# Hotel Recommendation System

Jaskirat Singh
Department of Computing Science (Multimedia)
University of Alberta
Edmonton, Canada 30332–0250
Email: jaskira2@ualberta.ca

Pamila Tina
Department of Computing Science (Multimedia)
University of Alberta
Edmonton, Canada 30332–0250
Email: pviswana@ualberta.ca

Samarth Patel
Department of Computing Science (Multimedia)
University of Alberta
Edmonton, Canada 30332–0250
Email: samarth3@ualberta.ca

***Abstract*—Due to advancements in the field of technology, Internet has become a vital part of our daily life. People's online presence has increased to many folds in the last few decades. People surf the internet to obtain the most relevant and accurate information on the subject matter and this can be accomplished very easily by using Recommendation Systems(RS). RS are information filtering systems that filter relevant information fragments from a large set of dynamically generated data based on user's preferences, interests, or past observed behavior. RS can predict whether a particular user would prefer to select/buy a particular item and recommend them items accordingly. We were intrigued by the working and concept of the recommender system so we selected it as a topic for our project. We have developed a web-based application in which users can enter some features and will be able to get recommendations intuitively. Our application will allow users to interact and get recommendations dynamically in real-time depending upon the input of users.**

***Index Terms*** **–** Recommendation Systems, Content-Based Filtering, Collaborative Filtering, , Hybrid Filtering.

## I. Introduction

We selected the recommender system as the application domain for the project as RS is currently being utilized in many sectors for suggesting relevant items to the users, enriching the user experience, and enhancing their business. eCommerce companies such as Amazon, Flipkart & eBay are using RS for recommending products, YouTube uses RS for recommending videos, Facebook, Instagram uses RS for recommending friends, Netflix, Hotstar & AmazonPrime are using RS for recommending Movies, TVShows, etc., Airbnb uses RS for recommending rooms and hotels. We have trained our model on the data present in the database and then we are using our model to give out recommendations accordingly.

### A. Motivation

- *What should be the balance between serendipity and familiarity?*

Whenever we make a recommendation, we aim to tailor it to the user's preferences, thus as a result if we were to fit our model solely on user-centered data approaches such as calculating user-user similarity score or using user-features such as age, gender, etc. then we encounter the problems such as cold-start wherein calculating similarity for a new user is not possible, and Outlier bias wherein a user with only one booking would get a similarity score higher (100%) compared to other users with say twenty bookings, but with only one different choice (95%). Moreover, a similar scenario is encountered when we model our system only on item-centered data methods such as calculating item-item similarities where we again encounter the cold-start problem or item-features based approach such as recommending based on ratings, in which case the system suffers from popularity bias i.e. New items do not get recommended and as a result, newer hotels find it harder to be discovered, thus might need to spend exorbitant sums of capital on promotions. .

## II. Related Work

The internet provides the user with a plethora of options to choose while the RS makes it easier for the user by providing them suggestions based on their interests. RS is broadly divided into four categories – Content-Based (CBF), Collaborative (CF), Demographic and Hybrid Filtering.

RS has been researched and developed from the early '90s [Goldberg et al., 1992] manually and GroupLens and Ringo gradually automating it in 1994 and 1995 respectively. Both were memory-based approaches. Later came Content-based or information-based filtering RS, which uses the user's query or other information for the recommendation (Mooney and Roy, 2000). While each system had drawbacks such as the system would fail if there is not much knowledge of the user or sometimes due to community endorsements. In 2000, [5] developed 'personality diagnosis' type CF in combination with a model-based approach. In this method, when a new item is not seen during training then it would have a cold start problem.

[6] provides a detailed view on Content-based RS along with the techniques used and also the merits and demerits in contrast with other RS. [3] proposed a CBF recommendation algorithm along with Hidden Markov Model (HMM). However, the HMM has few limitations like unable to express dependencies between two hidden states and also have a large number of parameters.

[1] have done a good comparison on the two most commonly used approaches, content-based and collaborative filtering giving insights on the advantages and disadvantages of both the approaches. The same year, [4] gave a content-based approach for Twitter user recommendation using tweets, where they combined two algorithms such as noun phrase detector

and Naïve Bayes Text Classifier to detect tweets. Lately in 2020, [2] have developed an overview of user-based collaborative filtering for book selection that is currently employed in the online book store for which it was hard to include side features. Keeping [1] in consideration and all the drawbacks mentioned above, we propose a Hybrid filtering approach using both content and collaborative filtering methods in a way that one approach will nullify the disadvantages of the other.

A featured prediction competition of Expedia hotel recommendation was held on 2016 where 1971 teams participated. The highest accuracy that was achieved was 60 percent and the average accuracy was around 48 percent.The method that won the competition was using Distance Matrix Completion to map the users and hotel location. For each set of hotel clusters library a Field-aware Factorization Machines (LIBFFM) implementation was done using the categorical features in the training set. Another approach which used the XGBoost algorithm but with a different approach was collaborative filtering by using a pivot table to find out all the clusters the user has clicked and merge it with both the training and test set.

## III. Relation to Our Other Projects

This project is built on Django framework from scratch and has no relation to any of our other projects.

## IV. Development Work and Novelty

### A. *Tools and Resources*

- GitHub Link for Source Code: Recommender System
- Dataset: Recommender System Dataset
- Visualization Library: charts.js
- Web Development:
  - *Framework*: Django
  - *Front-End:* HTML, CSS, Bootstrap, JavaScript
  - *Back-End*: Python
  - *Database*: SQLite

### B. *Django Framework Setup*

Primary task is to perform setup of Django framework which can be done by following sequence of steps given below:

- ***Step 1:*** **Create and Activate Virtual Environment:** We need to create new virtaula environment which can done by running this command **"conda create -n my_django_environment python=3.9"** ans we need to activate virtual environment in order to work in it. We can activate and deactivate Virtual environments using following commands. **"conda activate my_django_environment"** and **"conda deactivate"**.
- ***Step 2:*** **Install Dependencies**
  After creating virtual environment next step is to set up environment by installing all the dependencies. We can do that by executing **"pip install django =3.1"** command in the Command Prompt.

- ***Step 3:*** **Start Project**
  We have setup the environment now it is time to create project. We can create project by executing **"django-admin startproject HRS"** command and project will be created with below given file structure.

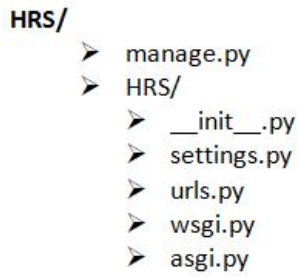


Fig. 1. File Structure of Project

The HRS project sub-folder is the entry point for the website:
**"_init_.py"** is an empty file that instructs Python to treat this directory as a Python package.
**"settings.py"** contains all the website settings, including registering any applications we create, the location of our static files, database configuration details, etc.
**"urls.py"** defines the site URL-to-view mappings. While this could contain all the URL mapping code, it is more common to delegate some of the mappings to particular applications, as you'll see later.
**"wsgi.py"** is used to help your Django application communicate with the webserver. You can treat this as boilerplate.
**"asgi.py"** is a standard for Python asynchronous web apps and servers to communicate with each other.
**"manage.py"** script is used to create applications, work with databases, and start the development web server.

- ***Step 4:*** **Create Application**
  After project creation we need to navigate to HRS folder and here we can create multiple independent applications which can used in same project. Application can be created using **"python manage.py startapp catalog"** command.
  **css** folder will conatin all css files which can be imported in HTML files which are present in **templates** folder.
  **images** folder will contain images for the whole project.
  **migrations** folder is used to store "migrations" — files that allow you to automatically update the database as you modify your models.

- ***Step 5:*** **Add Data to project**

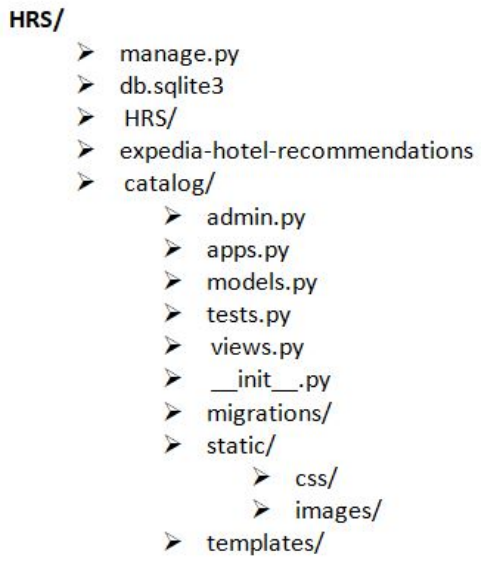


Fig. 2. File Structure of Project

We need to download dataset from the link given in "Tools and Resources" section and place it in "expedia-hotel-recommendations" folder.
SQLite database is default database in Django framework and we are using "DB Browser for SQLite" tool for managing data as it provide interactive user interface where it is very easy to interact and visualize data in the tables of database. Open "db.sqlite3" database present in HRS folder and Import the .csv data into database.
If any major changes are done in database then we need to run below commands so that Django framework and database remain in sync.
**"python manage.py makemigrations"**
**"python manage.py migrate"**

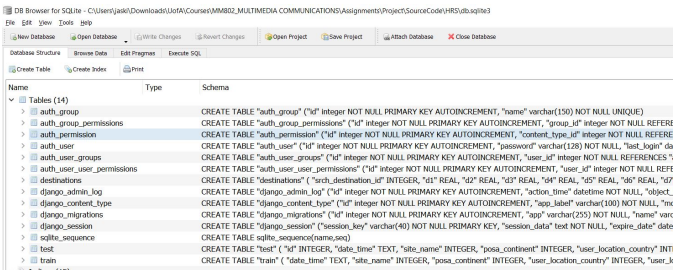

Fig. 3. Tables in Databaset

- ***Step 6:*** **Run Server**
  We can run the server using command **"python manage.py runserver"** and server will be started on default port 8000. We can access project using link: http://127.0.0.1:8000/
  and if we want to run server on any specific port we can use this command **"python manage.py runserver 8080"**. Now server will run on port 8080.

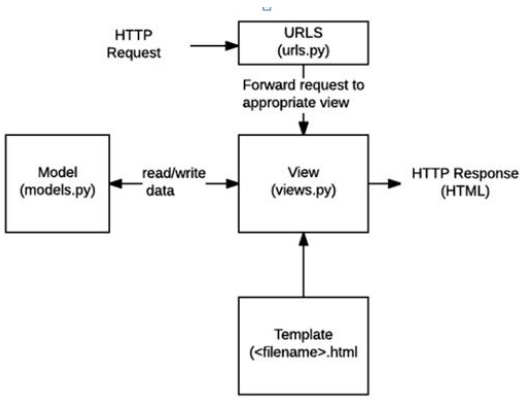


Fig. 4. Django Framework

### C. *Task Breakdown*

The breakdown of tasks is as follows:

**Jaskirat Singh**

- Data Collection
- Setup of Django Framework
- User Interface Design and Implementation
- Implementation of HRS
- Deployment and Integration Testing
- Report Writing

**Samarth Patel**

- Data Preprocessing
- Implementation of HRS
- Tuning of Hyper parameters for HRS
- Integration of UI and HRS
- Unit Testing
- Report Writing

**Pamila Viswanathan**

- Data Validation
- Implementation of HRS
- Integration of UI and HRS
- Bar Chart Visualization using Charts.js
- Report writing.

### D. *Dataset*

The proposed recommender system dataset is taken from Kaggle. There are three CSV (comma-separated values) files which are as follows:

| Name | No. of rows | No. of Columns | Size |
|---|---|---|---|
| *train.csv* | 37,670,293 | 24 | 3.79 GB |
| *test.csv* | 2,528,243 | 22 | 263.74 MB |
| *destinations.csv* | 62,109 | 150 | 131.76 MB |

The *train.csv* and *test.csv* files contain features that can be categorized into 3 main categories:

- Geographical Features
- Temporal Features
- Search Features

The description of each of the features in *train.csv* and *test.csv* files is given in the table I.

| Feature | Description | Datatype |
|---|---|---|
| date_time | Timestamp | string |
| site_name | ID of the Expedia point of sale (i.e. Expedia.com, Expedia.co.uk, Expedia.co.jp, ...) | int |
| posa_continent | ID of continent associated with site_name | int |
| user_location_country | The ID of the country the customer is located | int |
| user_location_region | The ID of the region the customer is located | int |
| user_location_city | The ID of the city the customer is located | int |
| orig_destination_distance | Physical distance between a hotel and a customer at the time of search. A null means the distance could not be calculated | double |
| user_id | ID of user | int |
| is_mobile | 1 when a user connected from a mobile device, 0 otherwise | tinyint |
| is_package | 1 if the click/booking was generated as a part of a package (i.e. combined with a flight), 0 otherwise | int |
| channel | ID of a marketing channel | int |
| srch_ci | Checkin date | string |
| srch_co | Checkout date | string |
| srch_adults_cnt | The number of adults specified in the hotel room | int |
| srch_children_cnt | The number of (extra occupancy) children specified in the hotel room | int |
| srch_rm_cnt | The number of hotel rooms specified in the search | int |
| srch_destination_id | ID of the destination where the hotel search was performed | int |
| srch_destination_type_id | Type of destination | int |
| hotel_continent | Hotel continent | int |
| hotel_country | Hotel country | int |
| hotel_market | Hotel market | int |
| is_booking | 1 if a booking, 0 if a click | tinyint |
| cnt | Numer of similar events in the context of the same user session | bigint |
| hotel_cluster | ID of a hotel cluster | int |

TABLE I
FEATURES IN *train.csv* AND *test.csv*

The *destinations.csv* file contains 149 latent features that have been extracted from reviews of hotel given by customers, relating to cleanliness, distance from tourist attractions, etc. The label and values for each of those features are anonymized, thus what each of those features pertain to exactly is not known.

### *E. Pre-processing of Data*

Upon analysing the data, it was found that it contained empty rows with Nan Values. Moreover upon analysing the data-set further it was found that not all rows resulted in a booking being made. Thus the data was prepossessed by applying a filter that selected rows with the *is_booking* feature value *1*. Subsequently the features were converted to integer format in order to ensure the model can ingest the values.

### *F. Feature Selection*

After processing the data, in order to select appropriate features, a correlation matrix was created. The features that most correlated to the 'hotel_cluster' parameter were chosen as features for the model. Below is the feature correlation heatmap for the complete dataset.

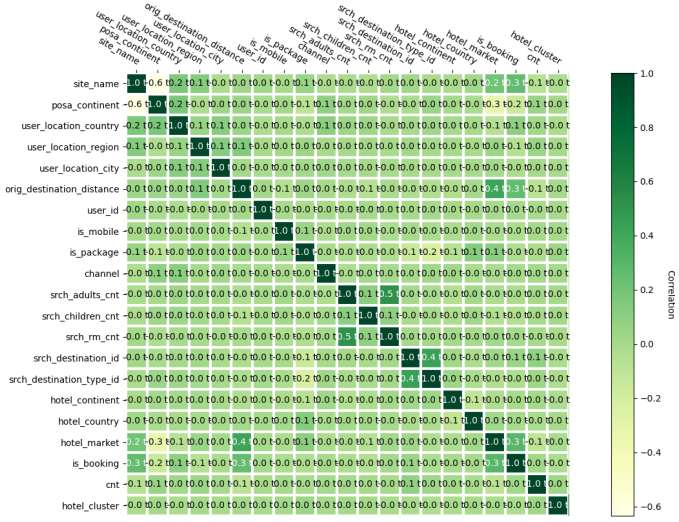


Fig. 5. Heatmap of Feature Correlation Matrix

### *G. Implemented System*

The implemented system consists of Django Web App, which is integrated with the recommendation system using a system call *run* command using the python module *run* and subsequently pipes the output to a variable in *views.py*. When the user visits the web application, they are greeted by a home page, upon clicking the hotels tab they are take to a form, that takes in the values of features such as *user_location_region*,*srch_children_cnt*,*src_destination_id* etc. After submitting the form, the request is routed to Urls.py in the HRS project directory, the name corresponding to the request is searched there, upon finding a application's urls.py linked the request is routed to the application. After finding the request with the matching name the associated function is executed form views.py.in the catalog web application directory. From views.py predict_result function executes the model to predict the recommended hotel cluster value.

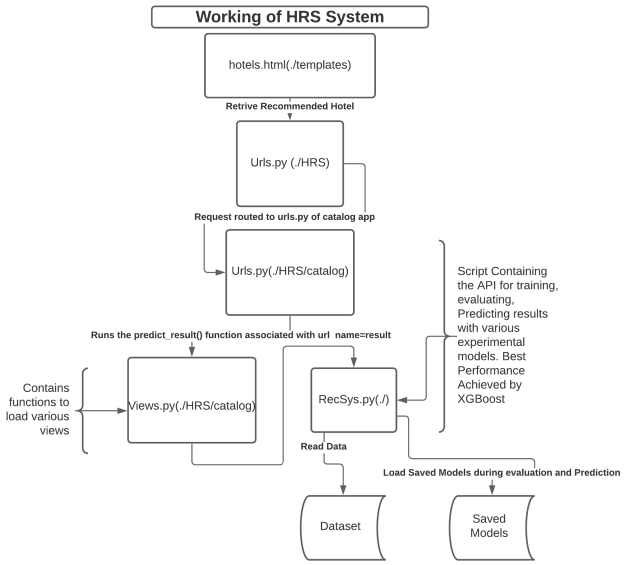


Fig. 6. Working of HRS System

### H. *Novelty*

To the best of our knowledge following aspects are a novel contribution:

- **Feature Selection:** All the past approaches exploit the data leakage in the *orig_destination_distance* column or create pivot tables containing grouped columns such as count of bookings for a particular hotel column.
- **Integration with a Web-application to present the results:** Previous attempts at building Recommender System using this particular dataset was focused on getting the highest possible accuracy, but made no attempt to present teh results as a web-application.

### I. *Experimental Results*

The accuracy results for various models are depicted below: As shown in *table 2* best performance is achieved by Extreme

| Model Name | Accuracy |
|---|---|
| SVM | 3-12% |
| SGD | 0-8% |
| KNN | 8-12% |
| XGB | 42-53% |

TABLE II
EXPERIMENTAL RESULTS

Gradient Boost algorithm (XGB) for the given feature selection. Here we do not exploit the data leakage of of about 1/3 dataset form *orig_destination_distance* column. Moreover below is a test run of all the models during testing along with the predictions for a test row.

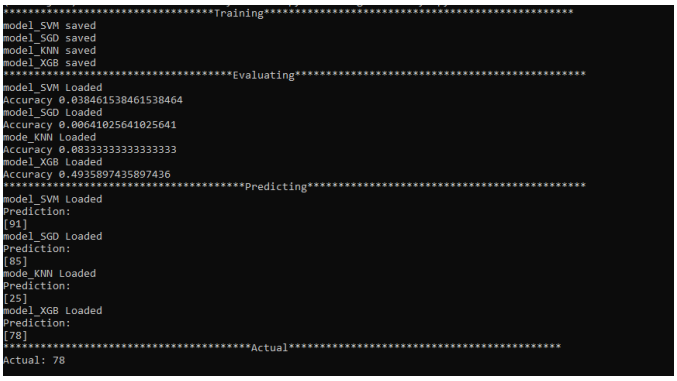


Fig. 7. Test results for different models

### J. *Contribution*

One of the contributions is the attempt at implementing UI to present the results of the recommender model. Which explored the challenges associated with presentation of anonymized data and why having a human readable dataset is important for making sense of various features included in the dataset.

## V. PROJECT STATUS

The basic functions have been achieved, but there is still room for improvement, such as increasing the accuracy of the recommender system by recognizing the most relevant features and training model by taking them into account, the user interface can be improved, and so on. Integration of standalone script and the Django was a very challenging part but the part where we faced the most difficulty is the training model with appropriate features and obtaining recommendations matching the ground truth values.

### A. *Screenshots*

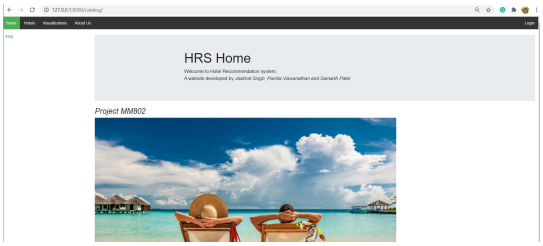


Fig. 8. Homepage

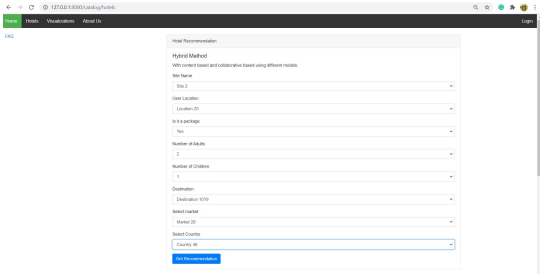

Fig. 9. Hotels Page

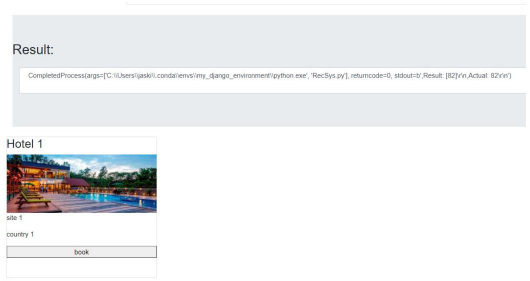


Fig. 10. Hotels page Results

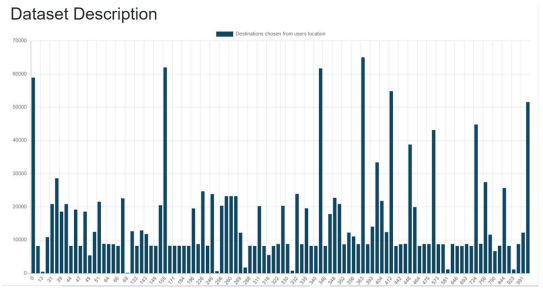


Fig. 11. Visualization of Data

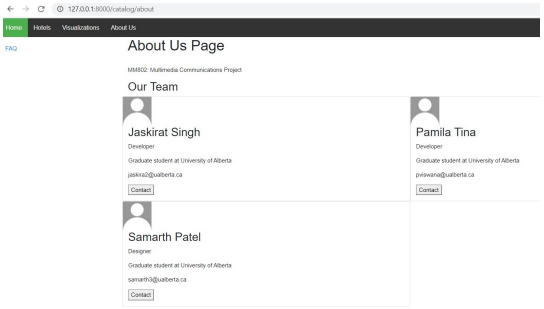


Fig. 12. About Us Page

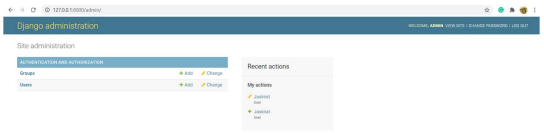


Fig. 13. Admin Console

## VI. Concluding Remarks

We would like to say that we have learned a lot while working in a team on this project. Our goal was to develop a hotel recommender system in which users can get recommendations based on the features provided. For the creation of web based application, we selected the Django framework which is a high-level open-source Python-based Web framework that encourages rapid development and clean, pragmatic design. While working on this project we have learned to work on Model-Template-Views architectural pattern and we were able to integrate our standalone recommender system into the framework to get recommendations on the user interface. We got more practical exposure about the framework and learned how to send fetch data from the default SQLite database and pass it to the server-side which can further pass it to the client-side where it can be displayed on the screen. We have barely scratched the surface there is a lot of room for improvement. In the future, We would like to try optimizing our algorithm and achieve higher accuracy. Moreover, User Interface can also be improved for users, and login & session management can also be enabled.

## References

[1] P. Aggarwal, V. Tomar, and A. Kathuria. Comparing content based and collaborative filtering in recommender systems. In *International Journal of New Technology and Research (IJNTR)*, volume 3, pages 65–67, 2017.

[2] M. Kommineni, P. Alekhya, T. M. Vyshnavi, V. Aparna, K. Swetha, and V. Mounika. Machine learning based efficient recommendation system for book selection using user based collaborative filtering algorithm. In *2020 Fourth International Conference on Inventive Systems and Control (ICISC)*, pages 66–71, 2020.

[3] H. Li, F. Cai, and Z. Liao. Content-based filtering recommendation algorithm using hmm. In *2012 Fourth International Conference on Computational and Information Sciences*, pages 275–277, 2012.

[4] R. H. Nidhi and B. Annappa. Twitter-user recommender system using tweets: A content-based approach. In *2017 International Conference on Computational Intelligence in Data Science(ICCIDS)*, pages 1–6, 2017.

[5] David M. Pennock, Eric Horvitz, Steve Lawrence, and C. Lee Giles. Collaborative filtering by personality diagnosis: A hybrid memory and model-based approach. In *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence*, UAI '00, page 473–480, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.

[6] Charilaos Zisopoulos, Savvas Karagiannidis, Georgios Demirtsoglou, and Stefanos Antaris. Content-based recommendation systems. *11*, 2008.