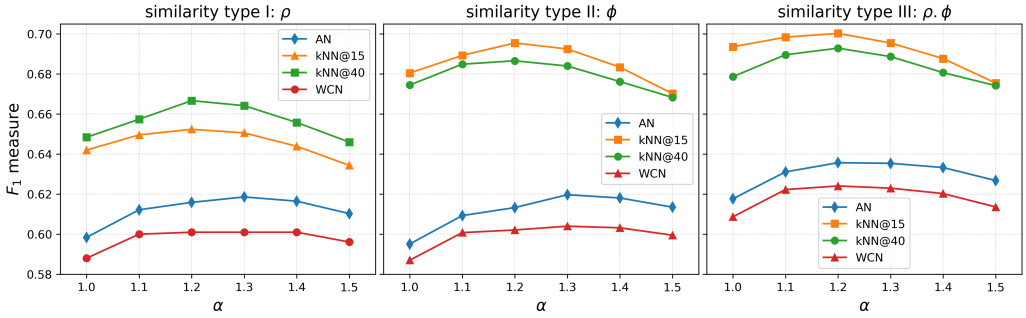



Fig. 10. Comparison of neighborhood types; $\alpha = 1.2$

We observe that kNN neighborhood outperforms the others for all similarity types, and for all values of $\alpha$. Moreover, for the first type of similarity: $sim_1 : \rho$, the best performance of kNN is achieved at $k = 40$; whereas for the other two types of similarity, the best performance is achieved at $k = 15$. We also observe that AN neighborhood is better than WCN, for all similarity types, and for all values of $\alpha$. This means that incorporating the number of common neighbors does not improve the performance.

Table 10. Performance of all neighborhood types

| $\alpha$ | | AN | kNN@15 | kNN@40 | WCN |
|---|---|---|---|---|---|
| 1.2 | $sim_1$ | 0.6159 | 0.6524 | 0.6667 | 0.6010 |
| | $sim_2$ | 0.6133 | 0.6955 | 0.6866 | 0.6021 |
| | $sim_3$ | 0.6357 | **0.7003** | 0.6929 | 0.6241 |
| 1.3 | $sim_1$ | 0.6186 | 0.6506 | 0.6642 | 0.6010 |
| | $sim_2$ | 0.6197 | 0.6925 | 0.6840 | 0.6040 |
| | $sim_3$ | 0.6354 | 0.6955 | 0.6887 | 0.6230 |

To summarize, Table 11 depicts the settings that give the overall best performance. The

best value of F-measure is about 0.70, which is achieved using kNN neighborhood, the third type of similarity $\rho.\phi$, at $k = 15$, and $\alpha = 1.2$.

Table 11. Overall best performance

| | |
|---|---|
| neighborhood type: | kNN |
| similarity type: | $sim_3 : \rho.\phi$ |
| parameters: | $k = 15$ |
| | $\alpha = 1.2$ |
| F-measure: | 0.7003 |

In order to compare our approach to other related work, we also calculated the Hit-Rate for these settings. Our best prediction rate is 94%.

Finally, table 12 show a comparison of our approach with some of the related work approaches (we compare with approaches that use textual features or user ratings). It is clear how our classification approach outperforms the others in terms of F-score as well as Hit-Rate. In particular, [14] is the only work that is based on user ratings, and it achieves a 70% hit rate, whereas our approach achieves 94% hit rate.

Table 12. Comparison with other related approaches. *: textual-features-based, **: user-ratings-based

| | approach | # movies | | # labels | F1 | HR |
|---|---|---|---|---|---|---|
| | | train | test | | | |
| * | Ho [32] | 12,800 | 3,200 | 10 | 0.55 | na |
| | Blackstock [1] | 359 | 40 | 22 | 0.55 | na |
| | Hoang [8] | 204,682 | 51,171 | 20 | 0.56 | 82% |
| ** | Makita [14] | 3,952 | | na | na | 70% |
| | ours | 10,347 | 2,586 | 13 | **0.70** | **94%** |

## 7. Taxonomy of Genres

### 7.1. *Movie Network as a Multilayer Network*

One of the interesting advantages of the movie-movie network that we constructed based on the similarity of movies with respect to the user ratings, is the fact that this network is actually a multilayer network as we take the genres of movies into consideration. Given the 13 genres to which the movies belong, when we project the movie-movie network on one of those genres, by retaining only the nodes (movies) that belong to that genre, we obtain a layer corresponding to that genre. Thus, performing this projection on all the 13 genres we obtain a multilayer network, where each layer corresponds to a single genre, the nodes are the movies of that genre, and two movies are connected if they are similar to each other with respect to user ratings, i.e., their similarity exceeds a certain threshold (in our case, $\phi > 0.05$). In this sense, each layer is a subgraph, of the original network, induced by nodes based on the genre attribute, while the edges are invariant across the layers.

Figure 11 shows a small portion of the multilayer networks, with three layers, representing the three genres *Thriller*, *Action* and *Sci-Fi*, and six nodes, representing the movies *Batman* (1989), *Independence Day* (1996), *Braveheart* (1995), *Jurassic Park* (1993), *The Matrix*

(1999), and *Gladiator* (1992). We can see that each layer contains the movies that belong to the genre of that layer. Two movies are connected by an edge if they are similar in terms of user ratings. Clearly, when two movies belong to the same layer (have the same genre), this does not necessarily imply that they should be connected by an edge. For example, the movies *Gladiator* and *Braveheart* both belong to *Action* layer, but they are not connected, which indicates that they are not similar to each other in terms of user ratings.

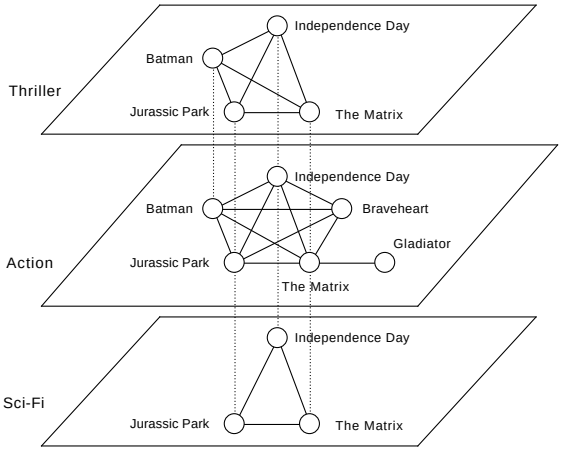



Fig. 11. A portion of the movie-movie multilayer network

In our case, not all the edges in the original network are between movies of the same genre, that is, there are many edges who connect movies of different genres, hence, belonging to different layers (inter-layer edges). However, we are only interested in the intra-layer edges, i.e., edges that connect movies within the same layer (of the same genre). Out of about 4 million edges of the original network, there are 2,532,993 of those intra-layer edges (64%).

For each layer, we extracted the following features: (1) number of nodes and edges, (2) average edge weight (using the first type of movie similarity, $\rho$), (3) number of connected components (CC), and (4) density. Table 13 shows the results, sorted by the number of edges in descending order. We can see that the largest layer is *Drama* followed by *Comedy*; whereas the smallest layers are *Children*, *Mystery* and *Documentary*. Figure 12 (left) shows a scatter plot of the different layers (genres) with respect to the number of nodes and number of edges, where their Pearson correlation coefficient is 99%.

We can also see that the average of edge weights ranges between 0.27 (for *Adventure* layer) and 0.47 (for *Horror* and *Documentary* layers. We can interpret this as follows. Movies of *Horror* genre (analogously, *Documentary* genre) are more similar to each other, in terms of user ratings, than how the movies of other genres are similar. In other words, this feature (the average of edge weights) of a layer is an indication of the *loyalty* of the users to their genres. For instance, users who like *Horror* movies do like this genre to an extent higher than how the users of other genres, say *Fantasy*, do like that genre. This aspect is also reflected by the number of connected components. Figure 12 (right) shows a scatter plot of the different layers (genres) with respect to the number of connected components (CC) and the average of

Table 13. Characteristics of the 13 layers of the movie multilayer network

| Layer/Genre | Nodes | Edges | Avg. Weight ($\rho$) | CC | Density |
|---|---|---|---|---|---|
| Drama | 6,454 | 1,115,191 | 0.31 | 25 | 0.054 |
| Comedy | 4,512 | 629,224 | 0.29 | 17 | 0.062 |
| Thriller | 2,387 | 302,389 | 0.29 | 17 | 0.106 |
| Action | 1,999 | 253,149 | 0.29 | 21 | 0.127 |
| Romance | 2,172 | 164,429 | 0.30 | 31 | 0.070 |
| Horror | 1,329 | 134,580 | 0.47 | 3 | 0.152 |
| Adventure | 1,377 | 110,574 | 0.27 | 49 | 0.117 |
| Crime | 1,557 | 103,875 | 0.28 | 27 | 0.086 |
| Sci-Fi | 1,061 | 82,304 | 0.30 | 15 | 0.146 |
| Fantasy | 822 | 36,848 | 0.28 | 22 | 0.109 |
| Children | 696 | 32,829 | 0.36 | 10 | 0.136 |
| Mystery | 766 | 28,445 | 0.29 | 22 | 0.097 |
| Documentary | 691 | 12,807 | 0.47 | 22 | 0.054 |

edge weights. We can see an inverse relation between those features (a negative correlation of -56%). A genre with a high average edge weight, such as *Horror*, also exhibits a low number of connected components, which also means that the movies of this genre are grouped in a smaller number of groups comparing to the movies of other genres. On the other extreme, a genre with a low average edge weight, such as *Adventure*, exhibits a high number of connected components.

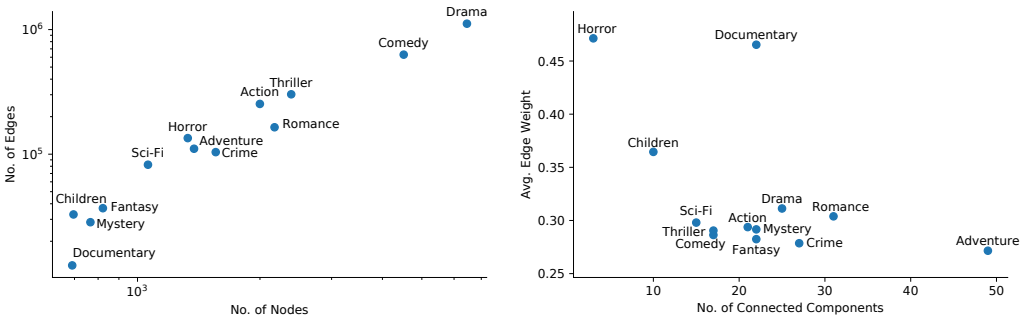



Fig. 12. Correlation of number of nodes and edges (left), and number of connected components and average edge weights (right) over the layers of the movie multilayer network.

### 7.2. *Measuring the Similarity of Layers*

With the movie multilayer network at hand, we would like next to arrange the layers (genres) of this network in such a way that reveals the similarities of the genres in terms of user preferences. To do so, we need first to be able to measure the similarity between any pair of layers.

In the literature, several works have addressed the problem of quantifying the layer similarity in multiplex networks ([34, 3]). For our studied multilayer network of movies, we opted to use three measures of layer similarity:

- Jaccard similarity of nodes (NJ).

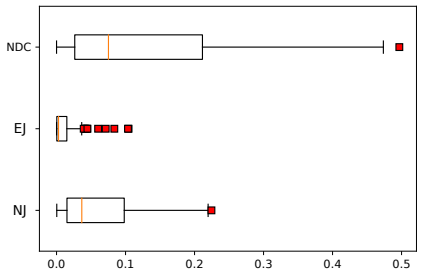



Fig. 13. Distribution of layer similarity values, using the three measures: NJ, EJ, and NDC.

- Jaccard similarity of edges (EJ).
- Cosine similarity of node degree vectors (NDC).

For the third measure, NDC, we constructed the node degree vectors taking the weights of edges into consideration. More precisely, in a layer, the degree of each node is calculated as the sum of weights of the edges incident to that node. Here, we used the first type of movie similarities, $\phi$, as edge weight.

Figure 13 shows the distribution of layer similarity values, using the three measures: NJ, EJ, and NDC. We can see that EJ generally provide low values of similarities, while NDC provides higher values, and NJ lays in between. Despite that, the similarity values of the different measures are strongly correlated. The correlation between NJ and EJ is 0.90; whereas the correlation of NDC with NJ is 0.95, and with EJ is 0.88.

Using the similarity measures between layers, we can construct a *taxonomy* of layers which groups layers over a variety of scales by creating a cluster tree or dendrogram. Such a taxonomy can be constructed using *agglomerative hierarchical clustering* [22, 15] of layers, which is a "bottom-up" approach: each layer starts in its own cluster, and pairs of clusters are merged as one moves up the hierarchy.

First, a clustering algorithm needs a distance measure instead of a similarity measure. We can easily convert our similarity measures into distance measures by subtracting their values from 1, i.e.: $dist_*(.,.) = 1 - sim_*(.,.)$. This is possible because all the similarity measures have their values in the range $[0, 1]$. The values of distance measures are in the range $[0, 1]$ too, but low values indicate similar items whereas high values indicate dissimilar items (layers). A clustering algorithm also needs a distance measure between clusters in terms of a *linkage* criterion, which determines the distance between clusters (sets of elements) as a function of the pairwise distances between elements. Some commonly used linkage criteria are: average linkage, single linkage, and complete linkage, and many others. We chose to use *average* linkage, where the distance between two clusters $\mathcal{A}$ and $\mathcal{B}$ is taken to be the average of all distances $d(x, y)$ between pairs of objects $x \in \mathcal{A}$ and $y \in \mathcal{B}$, i.e., the mean distance between elements of each cluster:

$$\frac{1}{|\mathcal{A}| \, . \, |\mathcal{B}|} \sum_{x \in \mathcal{A}} \sum_{y \in \mathcal{B}} d(x, y)$$

Figure 14 shows the results of the agglomerative clustering in form of a dendrogram of the layers of our movie multiplex network. This indeed represents the taxonomy of movie genres

based on the similarity of movies in terms of user preferences. This taxonomy is constructed using the layer distance derived from NDC similarity measure, and average linkage as distance measure of clusters.

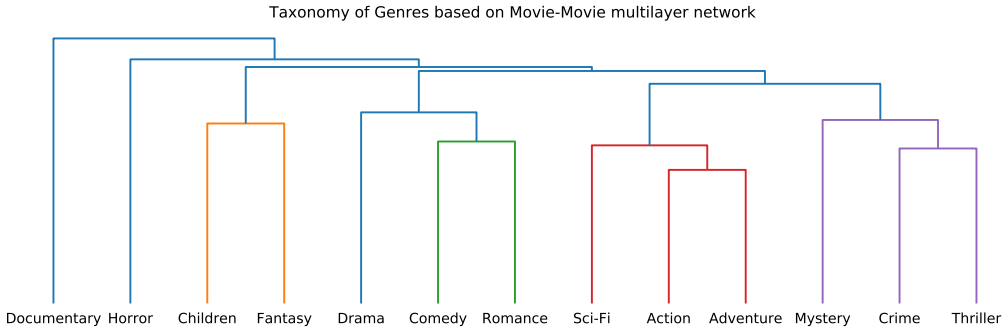



Fig. 14. Taxonomy of Genres based on Movie-Movie multilayer network

This taxonomy reveals a well established partitioning of genres, that looks reasonable and easily interpretable. We can see how similar layers are grouped together to form several clusters of layers. First, we can see that *Thriller*, *Crime*, and *Mystery* genres form together a cluster, indicating how similar those genres are. Second, *Action*, *Adventure*, and *Sci-Fi* genres are also similar to each other such that they are clustered together. The movies *Jurassic Park*, and *Independence Day* are examples of movies that combine those genres. A third notable cluster combines *Drama*, *Comedy*, and *Romance* genres. Although *Children* genre would seem distinct, our approach succeeds to find a similar genre, *Fantasy*, such that they fit together into a cluster. The actual distinct genres that are dissimilar with others, and hence do not fit into a comprehensive clusters are *Horror*, and *Documentary*.

## 8. Conclusion

The classification of movies according to their corresponding genre is a challenging multi-label classification problem. Many existing classification methods are based on textual features, or audio-visual features of movies. In this paper, we have presented an approach to classify movie genres based on user ratings. Our classification approach uses collaborative filtering (CF), a common technique used in recommendation systems. The idea behind classification of movie genres is to use the similarity between movies based on user ratings, to predict the genres of movies.

To facilitate computations, we pre-compute similarity scores among movies, and arrange them into a network where similar movies are connected to each other. We explored several types of neighborhoods (AN, kNN, and WCN), and different types of similarity measures (Pearson correlation $\rho$, Jaccard index $\phi$, and their product $\rho.\phi$). The results of conducted experiments show that our genres classification approach outperforms many existing approaches, by achieving an F1-score of 0.70, and a hit-rate of 94%.

The constructed network of movies is actually a multilayer network, as we consider each genre as a layer comprising the movies that belong to that genre. Hence, we have been able to measure the similarity of layers, using different measures (NJ, EJ, and NDC). We have also applied agglomerative clustering on the layers of this network. As a result, we have obtained a

comprehensible taxonomy of genres which groups together similar genres using the similarity of their movies in terms of user preferences.

## References


1. Alex Blackstock and Matt Spitz, "Classifying movie scripts by genre with a memm using nlp-based features" *Technical report*, Stanford, 2008.
2. Darin Brezeale, "Using Closed Captions and Visual Features to Classify Movies by Genre" In *In Poster session of the Seventh International Workshop on Multimedia Data Mining*, MDM/KDD2006, 2006.
3. Piotr Bródka, Anna Chmiel, Matteo Magnani, and Giancarlo Ragozini, "Quantifying Layer Similarity in Multiplex Networks: A Systematic Study", *Royal Society Open Science*, 5(8):171747, 2018.
4. Andree Elisseeff and Jason Weston, " A kernel method for multi-labelled classification. In Proceedings of the 14th International Conference on Neural Information Processing Systems: Natural and Synthetic", *NIPS01*, pages 681687, Cambridge, MA, USA, 2001. MIT Press.
5. Stephan Fischer, Rainer Lienhart, and Wolfgang Effelsberg, " Automatic recognition of film genres", In *Proceedings of the Third ACM International Conference on Multimedia*, MULTIMEDIA 95, pages 295304, New York, NY, USA, 1995. ACM.
6. Raji Ghawi and Jürgen Pfeffer, "Movie Genres Classification Using Collaborative Filtering", In *Proceedings of the 21st International Conference on Information Integration and Web-Based Applications & Services*, iiWAS2019, page 3544, New York, NY, USA, 2019. Association for Computing Machinery.
7. F. Maxwell Harper and Joseph A. Konstan, "The movielens datasets: History and context", *ACM Trans. Interact. Intell. Syst.*, 5(4):19:119:19, December 2015.
8. Quan Hoang, " Predicting movie genres based on plot summaries", CoRR, abs/1801.04813, 2018.
9. H. Huang, W. Shih, and W. Hsu, "A film classifier based on low-level visual features", In *2007 IEEE 9th Workshop on Multimedia Signal Processing*, pages 465468, Oct 2007.
10. Hui-Yu Huang, Weir-Sheng Shih, and Wen-Hsing Hsu, "A movie classifier based on visual features", In *Computer Analysis of Images and Patterns (Walter G. Kropatsch, Martin Kampel, and Allan Hanbury, editors)* , pages 937944, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.
11. Yin-Fu Huang and Shih-Hao Wang, "Movie genre classification using svm with audio and video features", In *Active Media Technology (Runhe Huang, Ali A. Ghorbani, Gabriella Pasi, Takahira Yamaguchi, Neil Y. Yen, and Beijing Jin, editors)*, pages 110, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.
12. Tae-Gyu Hwang, Chan-Soo Park, Jeong-Hwa Hong, and Sung Kwon Kim, "An algorithm for movie classification and recommendation using genre correlation", *Multimedia Tools Appl.*, 75(20):1284312858, October 2016.
13. Marios Ioannou, George Sakkas, Grigorios Tsoumakas, and Ioannis Vlahavas, "Obtaining bipartitions from score vectors for multi-label classification", In *Proceedings of the 2010 22Nd IEEE International Conference on Tools with Artificial Intelligence*, Volume 01, ICTAI 10, pages 409 416, Washington, DC, USA, 2010. IEEE Computer Society.
14. Eric Makita and Artem Lenskiy, "A multinomial probabilistic model for movie genre predictions", CoRR, abs/1603.07849, 2016.
15. Frank Nielsen, *Hierarchical Clustering*, pages 195211. 02 2016.
16. Chandrasekar Ramachandran, Rahul Malik, Xin Jin, Jing Gao, Klara Nahrstedt, and Jiawei Han, "Videomule: A consensus learning approach to multi-label classification from noisy user-generated videos", In *Proceedings of the 17th ACM International Conference on Multimedia*, MM 09, pages 721724, New York, NY, USA, 2009. ACM.
17. Z. Rasheed, Y. Sheikh, and M. Shah, "On the use of computable features for film classification", *IEEE Trans. Cir. and Sys. for Video Technol.*, 15(1):5264, January 2005.
18. Zeeshan Rasheed and Mubarak Shah, "Movie genre classification by exploiting audio-visual fea-

tures 26 Movie Genre Classification with Collaborative Filtering of previews", *Object recognition supported by user interaction for service robots*, 2:10861089 vol.2, 2002.
19. Jesse Read, Bernhard Pfahringer, and Geoff Holmes, "Multi-label classification using ensembles of pruned sets", In *Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*, ICDM 08, pages 9951000, Washington, DC, USA, 2008. IEEE Computer Society.
20. Jesse Read, Bernhard Pfahringer, Geoff Holmes, and Eibe Frank, "Classifier chains for multi-label classification", In *Proceedings of the 2009th European Conference on Machine Learning and Knowledge Discovery in Databases*, Volume Part II, ECMLPKDD09, pages 254269, Berlin, Heidelberg, 2009. Springer-Verlag.
21. Mack Iii Roach and Jonathan W. D. Mason. Classification of video genre using audio. In Eurospeech, pages 26932696, 2001.
22. Lior Rokach and Oded Maimon, *Clustering Methods*, pages 321352. Springer US, Boston, MA, 2005.
23. Gerhard Sageder, Maia Zaharieva, and Christian Breiteneder, "Group feature selection for audiobased video genre classification", In *Proceedings, Part I, of the 22Nd International Conference on MultiMedia Modeling* - Volume 9516, MMM 2016, pages 2941, Berlin, Heidelberg, 2016. SpringerVerlag.
24. Mohammad S. Sorower, "A literature survey on algorithms for multi-label learning", *Technical report*, 2010.
25. Eleftherios Spyromitros, Grigorios Tsoumakas, and Ioannis Vlahavas, "An empirical study of lazy multilabel classification algorithms", In *Artificial Intelligence: Theories, Models and Applications (John Darzentas, George A. Vouros, Spyros Vosinakis, and Argyris Arnellos, editors)*, pages 401406, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.
26. Lei Tang, Suju Rajan, and Vijay K. Narayanan, "Large scale multi-label classification via metalabeler", In *Proceedings of the 18th International Conference on World Wide Web*, WWW 09, pages 211220, New York, NY, USA, 2009. ACM.
27. Grigorios Tsoumakas, Ioannis Katakis, and Ioannis Vlahavas, "Random k-labelsets for multilabel classification", *IEEE Transactions on Knowledge and Data Engineering*, 23(7):10791089, July 2011.
28. Grigorios Tsoumakas and Ioannis Vlahavas, "Random k-labelsets: An ensemble method for multilabel classification", In *Machine Learning: ECML 2007 (Joost N. Kok, Jacek Koronacki, Raomon Lopez de Mantaras, Stan Matwin, Dunja Mladenic, and Andrzej Skowron, editors)*, pages 406417, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.
29. Zheshen Wang, Ming Zhao, Yang Song, Sanjiv Kumar, and Baoxin Li, "Youtubecat: Learning to categorize wild web videos", In *The Twenty-Third IEEE Conference on Computer Vision and Pattern Recognition*, CVPR 2010, San Francisco, CA, USA, 13-18 June 2010, pages 879886, 2010.
30. Jonatas Wehrmann and Rodrigo C. Barros, "Convolutions through time for multi-label movie genre classification", In *Proceedings of the Symposium on Applied Computing*, SAC 17, pages 114119, New York, NY, USA, 2017. ACM.
31. Jonatas Wehrmann and Rodrigo C. Barros, "Movie genre classification: A multi-label approach based on convolutions through time", *Applied Soft Computing*, 61:973 982, 2017.
32. Ka wing Ho, "Movies genres classification by synopsis". 2011.
33. Yiming Yang, "A study of thresholding strategies for text categorization", In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR 01, pages 137145, New York, NY, USA, 2001. ACM.
34. Ronda J. Zhang and Fred Y. Ye, "Measuring similarity for clarifying layer difference in multiplex ad hoc duplex information networks", *Journal of Informetrics*, 14(1):100987, 2020.
35. Howard Zhou, Tucker Hermans, Asmita V. Karandikar, and James M. Rehg, "Movie genre classification via scene categorization", In *Proceedings of the 18th ACM International Conference on Multimedia*, MM 10, pages 747750, New York, NY, USA, 2010. ACM.