FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

ASSIGNMENT OF BACHELOR’S THESIS

Title:	Recommendation of new movies to obtain
Student:	Tatiana Lekýrová
Supervisor:	doc. Ing. Pavel Kordík, Ph.D.
Study Programme:	Informatics
Study Branch:	Knowledge Engineering
Department:	Department of Applied Mathematics
Validity:	Until the end of summer semester 2019/20

Instructions

Explore the problem of new content acquisition and related work in movie recommendation domain. The task is to identify new movies based on meta-data that would be a good fit for the existing user base. Create an experimental prototype that will predict user engagement (popularity) for a new movie based on meta-data such as a production year, genres, director, actors, or synopses or video popularity which can be gathered from various Internet rankings. Demonstrate the functionality of the prototype on real movie user data provided by Showmax.

References

Will be provided by the supervisor.

Ing. Karel Klouda, Ph.D.
Head of Department

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
Dean

Prague October 10, 2018

FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Bachelor’s thesis

Recommendation of new movies to obtain

Tatiana Lekýrová

Department of Applied Mathematics
Supervisor: doc. Ing. Pavel Kordík, Ph.D.

February 14, 2019

Acknowledgements

In the first place, I would like to thank my supervisor, doc. Ing. Pavel Kordík, Ph.D., for all his advice, patience, and time he spent helping me work on my thesis. Secondly, I would like to thank Ing. Adam Činčura for providing me with initial insights into data obtained from Showmax. Many thanks belongs to my family, friends and boyfriend who supported me during this thesis and my studies.

Declaration

I hereby declare that the presented thesis is my own work and that I have cited all sources of information in accordance with the Guideline for adhering to ethical principles when elaborating an academic final thesis.

I acknowledge that my thesis is subject to the rights and obligations stipulated by the Act No. 121/2000 Coll., the Copyright Act, as amended. In accordance with Article 46(6) of the Act, I hereby grant a nonexclusive authorization (license) to utilize this thesis, including any and all computer programs incorporated therein or attached thereto and all corresponding documentation (hereinafter collectively referred to as the "Work"), to any and all persons that wish to utilize the Work. Such persons are entitled to use the Work for non-profit purposes only, in any way that does not detract from its value. This authorization is not limited in terms of time, location and quantity.

In Prague on February 14, 2019 …………………

Citation of this thesis

Lekýrová, Tatiana. *Recommendation of new movies to obtain.* Bachelor's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2019.

Abstract

This bachelor's thesis explores the problem of movie content acquisition in a recommendation domain. It proposes two solutions to tackle the cold-start setup of predicting popularity of newly obtained content. The first method it proposes is content-based filtering and the second is neural network embeddings. The implementation is supported by the state-of-the-art analysis. The evaluation of the methods shows that content-based filtering has better results in contrast to neural network embeddings.

Keywords cold-start, recommender system, popularity prediction, content-based filtering, embeddings

Abstrakt

Táto bakalárska práca sa zaoberá problémom doporučovania filmov k akvizícii. Navrhuje dve riešenia problému cold-start, ktorý sa objavuje pri predikcii popularity nového obsahu. Prvou navrhovanom metódou je filtrovanie založené na obsahu a druhou metódou sú embeddingy s využitím neurónovej siete. Implementácia je podložená analýzou najmodernejších postupov. Evaluácia metód ukazuje, že filtrovanie založené na obsahu dosahuje lepšie výsledky oproti metóde embeddingov neurónovej siete.

Kľúčové slová cold-start, doporučovacie systémy, predikcia popularity, filtrovanie založené na obsahu, embbedingy

Contents

List of Figures

List of Tables

Introduction

In today's world of fastly evolving technologies, one might get lost in information overload. Responding to this fact may be challenging for firms when hunting for new potential customers. To stand out from the crowd or to introduce new products and content to users likely to be interested in such content, companies make use of recommender systems.

Typical use case of recommender systems is in e-commerce and digital media domains. A broad variety of businesses are searching for solutions to increase their user interaction or enrich their shopping potential. These domains are where recommender systems are promising.

One can see how the aforementioned benefits resulting from an application of recommendation engines had a direct impact on companies' user retention and revenue. The biggest players in the recommendation domain, such as Netflix, YouTube, Spotify or Amazon developed and utilized their own recommendation algorithms that helped them increase their revenue rapidly. Critical part of increasing revenue for a firm is not to only endorse the existing content but also to obtain new content that will reach the existing customer base.

Considering the foreseen potential of recommender systems being used for content acquisition I chose the topic of recommending new movies to obtain to bring a solution for Showmax to help them with content procurement.

The aim of this thesis is to tackle the question of content acquisition. The task is to identify new movies that would be a good fit for the existing user base. To do so, I created an experimental prototype that predicts a popularity of a new item based on movie metadata, such as genres, director and actors. The task is supported by data exploration part that finds popular trends within the existing user base.

In this thesis, starting from chapter one, I explain fundamental terminology of recommender systems and popularity prediction. I provide the readers of this work with a comprehensive study of various prediction domains, where I compare several methods.

I chose two final methods to develop solutions to this thesis using content-based filtering and embeddings with neural network. Critical part of the solutions was preprocessing, which I explain in chapter two. Data exploration and the outlined design of the prediction models are in the same chapter. I trained and tested the models on dataset provided by Showmax. In the third chapter, I evaluate the methods and demonstrate their results.

Chapter 1

State-of-the-art

In this chapter I will review two bodies of related work. Firstly, I will explain what recommender systems are and approaches they can be implemented with, as well as major challenges and predictive modeling and I will finish this part with evaluation metrics for recommender systems. Then I will introduce popularity prediction and state-of-the-art research works done on popularity and user engagement prediction.

1.1 Recommender Systems

Recommender systems are systems designed to suggest user the next activity for entertainment, based on their preferences, history or a variety of other factors.

The design of recommendation engines depends on the area of expertise and the specifications of the obtained data. I will describe this on an example of movie watchers. Watchers give ratings on a scale of 1 (dislike) to 5 (like). When the ratings are saved into a database as records of data, recommender systems can be utilized to find patterns within them. Such data document the quality of interactions between users and items.

Moreover, the system has access to profile attributes specific for users and items such as demographics and product descriptions. Recommender systems vary in the way they analyze these data to develop notions of similarity among items and users, which can later be adopted to identify well-matched pairs. [1]

The input of a recommender system varies according to what data one have and consequently what algorithm one decides to use. Filtering algorithms are introduced in the subsection describing fundamental approaches.

The output of a recommender system can be either a *prediction* or a *recommendation.* A prediction indicates a numerical value, that illustrates the anticipated opinion of user u for item i. The predicted value should be within the same scale as the initial input for user u. A recommendation is described by a list of N items, that the user u is expected to like the most.

1.1.1 Fundamental Approaches

As I mentioned previously, in recommendation domain it is important to choose suitable approach based on value proposition that is being set and most importantly, obtained data. There are several approaches, each appropriate for slightly different dataset one is being provided. My objective of this chapter is to explain the differences in between them. Approaches are as follows.

Collaborative filtering is a method allowing users to rate a set of elements (e.g. movies). When enough information is gathered on the system, recommendations can be made to each user based on information provided by similar users. The most popularly used algorithm for collaborative filtering is the k Nearest Neighbors (kNN). In the version where I have user to user, kNN has the three following steps to generate recommendations for an active user:

1. determine k nearest user neighbors for the active user a,
2. aggregate ratings for the k user neighbors in items not rated by user a, and
3. extract the predictions from step 2 then select the top N recommendations. [2]

Content-based filtering makes recommendations according to past user's choices. Let me describe a sample scenario. If the user purchased some action movies previously, the recommender system will most likely recommend new action movie that the user has not purchased before. Recommendations in content-based filtering are also generated using the content from objects intended for recommendation. For that reason, some content can be analyzed, such as text, images and sound. From this analysis, a similarity can be incorporated between objects as the base for recommendation engine to recommend similar items to one another that a user has bought, visited, viewed and ranked positively. [3]

Demographic based recommender system is based on the proposition that individuals with specific common personal attributes (e.g. age, country, etc.) will also have common preferences. [2]

Utility-based recommender system makes recommendations according to the computation of the utility of each item for the user. Such a system uses features of items as background data, draws out utility functions over items from users to characterize user preferences, and applies the function to calculate the rank of items for a user. The advantage of a utility-based recommender system lies in the fact that this system does not face problems involving new users, new items and sparsity. However, user must build a full utility function and determine each attribute importance, called weight. Determining how to make accurate recommendations with little user effort is a critical issue when designing a utility-based system. [4]

Knowledge based recommender system suggests items based on inferences about a user's needs and preferences. Functional knowledge means how a certain item meets a particular user's need. The user model can be any knowledge structure that supports the mentioned inference (e.g. a query, a case, an adapted similarity metric or a part of an ontology).

Hybrid filtering usually uses a combination of collaborative filtering with demographic filtering or collaborative filtering with content-based filtering to take advantage of benefits of each one of these techniques. Hybrid filtering is for the most part based on techniques such as genetic algorithms, fuzzy genetic, neural networks, Bayesian networks, clustering and latent features. [2]

1.1.2 More Complex Methods

Let me explain various more complex methods how to build a recommender system. One of the algorithms used by the winning team of Netflix prize, BellKor's Pragmatic Chaos, is matrix factorization. Matrix factorization is a realization of latent factor model. Its benefits are good scalability and predictive accuracy. The following is the explanation of matrix factorization models. [5]

Matrix factorization models map both users and items to a joint latent factor space of dimensionality f, such that user-item interactions are modeled as inner products in that space.

Accordingly, each item i is associated with a vector $q_i \in \mathbb{R}^f$, and each user u is associated with a vector $p_u \in \mathbb{R}^f$. Estimate of the rating is then denoted by

$$\hat{r}_{ui} = q_i^T p_u. \tag{1.1}$$

To learn the factor vectors (p_u and q_i), the system minimizes the regularized squared error on the set of known ratings:

$$\min_{p*,q*} \sum_{(u,i)\in\kappa} (r_{ui} - q_i^T p_u)^2 + \lambda(||q_i||^2 + ||p_u||^2) \tag{1.2}$$

Here, κ is the set of the (u, i) pairs for which r_{ui} is known (the training set). The constant λ controls the extent of regularization and is usually determined by cross-validation. However, rating values can vary due to effects associated with either users or items known as biases. Estimate for the Equation 1.1 extended by biases is:

$$\hat{r}_{ui} = \mu + b_i + b_u + q_i^T p_u. \tag{1.3}$$

The overall average rating is denoted by μ; the parameters b_u and b_i indicate the observed deviations of user u and item i, respectively. With added bias and regularized squared error on the set of known rating the final formula is as follows:

$$\min_{p*,q*} \sum_{(u,i)\in\kappa} (r_{ui} - \mu - b_u - b_i - p_u^T q_i)^2 + \lambda(||p_u||^2 + ||q_i||^2 + b_u^2 + b_i^2) \tag{1.4}$$

Singular value decomposition abbreviated as SVD, is a matrix factorization technique that is used to decrease the amount of features of a dataset by reducing space dimensionality. The SVD of a matrix A is the factorization of A into the product of three matrices:

$$A = UDV^T \tag{1.5}$$

where the columns of U and V are orthonormal, which means they are both orthogonal and normalized and the matrix D is diagonal with positive real entries. [6]

1.1.3 Neural Networks

A neural network is a network composed of nodes that have output and input values. When a node is activated it produces output value which is passed along links to other nodes. An input value of a node is computed as the weighted sum of all incoming links. The weight of a link can be modified according to training data. Neural networks can be leveraged to implement recommender systems.

Essential part of neural network is a neuron. Artificial neuron was inspired by biological neurons and it is an elementary unit of a neural network. Activation function is the substantial part of a neuron. The activation function can be e.g. a sigmoid function, as shown in Figure 1.1.

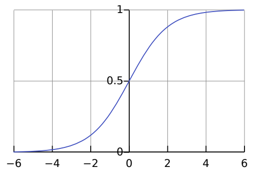


Figure 1.1: Sigmoid activation function [7]

Neural network has the following structure that is visualized in Figure 1.2:

- input layer - feeds input data to the hidden layer,
- hidden layer - encapsulates functions that create predictors, these functions modify the input data and are called neurons,
- output layer - gathers the prediction made in the hidden layer and provides a resulting output. [8]

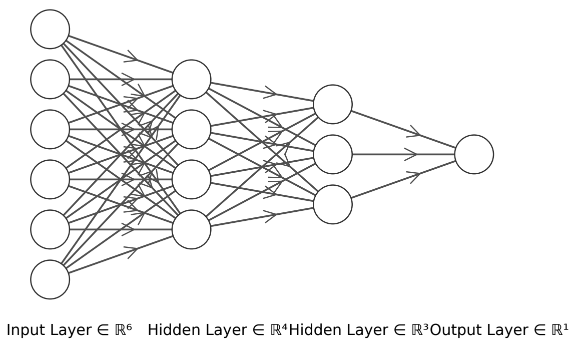


Figure 1.2: Structure of a neural network

Following are the models that can be used when implementing a neural network for recommender systems.

Factorization model is based on an idea that is very similar to SVD where users and items are mapped into a latent space so that they are directly comparable. Typically, one can use two embedding layers to represent users and items, respectively. The target is the interaction, called utility matrix that one passed in. To compute the score for a user-item pair, one can take the dot product of the latent representation for that user and item, and passing it through a sigmoid activation function. By computing the loss for all the user-item pairs with regard to the true interaction, one can optimize the embedding layers by back-propagation.

Sequential model considers the recommendation problem as a sequential prediction problem. Given the past interaction, it is essential to know which item the user is most likely to like in the next step.

When specifying the model, there is a flexibility to change the loss function. There are various loss functions, I will mention one of them, which is Mean

Absolute Error (MAE) which measures the difference of two continuous variables. Models with different loss functions may have significant difference in the performance.

Hyperparameters are crucial when designing a neural network architecture. There is no general policy on how to choose them, settings are dependent on a situation. Overall performance of a neural network can be tuned by testing various hyperparameter settings. I already mentioned some of them, such as activation and loss function. Furthermore it is essential to consider the number of layers, neurons per layer, optimizer and regularization according to the task one is solving.

1.1.4 Embeddings

In previous section I described what neural networks are and two types of models they can be implemented with. One of these models was using embeddings that map users and items into a latent space.

An embedding can be described as mapping from discrete objects to vectors of real numbers. Discrete objects can be for example words. Words do not have a vector representation, therefore they need to be transformed to continuous vectors and then can be used in machine learning algorithms.

As I mentioned neural network embeddings, now I will introduce their aims:

- to locate nearest neighbors in the space of the embedding,
- they are used as an input for a machine learning model,
- to find pattern of concepts and relations among categories and visualize them. [9]

Embeddings can be visulized using various techniques, e.g. t-SNE, which is a t-distributed stochastic neighbor embedding. It is used for dimensionality reduction and it is suitable for visualization of high-dimensional datasets.

1.1.5 Word2vec

Word2vec is an algorithm to generate embeddings. Word2vec uses a neural network with one hidden layer, which is fully connected. The input layer of the neural network has as many neurons as there are words in vocabulary. The size of the hidden layer equals to the dimensionality of the resulting word vectors. Output layer is the same as input layer. There are two models of how to obtain word2vec embeddings that are outlined in diagram in Figure 1.3.

Skip-gram model is based on an idea of having a corpus of words and their context. Skip-gram model takes each word of the corpus and takes its surrounding words within a defined *window*. This is an input for a neural network which then outputs the probability of each word to be nearby (in the *window*) of the center word.

CBOW model which is an abbreviation of Continuous Bag-of-Words. The model predicts the current target word using the source context words, which means that instead of predicting a context word from a word vector, a word is predicted from the sum of all the word vectors in its context.

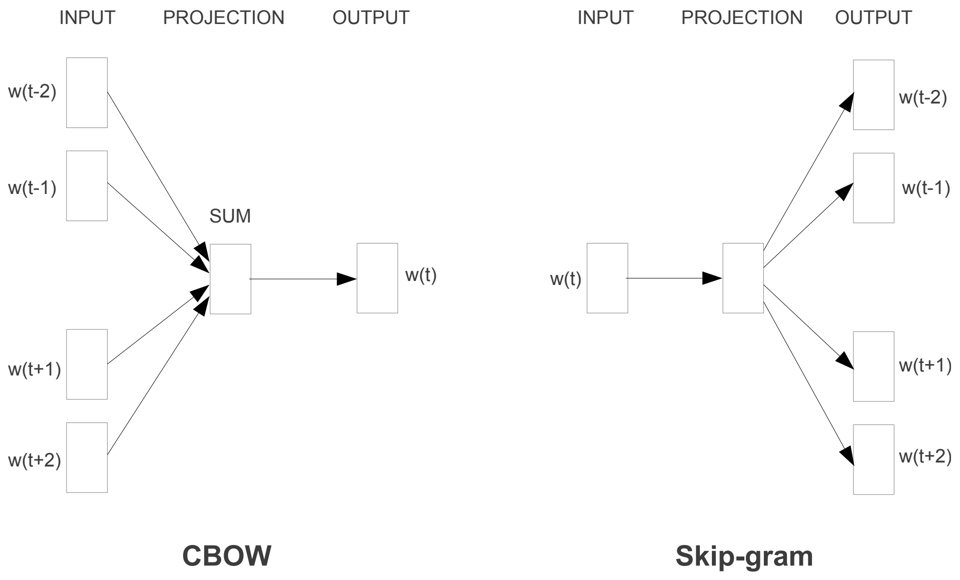


Figure 1.3: Two methods of word2vec [10]

1.1.6 Major Challenges

In the following section I will describe major challenges and problems recommender systems have to deal with and I will present existing solutions to these problems provided by experts in recommender domain.

Cold start is a problem when new user is entering the system or new items are being added to the inventory, or it can be a combination of both new items being recommended to new user. In the case of cold start, system is incapable of drawing any inferences of users or items for which it has insufficient information. There are three cases of cold start:

New community: refers to the start-up of the recommender, when, there are almost no users. The lack of user interaction makes it very hard to provide reliable recommendations.

New item: a new item is added to the system. Although it might have some content information, no interactions are present.

New user: a new user enters the system and has not provided any interaction yet, therefore system is unable to provide personalized recommendations.

Long tail is a problem that occurs when there is a small number of popular items and the rest of items that do not sell that well are located in the heavy tail. The solution to long tail problem in recommendation domain is clustering the whole itemset, specifically the solution can be partitioning the itemset into head and tail and clustering only the tail partition. [11]

Scalability is an issue of designing algorithms to be active enough to compute recommendations for users in a small amount of time. Algorithms must be fast enough to handle large datasets in less than a second. Collaborative filtering algorithms cannot operate without users. When using kNN method, the algorithm is computing similarities between all pairs of users, therefore the running time is quadratic in the number of users. Several approaches have been developed to tackle scalability issue, such as an online learning algorithm that processes each user and incorporate updates simultaneously. Another method to tackle scalability is to use distributed algorithm running on multiple machines in parallel or to model the recommender system as a matrix completion problem and complete the matrix given partially observed entries. [12]

Sparsity in recommendation domain originates from the phenomenon that users in general rate only a limited number of items. In collaborative filtering systems, users or consumers are typically represented by the items they have purchased or rated. In such systems, a consumer-product interaction matrix can be created. Consumer-product matrix is composed of all vectors representing consumers. If user has rated an item, system uses a value typically from 1 to 5 to denote they rating. If user has not rated an item, this element of a matrix (of a certain user having no interaction with a specific item) is set to 0. Both the number of items and the number of users are large and therefore many events may be recorded. For this reason, consumer-product matrix can be extremely sparse. [13]

Synonymy refers to the tendency of a certain amount of the same or very similar items to have different names or entries. Most recommender systems are unable to discover this latent association and thus treat these products differently. For example, the seemingly different items "children movie" and "children film" are actually the same item. [14]

1.1.7 Predictive Modeling

Predictive modeling is a technique that uses statistical methods to predict outcomes. Prediction can be carried out using various models and I will briefly introduce them in the following paragraphs.

Naive Bayes is a probabilistic model used to classification [15]. It is based on the Bayes Theorem which is as follows:

$$P(A \mid B) = \frac{P(B \mid A)\, P(A)}{P(B)} \tag{1.6}$$

where A and B are events, where B has occurred and A is the one happening,

$P(A \mid B)$ is the conditional probability, which is the likelihood of event A occurring given that event B is true,

similarly, $P(B \mid A)$ is also the conditional probability,

$P(A)$ and $P(B)$ are the probabilities of A and B being observed independently of each other.

K Nearest Neighbors model has a purpose of predict the classification of a new sample point among the data points which are separated into several classes. K is a constant defined by user and a sample point is classified by giving it the label which is most frequent among the k training samples closest to the test point.

Support Vector Machines abbreviated as SVM uses an approach called kernel trick to transform the data and based on this transformation it finds a borderline between the possible outputs. This borderline should be optimal. SVM separates the hyperplane which then categorizes new outputs [15].

Random Forest model builds a forest as an ensemble of decision trees, which are tree-like models of decisions and their possible consequences. Usually, random forests models are trained using the bagging method. Random forest adds randomness to the model. It does not search for the most important feature when it splits a node, however, it looks for the best feature from a random subset of features.

Another technique of predictive modeling is neural networks, which I explained in Section 1.1.3. It finds patterns in input data and according to that outputs the result.

1.1.8 Evaluation Metrics

Recommender systems have various methods how to evaluate their models. I will shortly discuss some of the evaluation methods.

Precision recall and accuracy characterize the measure of relevance. There are four classes of results: true positive, true negative, false positive and false negative. A true positive is a result where the system accurately predicts the positive class. A true negative is a result where the system accurately predicts the negative class. The outcome when the system inaccurately predicts the positive class is called a false positive. Similarly, when the system inaccurately predicts the negative class is called a false negative. [16] Precision identifies the percentage of positive identifications that was actually correct. Recall determines the percentage of actual positives that was identified correctly. Accuracy can be described as a ratio of the number of correct predictions to the total number of predictions. [17, 18]

Mean Absolute Error or MAE, which I already mentioned can be also used as a loss function for neural network. It measures the differences between two variables that are continuous, for example two variables of paired observation of *predicted* versus *observed.*

Root Squared Mean Error or RMSE portrays the standard deviation of the residuals, which are a measure of how far from the regression line data points are.

Normalized Discounted Cumulative Gain abbreviated as NDCG is a measure of ranking quality [19]. In information retrieval it measures the gain of a document based on its position in the result list. Each item has a relevance score called gain. Each score is divided by a logarithm of the item position - this is called discounting. These scores are summed up to obtain cumulative gain and get a discounted cumulative gain (DCG). DCGs are then normalized to a number from [0, 1] interval. The NDCG formula is as follows:

$$NDCG_k = \frac{DCG_k}{IDCG_k} \tag{1.7}$$

where IDCG is ideal discounted cumulative gain, which has a formula:

$$IDCG_k = \sum_{i=1}^{|Z|} \frac{2^{z_i} - 1}{\log_2(i+1)} \tag{1.8}$$

where Z is a list of relevant documents that are ordered by their relevance up to position k.

1.2 Popularity Prediction

Now let me move to the prediction part of recommender systems. In this thesis, I aim to predict popularity of new items within the existing consumer base and therefore in the following paragraphs I will introduce the popularity prediction domain.

One way of explaining popularity is that it is a degree of success an item can get in the market. It can be determined by number of downloads, user ratings, number of views, viewing duration, etc. Item popularity is known to have vigorous effects on user behaviour and is leveraged by many existing recommender systems.

There may be some differences between user engagement and item popularity, e.g. if I take as an example popularity on YouTube, the largest video hosting site, where popularity is defined as the willingness to click a video, whereas engagement is user's watch pattern after clicking. [20] However, in this work, I consider user engagement and popularity to be describing the same matter.

User engagement can also be explained as the quality of the user experience that highlights the positive parts of the interaction. In particular, it is the phenomena of users being attracted and motivated to use a web application, specifically when users invest time, attention and emotion into interaction with a web application. As a proxy for user engagement, various metrics are employed, such as view time, page views or number of unique users. [21]

All of the methods from the previous subchapter describing more or less complex approaches to recommendation and the problems occuring when building a recommender engine are most common when developing a standard recommender system, which is a system that returns a list of items for a given user. There exist many research papers describing utilization and comparison of basic or more complex prediction and recommendation methods. However, if one want to implement recommender system that generates list of users for an item, this research area is less documented.

In the prediction domain, several studies have been carried out on popularity prediction, nevertheless, there is no comprehensive study that would gather them all and do popularity prediction on movies yet to be published on a streaming platform and then for each particular movie find a list of users potentially interested in that movie. As part of my thesis, I would like to provide the readers with a review describing methods used across various domains of predictive analysis whose results provide a backbone for building recommender systems suitable for content acquisition. Some of the works tackle also the problem of cold start, which is exactly the case of identifying new content that would be a good fit for the existing user base. I divided the review into sections of various domains which the works tackle.

1.2.1 Movies

To begin, let me summarize the outcomes of Istvan Pilaszy and Domonkos Tikk's work of investigating the advantage of movie metadata compared to movie ratings in regards of predictive power. [22]

Pilaszy and Tikk describe factorization methods used to solve Netflix Prize problem tested against the Netflix Prize dataset consisting of more than 100 million ratings of about 480 000 users on more than 17 000 movies. Firstly, they experimented with matrix factorization methods using no metadata. Secondly, they tested approach called NSVD1 developed by Arkadiusz Paterek [23] which is an SVD method where parameters are learned using gradient descent with regularization and early stopping. Within the later mentioned method, they experimented with various subsets of metadata denoted by the metadata occurrences in specific amount of movies.

Let me explain the interpretation of Paterek's NSVD1 algorithm. Users are characterized by M dimensional binary vectors. These vectors demonstrate which movies are rated by a user and which not and are used to infer user feature vector by a linear transformation, which is denoted by matrix W. Whereas a learning algorithm for matrix factorization finds user feature matrix and movie feature matrix, a learning algorithm for NSVD1 finds movie feature matrix and matrix W. Users can be as well described by vectors derived not only from their ratings but from their metadata too.

In terms of predicting ratings on new movies, Pilaszy and Tikk's work indicates that content-based filtering has a great advantage over collaborative filtering. When a new movie is added to a recommendation engine, there are no ratings for this movie, however metadata are usually available. Their method is a slightly modified Paterek's NSVD1, where movies are represented with sparse vector representation of their metadata. However, the outcome of their work suggests that *"…even 10 ratings of new movie are more valuable than the best solely metadata-based representation."* This is due to the difference between the movie descriptions and the movies themselves.

The research conducted by Haifeng Wang and Haili Zhang [24] introduces *"…a predictive model usable by small and medium-sized enterprises who are in need of a data-based and analytical approach to stock proper movies for local audiences and retain more customers."*

This work takes a look at collaborative filtering and content-based filtering approaches and their known problems and drawbacks. The work designs a machine learning based collaborative filtering recommender system and analyzes how to process selected dataset from Netflix prize [25] and improve prediction performance. The research describes several methods, such as how to handle diverse and big data, how to prune data, balance data, investigate and select features, describes implemented cross-validation, least absolute shrinkage and selection operator, logistics regression model, and a support vector machine (SVM) model with gaussian kernels.

In the evaluation part they classify the test results into four classes: true positive, true negative, false positive and false negative, and then calculate precision, recall and accuracy. With the methods provided, Haifeng Wang and Haili Zhang were able to make improvements over Netflix Cinematch, which is the algorithm behind Netflix recommendations.

Another research paper that is tackling user-preferences prediction is paper written by YingSi Zhao and Bo Shen [26]. In their paper they state: *"The basic idea is to use the information about the relations between movies to estimate users' opinions: if we want to know the opinion of user i about movie a, we could use the opinion of user i about movie b that is a first-order h-neighbor of movie a for the estimate."* H-neighbor is the nearest neighbor and then the connecting relation is called h-connected. In their work, Zhao and Shen are using hyper networks where users are treated as nodes and movies as hyper-edges. Hyper network can be imagined as a hypergraph. Hypergraph generalizes a graph where an edge can join any number of vertices.

Semantics of metadata may also be useful. The reasons why users give particular ratings to movies is correlated with metadata. Preferences of the audience may be guided by the movie genre or by the movie crew, therefore if information on the rating and on the metadata associated to the content is analysed, these information can be used to distinguish users. A user profile can be created such that it represents the level of interest a user had for all particular elements of metadata. Using this technique of content-collaborative hybrid approach combining metadata with ratings lead to improvements in results compared to using just content-collaborative approach itself as can be seen from Marcio Soares's work. [27]

In the paper written by Kevin Dela Ros et al., an explanation of the rating problem can be found. In rating problem, which is also called scoring problem, the goal is to predict scores (or ratings) corresponding to specific items given a query. In the example of Netflix domain, the rating problem lies in predicting the rating of movies by users. Formally, a rating model or scoring model is a function S : $Q{\times}U \rightarrow R$, where $S(Q,U)$ is defined as the score or rating of item U under query Q. [28]

The work carried out by Eugene Seo and his colleagues describes a metadata-based collaborative filtering, abbreviated as MMCF, using k-partite graph for movie recommendation. Their proposed solution utilizes meta-level information, such as genre, director and actor. [29]

K-partite graph is a graph whose vertices can be subdivided into k disjoint sets so that no two vertices within the same set are adjacent. In movie recommendation, the graph contains information such as movie titles, movie genres, Motion Picture Association of America (MPAA) ratings, actors, directors, and user preference ratings. [30]

Eugene Seo et al. implemented their solution in the following steps: from the given dataset by using the implicit relationship extraction build a k-partite graph that maps the relationship between users and metadata, such as genres. Then they use a random walk propagation to get a dense user-metadata implicit relationship. After that they cluster the users to similar groups and predict the ratings.

Random walk is a probabilistic mechanism to move from one node to another. Eugene Seo at al. defined the random steps in a movie k-partite graph as movie selection steps. They provide a formulation of the way of movie selection process based on user's interests in multiple movie metadata. The formulation describes the stochastic process of a user's preference for different metadata based on their initial interest in multiple metadata and the other metadata implicit information.

1.2.2 Video Content

Siqi Wu et al. work on a problem of relations between popularity and engagement in their paper. Specifically, they tackle the question whether *"in a cold-start setup, can engagement be predicted?"* [20] To address this question, they use video content and channel features. They also measure the interaction between view count, watch time, watch percentage and video duration. They visualise the results in an engagement map and propose a metric called relative engagement to estimate video engagement. Engagement map is a 2-dimensional map, where the x-axis shows video duration, and the y-axis shows average watch time. Relative engagement is a duration-calibrated rank of average watch percentage. Using engagement map, they observe that video duration is an important covariate on watch percentage.

Then they set up a task of predicting relative engagement and watch percentage before the video is uploaded. To solve this task, they use linear regression. Linear regression is a linear method used to model the relationship between a dependent variable and one or more independent variables. The outcome of Siqi Wu et al. work is that in a cold-start setting, before any behavioral data is collected, watch percentage and relative engagement are predictable. By the results of their work, they prove that the number of views is one of the key video popularity dynamics.

Video popularity, e.g. popularity of videos published on social media, can be biased by various factors. One of these biases is external context, i.e. if the subject of a video is trending in media, its popularity is anticipated to be high. Also, the network around the publisher of a video like number of their followers has an impact on the content distribution and its future popularity. Furthermore, aspects such as the relevance of the video to the final user and the connection between real world events and the video content are very complex and bring more noise into popularity prediction. [31]

1.2.3 Music Content

In the music content domain, the algorithm described by Mohamed Nasreldin [32] provides an interesting insight into song popularity prediction. In his article, he explains the procedures foregoing the actual prediction. His team obtained dataset of one million songs, which needed classification and cleansing at the beginning. With data exploration, they find out feature importances and trends. Then they test multiple prediction models, such as XGB which is an Extreme Gradient Boosting model provided by `Xgboost` library in Python. Alongside they test other models, such as logistic regression, random forest, kNN, decision trees and SVM. They achieve the highest prediction accuracy by using the XGB model. From their results, they discover that it is simpler to predict an unpopular song rather than a hit based on the data they have.

Another example of popularity prediction in music sphere is algorithm developed by a startup called Hyperlive [33] that claims to quantify musical engagement according to neurobiobehavioural responses of people to music.

1.2.4 News Articles

Yaser Keneshloo's et al. work tackles the problem of news articles prediction [34]. The problem of popularity prediction is considered as regression, where authors engineer different types of classes of features, such as metadata, contextual or content-based, temporal, and social features.

The work of news articles prediction carried out by Yaser Keneshloo at al. sets a goal to predict the number of page views a news article will receive upon the first day from its publication. They deployed the chosen methods in a real setting at The Washington Post, having datasets with and without viral articles. The method they tested include multi linear regression, least absolute shrinkage and selection operator (LASSO) regression, ridge regression and tree regression.

As I mentioned in the video content section, regression is a measure used to model the relationship between a dependent variable and one or more independent variables. Multiple linear regression models model the relationship between two or more descriptive variables and a response variable. To do so, it tries to fit a linear equation to observed data. LASSO regression is used with a goal to obtain the subset of predictors that minimizes the prediction error for a response variable. It forces a constraint on the parameters of the model that induces the regression coefficients for some variables to shrink to zero. [35] Multiple regression data may suffer from multicollinearity, when least squares estimates are unbiased. However, their variances may be far from true value. Then, using ridge regression and adding a degree of bias to the regression estimates, the standard errors may be reduced. Tree regression allow input variable to be a combination of continuous and categorical variables. A decision regression tree is generated when each node in the tree has a test on some value of input variable. The terminal nodes comprise of the predicted output variables.

Multi linear regression has the best performance over both test datasets, therefore they chose this approach for the rest of their experiments. Metadata features result to provide the maximum boost among other set of features.

1.2.5 Social Media

Another work on popularity prediction is the work of Alireza Zohourian et al. [36] about predicting popularity of images and videos on Instagram. This work defines popularity as the amount of interactions on a platform, which can be a social media, retail or streaming platform. These interactions may include shares, likes, comments, clicks, views, view time, etc. Doing the popularity prediction research, authors tested four different regression methods to predict the popularity score, which they define as a ratio of the number of likes to the number of followers. For classification, they also test different methods, such as kNN, random forest, naive Bayes, decision trees. The best results come from local polynomial regression and decision trees as a classification method.

Random forest is an supervised learning algorithm, specifically an ensemble of decision trees. Usually it is trained using the bagging method. The general idea of this method is that a combination of learning models increases the overall result.

The Naive Bayes classifier is based on Bayes' theorem. It provides a way of calculating the posterior probability from class prior probability, predictor prior probability and likelihood, that is the probability of predictor given class.

Research paper written by Benjamin Shulman [37] and his colleagues provides yet another insight into popularity prediction domain. In their work, they studied different problem formulations and kinds of features on popularity prediction. They also encourage content distributors to use temporal features to predict the future success of an item. Temporal feature can be e.g. the average rate of early adoption. The work shows that even a single temporal feature can be a better predictor than multiple non-temporal features combined.

1.2.6 Miscellaneous

There are various advantages if the popularity prediction problem is developed such that future popularity is estimated using only data available at upload time. Based on the results, users can be offered recommendation to modify their content or its description to improve its reception and visibility among users. Work that tackles prediction in a domain with rather less attention, which is recipes popularity prediction is work written by Christoph Trattner and his colleagues. [38]

As indicators of popularity, they chose number of comments and number of ratings. They examine which factors help to explain popularity. The features are related to e.g. a recipe's content, presentation, innovation, etc. They also derive features related to the authoring user and context surrounding the interaction. They do a correlation analysis on possible correlations between features and output variables. They implement an apriori classification and from established variables attempt to validate models with test data. They conducted the experiment with three method, random forest classifier, logistic regression and naive Bayes. For evaluation, they used 5-fold cross validation.

In k-fold cross-validation, the dataset is partitioned into k equal sized subparts. Of the k subparts, one is employed as validation data for testing, and the remaining $k-1$ subparts are used for training. The process is then repeated k times. Each of the k subparts is used exactly once for validation. After that, k results can be used to calculate an average to produce a single estimation. When combining all features, Trattner and his team were able to accurately predict popularity in more than 60% of cases in one and 80% of cases in the second dataset they used.

Chapter 2

Solution Design

This chapter starts with information about obtained data, which I divided into data description, data preprocessing and data exploration. Then I explain two solutions I decided to use to tackle the aims that were set at the beginning of this thesis.

2.1 Data Description

In this section I provide an insight into the dataset obtained from Showmax, where I describe a subset of attributes I decided to use. This subset was then used in preprocessing part and the preprocessed dataset was used to implement a popularity prediction model.

The dataset from Showmax consists of metadata and interactions. Data describes information about an asset, which can be either a movie or a TV series episode. Important information for my model that an asset has is `asset_id`, which is its unique identifier and `asset_name` that determines asset title.

Originally, asset had much more attributes which I decided not to use because they were not relevant for my model. I removed some attributes in the preprocessing part. These attributes include `asset_type` which determines whether asset is a movie or an episode of a TV show, because in this work I was focusing on metadata popularity in general and not the fact on whether audience is interested more in movies or in TV shows. Furthermore, there were attributes such as time when asset was added to the platform called `created_at`, or when it was updated called `updated_at`. Asset also had `published_at` attribute. This could be used for time estimation on how long an asset is on the platform, to not favor assets that have more interactions because they are longer on the platform. However, I decided to work with only a subset of interactions, as I describe later in Data Preprocessing part, and the distribution of asset time on platform was almost equal among all the assets in this subset. Therefore it would not help to calculate how long an asset is on the platform for further implementation. From the original attribute table I also dropped more features, which can be seen in Table 2.1. This table has two subtables, the one on the left shows original attributes and the one on the right describes attributes I used in the implementation.

Table 2.1: Asset attributes transformation

Original asset attributes

Attribute name	Data type
`asset_id`	string
`asset_name`	string
`asset_type`	integer
`provider_id`	string
`created_at`	datetime
`updated_at`	datetime
`published_at`	datetime
`parent_asset_id`	string
`slug`	string
`number`	integer
`disabled`	boolean
`published`	boolean
`network_id`	string
`vod_model`	integer
`weight`	integer
`deleted`	boolean
`download_policy_id`	string
`show_in_search`	boolean
`show_children`	boolean
`next_episode_start`	string
`auto_published_at`	datetime
`promoted`	boolean

Attributes used in implementation

Attribute name	Data type
`asset_id`	string
`asset_name`	string

Showmax dataset consists of various metadata supporting an asset, such as `genre` that characterizes asset genre, `tag` which describes an asset in further detail, `year` in which an asset was released. Moreover, there is metadata about person that outlines people participating in the movie such as actors and directors, and can be found in dataset with a respective `person_id`. Person has an `imdb_id` connected to them, linking an IMDb url. However the dataset contains only 112 records with IMDb link, therefore I decided not to use the popularity from IMDb rankings. Similarly, there is `imdb_id` with asset IMDb url, however this attribute does not contain any values therefore I did not use it to gather any rankings either. Original dataset contained more metadata supporting an asset, such as MPAA ratings, which I did not to use, because

I decided to focus only on selected metadata. Table 2.2 shows asset category information, which was joined with asset attributes using `asset_id` via a third table. I did this in the preprocessing part. I also kept asset descriptions in the dataset to be able to generate neural network and word2vec embeddings.

Table 2.2: Table describing category

Asset Category

Attribute name	Data type
`category_id`	string
`category_name`	string
`category_type`	string

Apart from metadata, dataset contains interactions between a user and an asset, interactions are called viewership. Viewership contains the duration of each interaction and start time, which is time when user started to watch an asset, which I consider as not relevant for the implementation of prediction model. The finalized data I used can be found in Table 2.3. Because of the features the dataset has, I used duration of interaction, which I call `net_time`, as a metric of popularity. If the duration is short (e.g. a few seconds only), I assume that a user did not like a movie and a movie that has a lot of short interactions can be described as unpopular one, on the other hand if an interaction is long (e.g. thousands of seconds), I consider asset as popular.

Table 2.3: Table describing interactions between a user and an asset

Viewership Features

Attribute name	Data type
`user_id`	string
`asset_id`	string
`net_time`	int

2.2 Data Preprocessing

I did the preprocessing part using Microsoft Excel and Python. I removed unnecessary attributes in Microsoft Excel and then preprocessed the data in Python. I truncated the number of interactions to 100,000 to make further processing faster. I did data preprocessing in the following phases, where I:

- identified incomplete records and removed columns that were not useful for the predictor,

- merged tables of asset with its metadata,
- concatenated metadata to a usable form for the model,
- merged tables of asset with interactions,
- removed records with anonymous interactions,
- de-duplicated the data,
- removed interactions with negative `net_time`,
- validated the data.

To simulate the cold-start setup I deleted interactions from a subset of assets. I split the interactions into two halves according to assets, having new assets with *future* interactions in one half and assets already existing with *historical* interactions in the other. New assets are unique for *future* set only and *historical* set does not contain them. I used the subset with new assets to test the model and the subset with old asset to train the model. However, as I needed to keep the already existing user base, I had to find and delete interactions from *future* set with users that did not have interactions in the *historical* set.

Table 2.4 shows finalized data example after preprocessing. `Asset_id` links in between the asset and viewership table. `Asset_id` is a string, however in the example I use integer value to portray it. When I finished data preprocessing, I moved on to data exploration by looking into trends in the dataset.

Table 2.4: Example of an asset merged with its metadata

`asset_id`	`asset_name`	`genre`
23456	Friends	Comedy|Romance
12399	Disturbia	Thriller|Crime|Mystery

2.3 Data Exploration

In the data exploration phase, I looked into metadata features and trends among them. I started off with looking into genres. I filtered out all the genres and number of their occurences, which resulted in 22 diverse genres. Figure 2.1 shows that *drama* is the most popular genre. The figure was generated using 5,269 interactions.

Similarly, Figure A.1 in the Appendix shows obtained tags. The figure was generated using a dataset of 20,000 interactions and 472 tags were extracted, however figure only contains 30 of them to be more readable. The figure shows that *light-hearted*, *captivating* and *21st-century* are among the most popular tags out of all 472 extracted tags.

Figure A.2 in the Appendix shows acquired people. 20,000 interactions were used to generate the figure and more than 4,500 people were extracted participating in all assets. Similarly to tags, figure only captures 30 of the people to be more readable. As people had only `person_id` assigned, I renamed them in the figure. *Person1* is the most popular. The generated figures identify popular trends and indicate what genres, tags and persons would be a good fit for the existing user base.

2.4 Proposed Solutions

2.4.1 Content-Based Model

Because of the cold-start setup I had, where I was predicting popularity of new assets without interactions, I decided to implement a content-based filtering recommendation model.

Content-based recommender system is based on an assumption that there is a content available for different assets. Content is represented by features that characterize the content of these movies, such as what genre does a movie have.

Content-based approach uses tf-idf, which stands for term frequency–inverse document frequency. Tf-idf statistic reflects how important a term is to a document in a collection of documents, or in my case, how important an asset is in a collection of assets.

To predict the rating of user u for asset i, I:

- calculated the weighted average of `net_time` of every other asset that user u has rated,
- restrained the weighted average to assets that have a positive cosine similarity with asset i,
- calcucated the weight for asset a that conforms to the cosine similarity between a and i,
- if there were no other assets with positive cosine similarity to use in the prediction, I used the mean `net_time` of the target user in train set as the prediction.

As a similarity metric I used cosine similarity. Cosine similarity calculates the cosine angle between two vectors (or two assets). It is a measurement of orientation and not magnitude, therefore it can be seen as a comparison on a normalized space. Cosine similarity formula is as follows:

$$\cos\theta = \frac{A \cdot B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}} \tag{2.1}$$

In the implementation, asset with metadata and viewership were represented as a `pandas dataframe` object in Python. In further detail, the prediction model is executed in the following steps:

1. load the preprocessed asset dataset and the preprocessed viewership datasets with *historical* and *future* interactions,
2. tokenize assets by extracting one token per string that is extracted from metadata of each asset (exracted from genre, year or person),
3. featurize movies by calculating a matrix of shape [1, number of features] where each entry will contain a tf-idf value of the term:

$$\text{tf-idf}(i, d) := \frac{\text{tf(i, d)}}{\max_k \text{tf}(k, d) * \log_{10}(\frac{N}{\text{df}(i)})} \tag{2.2}$$

   where i is a term and d is a asset,

   $\text{tf}(i, d)$ is the frequency of i in d,

   $\max_k \text{tf}(k, d)$ is the maximum frequency of any i in d,

   N determines the number of assets,

   $\text{df}(i)$ is the number of unique assets accommodating term i,

4. predict the time for which user u will watch an asset i:
   - calculate weight for asset a as cosine similarity between asset a and m,
   - calculate the weighted average for every other asset user u watched,
   - restrict this weighted average to assets that have a positive cosine similarity with asset i,
   - if no other assets with positive cosine similarity exist, use the mean watch duration of the target user from train set as prediction.

2.4.2 Neural Network Embeddings

As my second proposed solution, I decided to observe the performance of neural network embeddings on the dataset I have. In section 1.1.2 I described matrix factorization, which provides a base to the proposed model.

Matrix factorization is capable of learning a low-dimensional representation, also called embedding, of user and asset. Key essential part of the algorithm is to obtain an embedding for user, asset and asset description. Matrix factorization technique I used is called non-negative because it requires the embeddings to be non-negative.

I created a neural network that produces watch time estimations. User, asset and asset description embeddings are concatenated and used as features for proposed neural network. This way the neural network can learn more complicated relationships that are non-linear. User, asset and asset description embeddings can now have different dimensions. This is useful especially when one dimension varies from another.

Neural network has the following architecture. I created three embedding layers for asset, asset descriptions and user. Embeddings were created using latent factors. Firstly, a tensor was created for each embedding and then reshaped according to latent factors. Tensor is a generalization of vector to dimensions, which can be potentially higher.

After creating embeddings, all three layers were concatenated together. I added several hidden layers to my network, experimenting with various hyperparameters, which are described in Experimental part 3.2.2 of this thesis. After that I trained the model with *historical* interactions and did predictions on *future* interactions.

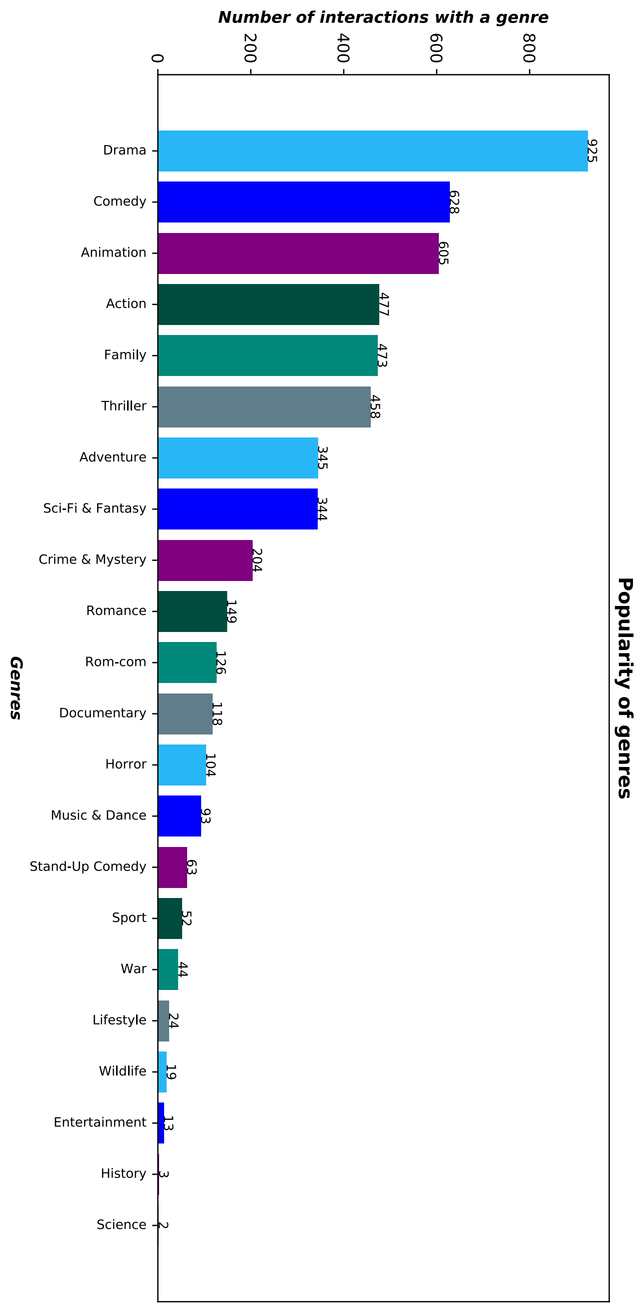


Figure 2.1: Popularity of extracted genres

Chapter **3**

Experimental Part

3.1 Environment

Python is an interpreted programming language that presents automatic memory management. Python code is executed without previous compiling. The interpreter executes the program directly, with each statement being translated into machine code.

Python is very popular in data science and due to its popularity has an extensive collection of frameworks and libraries to be used for machine learning algorithms. Thus Python is prioritized among developers to be used for machine learning tasks and that was also the reason why I chose this language to implement my designed solution.

In Python I mainly worked with the following libraries:

pandas is an open source, data analysis library. It provides high performance and easy-to-use data structures. As I mentioned, I used `Pandas dataframe` object.

NumPy is the package for scientific computing. It mainly contains a powerful N-dimensional array object, sophisticated functions, useful linear algebra, and random number capabilities. Besides the mentioned, `NumPy` can be used as a multi-dimensional container of generic data as well.

scikit-learn is a machine-learning library providing efficient tools for data mining and data analysis. I calculated cosine similarity with `sklearn.metrics`.

gensim library analyzes semantic structure of documents and provides scalable statistical semantics. I implemented a `word2vec` model from this library.

Keras is an open source deep learning library. It is capable to be running on top of Tensorflow. I used it to build a neural network.

3.2 Results

For evaluation of the implemented recommendation model I used NDCG measure of ranking quality, which I explained previously in 1.1.8 I implemented two models to estimate user engagement. With these two models I decided to observe the performance of various metadata. First proposed solution is based on content-based filtering and I observe how well does genre, year and person metadata perform within this method. For my second solution, I looked at asset descriptions and what is their performance on creating neural network embeddings.

3.2.1 Content-based Model

As described before in the Data Preprocessing section of the second chapter, I divided the dataset into two subsets with new and already existing assets. From these two subsets I randomly selected a sample to train and test the model from subset with already existing assets and new assets, respectively. I selected the sample in the 0.85:0.15 ratio, from *historical* set and*future* set respectively, as this proved to have the best results.

The obtained results from the content-based algorithm trained and tested with multiple metadata, such as genre, person and year can be seen in the Table 3.1 Each method was tested several times and the results in the table shows an average of these experiments. The table shows that genre from these three compared metadata is the best to predict duration the viewer will spend watching an asset, whereas year is the worst. The reason for this is influenced by *future* interactions and *historical* interactions datasets. Both these sets, after preprocessing that started with 100,000 interactions, ended with very few data, with about 900 interactions in *historical* interactions set and only 45 interactions in *future* set for year as a metadata feature. In the model, a train and test subset is chosen from these two, and because the number of interactions is very small, the prediction is not very accurate.

Table 3.1: Table comparing NDCG measures of content-based method

Content-based Method

Metadata Used	NDCG
Genre	0.6752
Person	0.6097
Year	0.3729

Figure 3.1 shows predicted watch durations vs. actual watch durations when using genre. Dark blue line shows ideal result, if all predicted durations

would match the actual ones. Light blue dots show deviation from this ideal result.

Similarly, Figure A.3 in the Appendix shows predicted watch durations versus observed watch durations when using person as metadata and Figure A.4 in the Appendix shows predicted versus observed when using year as metadata.

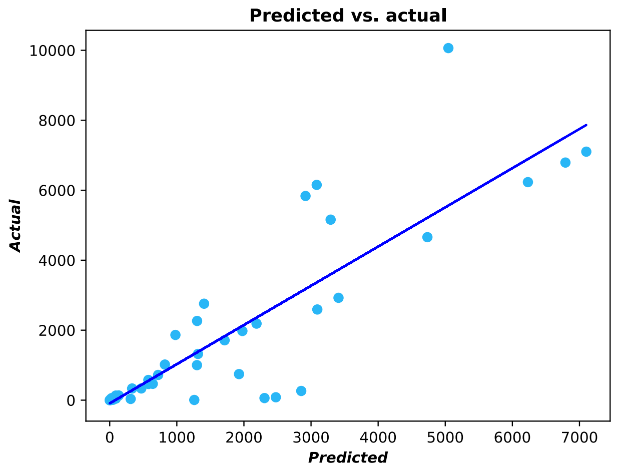
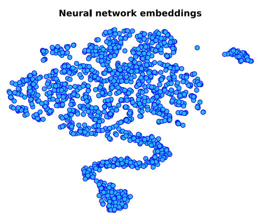


Figure 3.1: Predicted versus actual watch durations using genre

3.2.2 Embedding with Neural Network

I visualized obtained embeddings using t-SNE. The embeddings generated can be seen in Figure 3.2 and Figure 3.3 The Figure 3.2 shows neural network (NN) embeddings generated using. Just for comparison, I also used word2vec and its skip-gram model to visualize how were words from asset descriptions clustered, this can be seen in Figure 3.3.



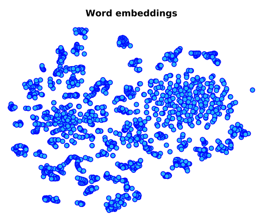


Figure 3.2: NN embeddings

Figure 3.3: Word2vec embeddings

In my neural network implementation, I experimented with various hyper-parameters. I tested variable number of layers and neurons per layer. Results of this testing can be seen in Table 3.2.

Table 3.2: Table of NDCG results of neural network embeddings

Neural Network Embeddings

Number of layers	NDCG
19	0.0174
15	0.0152
14	0.0157

I constructed pyramid with 256-128-64-32-16-8-4-2-1 neurons on each layer and also tested smaller versions of this pyramid, which resulted in smaller number of layers. The output layer always had one neuron. As an activation function I decided to use ReLU (Rectified Linear Unit), which had a slightly better performance than sigmoid. As a loss function I used mean absolute error. There were various optimizer algorithms I could choose and Adam yielded the best results. I tested my neural network with 20% dropout of all the neurons on each layer. I started training my neural network with 100 epochs, eventually getting to 250 and 1,000 epochs. Figure 3.4 portrays how neural network model trained and improved over epochs.

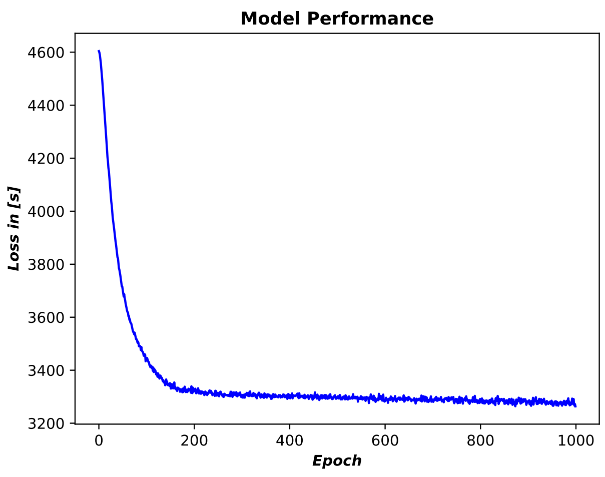


Figure 3.4: Performance of neural network

Conclusion

This thesis had a purpose of investigating the problem of new content acquisition and relevant work in movie recommendation domain. In the first chapter, I introduced the reader with the fundamental terminology of recommender systems that were further outlined when comparing the methods of the popularity prediction review. I tackled the popularity prediction problem from a more broader perspective to not only investigate the movie domain but also look at various spheres at which prediction domain is employed, such as video and music content, news articles, social media, recipes and last but not least, movie popularity prediction.

In the second chapter I proposed a solution for the task of new movies identification that would be a good fit for the existing user base. Critical part was not only the implementation of the prediction model, but also data preprocessing, which helped me to cleanse the data before the recommendation. The solution I proposed is called content-based filtering which is specifically useful in a cold-start setup of prediction popularity of new movies without any interactions.

Showmax metadata features provided IMDb link that links assets or people participating in an asset to an IMDb page, however, very few records having this link were available and therefore were not usable to gather popularity rankings.

In the experimental part of this thesis, I described how the prototype was tested and what results were carried out. To evaluate the model I used NDCG metric and content-based filtering proved to be the method having better results on the contrary to neural network embeddings. This can be caused due to high bias in the neural network which can be countered by hyperparameter setting.

To summarize, all of the aims of this thesis were accomplished, except for gatherings of the popularity Internet rankings. This could be done in the future, if more IMDb links were be available, I could use them to gather rankings and test the model on them.

Bibliography

1. SAMMUT, Claude; WEBB, Geoffrey I. *Encyclopedia of machine learning and data mining.* 2nd ed. Springer Publishing Company, Incorporated, 2017. ISBN 978-1-4899-7687-1.
2. THORAT, Poonam B; GOUDAR, RM; BARVE, Sunita. Survey on collaborative filtering, content-based filtering and hybrid recommendation system. *International Journal of Computer Applications.* 2015, vol. 110, no. 4. ISSN 0975–8887. Available also from: `https://pdfs.semanticscholar.org/51f4/78339cc86f20fff222baf17be5bab7f29bc3.pdf`.
3. BOBADILLA, Jesús; ORTEGA, Fernando; HERNANDO, Antonio; GUTIÉRREZ, Abraham. Recommender systems survey. *Knowledge-based systems.* 2013, vol. 46, pp. 109–132. ISSN 0950-7051. Available also from: `https://www.sciencedirect.com/science/article/pii/S0950705113001044`.
4. HUANG, Shiu-Li. Designing utility-based recommender systems for e-commerce: Evaluation of preference-elicitation methods. *Electronic Commerce Research and Applications.* 2011, vol. 10, no. 4, pp. 398–407. ISSN 1567-4223. Available also from: `https://www.sciencedirect.com/science/article/pii/S156742231000089X`.
5. KOREN, Yehuda; BELL, Robert; VOLINSKY, Chris. Matrix factorization techniques for recommender systems. *Computer.* 2009, no. 8, pp. 30–37. ISSN 1558-0814. Available also from: `https://ieeexplore.ieee.org/document/5197422`.
6. NGUYEN, Anh. *Singular Value Decomposition in Recommender Systems.* 2016. Available also from: `https://repository.tcu.edu/handle/116099117/11320`.
7. COMMONS, Wikimedia. *File:Logistic-curve.svg — Wikimedia Commons, the free media repository.* [online], 2017. Available also from: `https://commons.wikimedia.org/w/index.php?title=File:Logistic-curve.svg&oldid=266067083`. Accessed: 2018-11-02.

8. JAIN, Anil K; MAO, Jianchang; MOHIUDDIN, K Moidin. Artificial neural networks: A tutorial. *Computer*. 1996, vol. 29, no. 3, pp. 31–44. ISSN 0018-9162. Available also from: `https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=485891`.
9. KOEHRSEN, Will. *Neural Network Embeddings Explained*. [online]. Available also from: `https://towardsdatascience.com/neural-network-embeddings-explained-4d028e6f0526` Accessed: 2019-11-02.
10. MIKOLOV, Tomas; CHEN, Kai; CORRADO, Greg; DEAN, Jeffrey. Efficient Estimation of Word Representations in Vector Space. *CoRR*. 2013, vol. abs/1301.3781. Available from arXiv: `1301.3781`.
11. PARK, Yoon-Joo; TUZHILIN, Alexander. The long tail of recommender systems and how to leverage it. In: *Proceedings of the 2008 ACM conference on Recommender systems*. 2008, pp. 11–18. Available also from: `http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.421.1833&rep=rep1&type=pdf`.
12. XIN, Yu et al. *Challenges in recommender systems: scalability, privacy, and structured recommendations*. 2015. Available also from: `http://hdl.handle.net/1721.1/99785`. PhD thesis. Massachusetts Institute of Technology.
13. CHEN, Yibo; WU, Chanle; XIE, Ming; GUO, Xiaojun. Solving the sparsity problem in recommender systems using association retrieval. *Journal of computers*. 2011, vol. 6, no. 9, pp. 1896–1902. ISSN 1796-203X. Available also from: `http://www.jcomputers.us/vol6/jcp0609-17.pdf`.
14. ISINKAYE, FO; FOLAJIMI, YO; OJOKOH, BA. Recommendation systems: Principles, methods and evaluation. *Egyptian Informatics Journal*. 2015, vol. 16, no. 3, pp. 261–273. ISSN 1110-8665. Available also from: `https://www.sciencedirect.com/science/article/pii/S1110866515000341`.
15. RUSSELL, Stuart; NORVIG, Peter. *Artificial Intelligence: A Modern Approach*. 3rd. Upper Saddle River, NJ, USA: Prentice Hall Press, 2009. ISBN 0136042597, 9780136042594. Available also from: `https://faculty.psau.edu.sa/filedownload/doc-7-pdf-a154ffbcec538a4161a406abf62f5b76-original.pdf`.
16. GOOGLE, LLC. *Classification: True vs. False and Positive vs. Negative*. [online]. Available also from: `https://developers.google.com/machine-learning/crash-course/classification/true-false-positive-negative` Accessed: 2018-14-12.
17. GOOGLE, LLC. *Classification: Precision and Recall*. [online]. Available also from: `https://developers.google.com/machine-learning/crash-course/classification/precision-and-recall` Accessed: 2018-14-12.

18. GOOGLE, LLC. *Classification: Accuracy.* [online]. Available also from: `https://developers.google.com/machine-learning/crash-course/classification/accuracy` Accessed: 2018-14-12.

19. WANG, Yining; WANG, Liwei; LI, Yuanzhi; HE, Di; CHEN, Wei; LIU, Tie-Yan. A theoretical analysis of NDCG ranking measures. In: *Proceedings of the 26th Annual Conference on Learning Theory (COLT 2013).* 2013, vol. 8. Available also from: `http://proceedings.mlr.press/v30/Wang13.pdf`.

20. WU, Siqi; RIZOIU, Marian-Andrei; XIE, Lexing. Beyond Views: Measuring and Predicting Engagement on YouTube Videos. *CoRR*. 2017, vol. abs/1709.02541. Available from arXiv: `1709.02541`.

21. LEHMANN, Janette; LALMAS, Mounia; YOM-TOV, Elad; DUPRET, Georges. Models of user engagement. In: *International Conference on User Modeling, Adaptation, and Personalization.* 2012, pp. 164–175. Available also from: `https://www.researchgate.net/publication/233852009_Model_of_User_Engagement`.

22. PILÁSZY, István; TIKK, Domonkos. Recommending new movies: even a few ratings are more valuable than metadata. In: *Proceedings of the 2009 ACM Conference on Recommender Systems, RecSys 2009, New York, NY, USA, October 23-25, 2009.* 2009, pp. 93–100. ISBN 978-1-60558-435-5. Available from DOI: `10.1145/1639714.1639731`.

23. PATEREK, Arkadiusz. Improving regularized singular value decomposition for collaborative filtering. In: 2007. Available also from: `https://www.mimuw.edu.pl/~paterek/ap_kdd.pdf`.

24. WANG, Haifeng; ZHANG, Haili. Movie genre preference prediction using machine learning for customer-based information. 2018. ISBN 978-1-5386-4649-6. Available from DOI: `10.1109/CCWC.2018.8301647`.

25. NETFLIX, Inc. *Netflix Prize data | Kaggle.* [online]. Available also from: `https://www.kaggle.com/netflix-inc/netflix-prize-data` Accessed: 2018-16-12.

26. ZHAO, YingSi; SHEN, Bo. Empirical study of user preferences based on rating data of movies. *PloS one*. 2016, vol. 11, no. 1, pp. e0146541. ISSN 1932-6203. Available also from: `https://journals.plos.org/plosone/article/file?id=10.1371/journal.pone.0146541&type=printable`.

27. SOARES, Márcio; VIANA, Paula. *The Semantics of Movie Metadata: Enhancing User Profiling for Hybrid Recommendation.* 2017. ISBN 978-3-319-56534-7. Available from DOI: `10.1007/978-3-319-56535-4_33`.

28. ROSA, Kevin Dela; METSIS, Vangelis; ATHITSOS, Vassilis. Boosted ranking models: a unifying framework for ranking predictions. *Knowledge and information systems*. 2012, vol. 30, no. 3, pp. 543–568. ISSN 0219-1377. Available also from: `http://www.cs.cmu.edu/~kdelaros/kais2011.pdf`.
29. SEO, Eugene; KANG, Dongyeop; CHOI, Ho-Jin. Metadata-Based Collaborative Filtering Using K-Partite Graph for Movie Recommendation. In: *The Fourth International Conference on Emerging Databases- Technologies, Applications, and Theory (EDB)*. 2012, pp. 54–61. Available also from: `http://koasas.kaist.ac.kr/bitstream/10203/172288/1/2012_EDB_%EC%84%9C%EC%9C%A0%EC%A7%84_Metadata%20based%20collaborative%20filtering.pdf`.
30. CHENG, Haibin; TAN, Pang-Ning; STICKLEN, Jon; PUNCH, William F. Recommendation via query centered random walk on k-partite graph. In: *Data Mining, 2007. ICDM 2007. Seventh IEEE International Conference on*. 2007, pp. 457–462. ISSN 1550-4786. Available also from: `http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.151.4005&rep=rep1&type=pdf`.
31. TRZCIŃSKI, T.; ROKITA, P. Predicting Popularity of Online Videos Using Support Vector Regression. *IEEE Transactions on Multimedia*. 2017, vol. 19, no. 11, pp. 2561–2570. ISSN 1520-9210. Available from DOI: `10.1109/TMM.2017.2695439`.
32. NASRELDIN, Mohamed. *Song Popularity Predictor*. 2018. Available also from: `https://towardsdatascience.com/song-popularity-predictor-1ef69735e380` Accessed: 2018-14-12.
33. HYPERLIVE. *HyperLive*. [online], 2018. Available also from: `https://hyperlive.fm/` Accessed: 2018-14-12.
34. KENESHLOO, Yaser; AL., et. *Predicting the Popularity of News Articles*. 2016. ISBN 978-1-61197-434-8. Available also from: `http://people.cs.vt.edu/naren/papers/sdm2016.pdf`.
35. ROSE, Jen; DIERKER, Lisa. *What is Lasso Regression?* Available also from: `https://www.coursera.org/lecture/machine-learning-data-analysis/what-is-lasso-regression-0KIy7`. Accessed: 2018-14-12.
36. ZOHOURIAN, Alireza; AL., et. *Popularity prediction of images and videos on Instagram*. 2018. ISBN 978-1-5386-5364-7. Available from DOI: `10.1109/ICWR.2018.8387246`.
37. SHULMAN, Benjamin; SHARMA, Amit; COSLEY, Dan. *ICWSM*. Predictability of Popularity: Gaps between Prediction and Understanding. 2016. Available also from: `https://arxiv.org/abs/1603.09436`.

38. TRATTNER, Christoph; MOESSLANG, Dominik; ELSWEILER, David. On the predictability of the popularity of online recipes. *EPJ Data Science*. 2018, vol. 7, no. 1, pp. 20. ISSN 2193-1127. Available from DOI: `10.1140/epjds/s13688-018-0149-5`.

Appendix A

Appendix

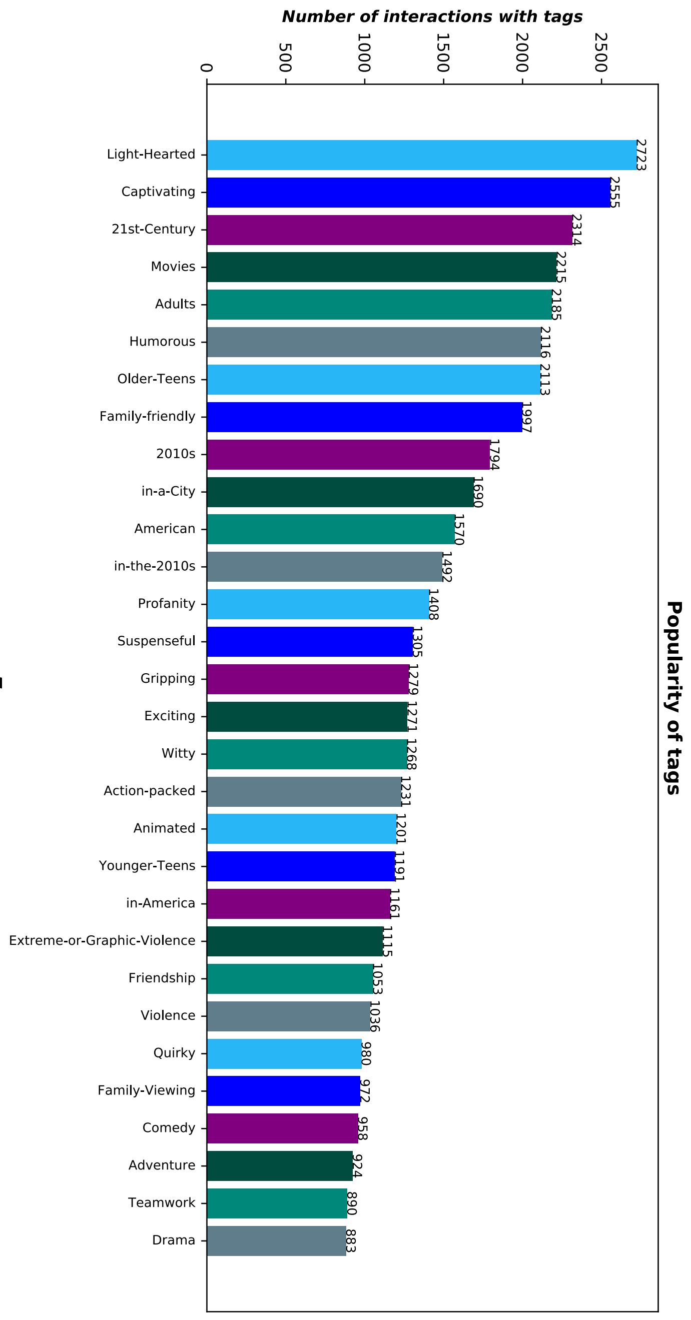


Figure A.1: Popularity of extracted tags

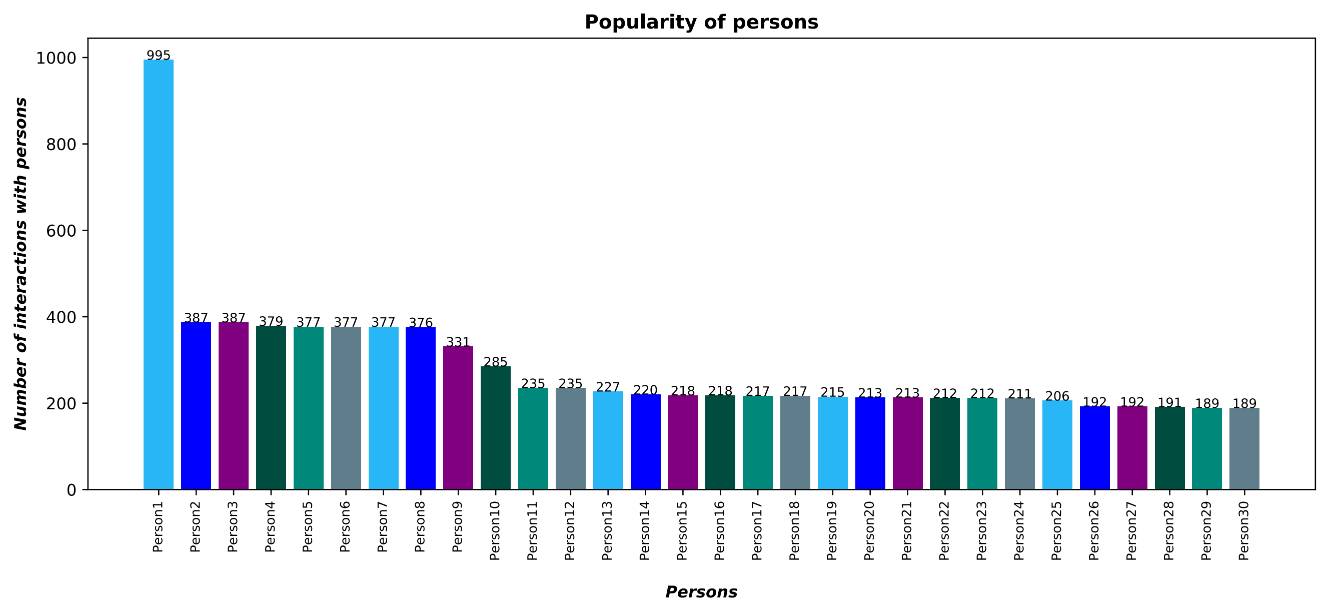


Figure A.2: Popularity of extracted people participating in movies or TV series

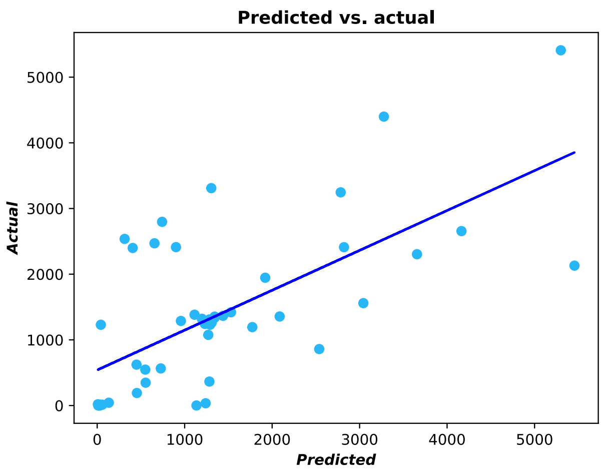


Figure A.3: Predicted versus actual watch durations using person

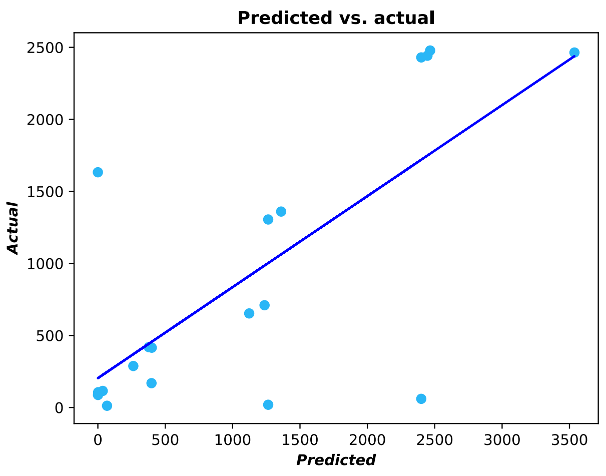


Figure A.4: Predicted versus actual watch durations using year

Appendix B

Acronyms

CBOW	Continuous Bag-of-Words
IMDb	Internet Movie Database
kNN	k Nearest Neighbors
LASSO	Least Absolute Shrinkage and Selection Operator
MAE	Mean Absolute Error
MMCF	Metadata-Based Collaborative Filtering
MPAA	Motion Picture Association of America
NDCG	Normalized Discounted Cumulative Gain
NN	Neural Network
RMSE	Root Mean Square Error
SVD	Singular Value Decomposition
SVM	Support Vector Machines
t-SNE	T-distributed Stochastic Neighbor Embedding
tf-idf	Term Frequency-Inverse Document Frequency
URL	Uniform Resource Locator
XGB	Extreme Gradient Boosting

Appendix C

Contents of enclosed CD

- `readme.txt` .... the file with CD contents description
- `thesis.pdf` .... the thesis text in PDF format
- `src` .... the directory of source codes
  - `preprocessing.py` .... data preprocessing
  - `content-based.py` .... impelementation of content-based method
  - `embeddings.py` .... impelementation of neural network embeddings
  - `pattern-finder.py` .... implementation of figures generator
  - `settings.py` .... settings for implementation
- `thesis` .... the directory of LaTeX source codes of the thesis