Turnitin Originality Report

Processed on: 01-Jun-2020 16:58 IST
ID: 1335319237
Word Count: 12293
Submitted: 2

Other_check By Btech2020 CR

Similarity Index	Similarity by Source	
31%	Internet Sources:	24%
	Publications:	14%
	Student Papers:	24%

3% match (Internet from 23-Oct-2010)
http://www.inf.unibz.it/~ricci/ISR/slides-2010/14-ContentBased-Hybrid.pdf

2% match (Internet from 04-May-2020)
https://towardsdatascience.com/creating-a-hybrid-content-collaborative-movie-recommender-using-deep-learning-cc8b431618af?gi=c3ae7f33de53

2% match (Internet from 12-Jul-2010)
http://www.inf.unibz.it/~ricci/ATIS/2008-2009/6-HybridMethods.pdf

1% match (Internet from 27-May-2020)
https://developers.google.com/machine-learning/recommendation/collaborative/basics

1% match (publications)
Yassine Afoudi, Mohamed Lazaar, Mohamed Al Achhab. "Impact of Feature selection on content-based recommendation system", 2019 International Conference on Wireless Technologies, Embedded and Intelligent Systems (WITS), 2019

1% match (Internet from 13-May-2020)
https://www.datacamp.com/community/tutorials/recommender-systems-python

1% match (student papers from 29-Apr-2019)
Submitted to Visvesvaraya National Institute of Technology on 2019-04-29

1% match (student papers from 26-Dec-2019)
Submitted to The Scientific & Technological Research Council of Turkey (TUBITAK) on 2019-12-26

1% match (Internet from 25-Mar-2019)
http://ijiet.com/wp-content/uploads/2016/12/20.pdf

1% match (student papers from 20-Nov-2019)
Submitted to Universiti Teknologi MARA on 2019-11-20

1% match (student papers from 27-May-2019)
Submitted to Monash University on 2019-05-27

1% match (Internet from 14-Apr-2020)
https://en.wikipedia.org/wiki/Recommender_system

1% match (student papers from 07-Jun-2017)
Submitted to National Research University Higher School of Economics on 2017-06-07

1% match (Internet from 10-Dec-2012)
http://www.grouplens.org/system/files/FnT%20CF%20Recsys%20Survey.pdf

1% match (Internet from 18-Nov-2015)
http://www.slideshare.net/TommyPickles1/ieeecomputerltx

1% match (Internet from 03-Jan-2019)
http://recommender-systems.org/hybrid-recommender-systems/

< 1% match (Internet from 08-Feb-2020)

http://www.ijresm.com/vol1iss5May18/IJRESM15_20.pdf

< 1% match (Internet from 11-Jun-2015)
http://www.powershow.com/view/19bfe-NGJiO/Introduction_to_Recommender_Systems_powerpoint_ppt_presentation

< 1% match (Internet from 15-Apr-2015)
http://www.m8j.net/data/List/Files-149/Matrix-Factorizations-Netflix-Hazan.pdf

< 1% match (Internet from 06-Feb-2019)
http://ijesrt.com/issues%20pdf%20file/Archive-2016/November-2016/63.pdf

< 1% match (student papers from 14-Apr-2019)
Submitted to Cardiff University on 2019-04-14

< 1% match (student papers from 25-Apr-2017)
Submitted to Visvesvaraya National Institute of Technology on 2017-04-25

< 1% match (student papers from 29-Aug-2014)
Submitted to King's College on 2014-08-29

< 1% match (publications)
Schirru, Rafael. "Contextualized Recommendations for the Socio-Semantic Web", KLUEDO, 2013.

< 1% match (student papers from 28-Apr-2019)
Submitted to Visvesvaraya National Institute of Technology on 2019-04-28

< 1% match (Internet from 01-Nov-2018)
https://fenix.tecnico.ulisboa.pt/downloadFile/563345090413333/Thesis.pdf

< 1% match (student papers from 27-Apr-2019)
Submitted to Visvesvaraya National Institute of Technology on 2019-04-27

< 1% match (student papers from 04-May-2016)
Submitted to Visvesvaraya National Institute of Technology on 2016-05-04

< 1% match (Internet from 14-Nov-2019)
https://www.bluepiit.com/blog/demystifying-hybrid-recommender-systems-and-their-use-cases/?repeat=w3tc

< 1% match (Internet from 24-May-2016)
http://docplayer.net/10892388-By-the-end-of-the-placement-in-july-2002-a-beta-version-was-being-prepared-for-testing-sites.html

< 1% match (student papers from 19-Apr-2020)
Submitted to University of Wales, Bangor on 2020-04-19

< 1% match (publications)
"Emerging Research in Computing, Information, Communication and Applications", Springer Science and Business Media LLC, 2019

< 1% match (Internet from 02-May-2020)
https://towardsdatascience.com/introduction-to-recommender-systems-6c66cf15ada?gi=e2f9726c6dba

< 1% match (Internet from 25-Apr-2020)
https://patents.google.com/patent/US20190034436A1/en

< 1% match (publications)
Panagiotis Symeonidis, Andreas Zioupos. "Matrix and Tensor Factorization Techniques for Recommender Systems", Springer Science and Business Media LLC, 2016

< 1% match (student papers from 18-Mar-2020)
Submitted to University of Colorado, Denver on 2020-03-18

< 1% match (Internet from 16-May-2020)
http://ibomalkoc.com/movies-dataset/

< 1% match (publications)
"Encyclopedia of Social Network Analysis and Mining", Springer Science and Business Media LLC, 2018

< 1% match (publications)
Jonathan L. Herlocker. "Evaluating collaborative filtering recommender systems", ACM Transactions on Information Systems, 1/1/2004

< 1% match (Internet from 12-Mar-2019)
https://www.slideshare.net/BenditoFreitasRibeir/content-based-filtering

< 1% match (student papers from 09-Sep-2018)
Submitted to University of Southampton on 2018-09-09

< 1% match (student papers from 10-May-2013)
Submitted to University of Warwick on 2013-05-10

< 1% match (student papers from 16-Jan-2020)
Submitted to University of Northumbria at Newcastle on 2020-01-16

< 1% match (Internet from 21-May-2020)
https://pure.tue.nl/ws/portalfiles/portal/109407567/CSE671_Master_Thesis_Alessandro_Terragni.pdf

< 1% match (student papers from 11-Apr-2020)
Submitted to Liverpool John Moores University on 2020-04-11

< 1% match (student papers from 01-Sep-2015)
Submitted to Kwame Nkrumah University of Science and Technology on 2015-09-01

< 1% match (Internet from 28-Sep-2019)
https://toolbox.google.com/datasetsearch/search?query=Movie+Ratings+Dataset

< 1% match (student papers from 20-Apr-2019)
Submitted to Visvesvaraya National Institute of Technology on 2019-04-20

< 1% match (Internet from 04-Nov-2017)
https://dl.gi.de/bitstream/handle/20.500.12116/939/lni-p-266-komplett.pdf?isAllowed=y&sequence=1

< 1% match (student papers from 12-Jul-2015)
Submitted to ABV-Indian Institute of Information Technology and Management Gwalior on 2015-07-12

< 1% match (student papers from 19-Jan-2012)
Submitted to University of Edinburgh on 2012-01-19

< 1% match (student papers from 02-Feb-2010)
Submitted to University of Abertay Dundee on 2010-02-02

< 1% match (student papers from 08-May-2020)
Submitted to CSU, Pomona on 2020-05-08

< 1% match (Internet from 31-Oct-2017)
https://link.springer.com/content/pdf/10.1007%2F978-0-387-85820-3.pdf

< 1% match (Internet from 31-Oct-2019)
https://gemslab.github.io/papers/koutra-2017-pnp.pdf

< 1% match (Internet from 22-Sep-2019)
http://export.arxiv.org/abs/1907.13286v2

< 1% match (student papers from 19-Jan-2012)
Submitted to University of Edinburgh on 2012-01-19

< 1% match (Internet from 12-Sep-2018)
http://www.ijritcc.org/download/browse/Volume_5_Issues/May_17_Volume_5_Issue_5/1496218776_31 05-2017.pdf

< 1% match (Internet from 07-Dec-2017)
http://scholarbank.nus.edu.sg/bitstream/10635/34369/1/Thesis_LeeLipKim_HT060742U.pdf

< 1% match (publications)
Chen, Y.. "Machine learning techniques for business blog search and mining", Expert Systems With Applications, 200810

< 1% match (publications)
The Electronic Library, Volume 31, Issue 3 (2013-10-05)

< 1% match (Internet from 27-May-2020)
https://developers.google.com/machine-learning/recommendation/collaborative/matrix

< 1% match (Internet from 19-Feb-2020)
https://journals.sagepub.com/doi/10.1177/1932296818789951?icid=int.sj-full-text.similar-articles.1

< 1% match (student papers from 28-May-2018)
Submitted to CTI Education Group on 2018-05-28

< 1% match (student papers from 11-May-2016)
Submitted to University of Nottingham on 2016-05-11

< 1% match (publications)
Recommender Systems Handbook, 2011.

< 1% match (student papers from 24-Apr-2015)
Submitted to University of Warwick on 2015-04-24

< 1% match (Internet from 14-Mar-2020)
https://www.mid-day.com/articles/kajol-excited-to-work-with-shah-rukh-khan-in-dilwale/16076166

< 1% match (Internet from 07-Apr-2019)
https://www.grin.com/document/280808

< 1% match (publications)
Lecture Notes in Computer Science, 2015.

< 1% match (student papers from 11-Sep-2018)
Submitted to University of Birmingham on 2018-09-11

< 1% match (student papers from 05-Dec-2019)
Submitted to University of Lincoln on 2019-12-05

< 1% match (Internet from 08-May-2019)
https://tel.archives-ouvertes.fr/tel-00789726/document

< 1% match (Internet from 04-Sep-2019)
http://ikee.lib.auth.gr/record/286955/files/GRI-2017-18235.pdf

< 1% match ()
http://hdl.handle.net/2142/89100

< 1% match (publications)
Bruno Veloso, Benedita Malheiro, Juan Carlos Burguillo. "A multi-agent brokerage platform for media content recommendation", International Journal of Applied Mathematics and Computer Science, 2015

< 1% match (student papers from 26-Apr-2010)
Submitted to Napier University on 2010-04-26

< 1% match (student papers from 02-Apr-2019)
Submitted to Ganpat University on 2019-04-02

< 1% match (student papers from 01-May-2020)
Submitted to Monash University on 2020-05-01

< 1% match (student papers from 24-Feb-2015)
Submitted to North-Eastern Hill University, Shillong on 2015-02-24

< 1% match (student papers from 27-Oct-2011)
Submitted to La Trobe University on 2011-10-27

< 1% match (student papers from 23-Jul-2009)
Submitted to Queen Mary and Westfield College on 2009-07-23

< 1% match (student papers from 03-Sep-2016)
Submitted to IIT Delhi on 2016-09-03

< 1% match ()
http://ethesis.nitrkl.ac.in/5734/

< 1% match (Internet from 07-May-2019)
https://ulir.ul.ie/bitstream/handle/10344/5224/Alahmadi_2016_automatic.pdf?sequence=6

< 1% match (Internet from 06-Sep-2017)
https://www.math.uci.edu/icamp/courses/math77b/lecture_12w/pdfs/Chapter%2003%20-%20Content-based%20recommendation.pdf

< 1% match (Internet from 03-Apr-2019)
https://www.suggestmemovie.com/film/1081/Toy-Story/

< 1% match (publications)
Can Bozdogan, A. Nur Zincir-Heywood, Ibrahim Zincir. "EMITS: An Experience Management System for IT Management Support", International Journal of Software Engineering and Knowledge Engineering, 2015

< 1% match (student papers from 30-May-2020)
Submitted to Visvesvaraya National Institute of Technology on 2020-05-30

< 1% match (student papers from 02-Jun-2017)
Submitted to CITY College, Affiliated Institute of the University of Sheffield on 2017-06-02

< 1% match (student papers from 15-Feb-2015)
Submitted to Bocconi University on 2015-02-15

< 1% match (student papers from 08-May-2006)
Submitted to University of Warwick on 2006-05-08

< 1% match (student papers from 13-Mar-2020)
Submitted to University of Wales Institute, Cardiff on 2020-03-13

< 1% match (Internet from 07-Apr-2020)
https://www.coursehero.com/file/39042280/svd-handoutpdf/

< 1% match (Internet from 11-Feb-2019)
http://cs229.stanford.edu/proj2018/report/128.pdf

< 1% match (Internet from 01-Jun-2020)
https://datamafia2.wordpress.com/author/datamafia007/

< 1% match (Internet from 14-Apr-2010)
http://csmr.ca.sandia.gov/~tgkolda/pubs/bibtgkfiles/umcp-cs-tr-3724.pdf

< 1% match (Internet from 08-Sep-2019)
http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.463.1205&rep=rep1&type=pdf

< 1% match (publications)
Sushmita Roy, Mahendra Sharma, Santosh Kumar Singh. "Movie Recommendation System Using Semi-Supervised Learning", 2019 Global Conference for Advancement in Technology (GCAT), 2019

< 1% match (publications)
Son Doan. "A General Fuzzy-Based Framework for Text Representation and Its Application to Text Categorization", Lecture Notes in Computer Science, 2006

< 1% match (publications)
Lina Zhou, Yin Xiao, Wen Chen. "Imaging Through Turbid Media With Vague Concentrations Based on Cosine Similarity and Convolutional Neural Network", IEEE Photonics Journal, 2019

< 1% match (student papers from 05-Mar-2016)
Submitted to South Bank University on 2016-03-05

< 1% match (student papers from 06-May-2012)
Submitted to University of Hong Kong on 2012-05-06

< 1% match (student papers from 21-Jan-2016)
Submitted to Cranfield University on 2016-01-21

< 1% match (student papers from 16-Jun-2013)
Submitted to Universiti Tunku Abdul Rahman on 2013-06-16

< 1% match (student papers from 27-Jun-2016)
Submitted to IIT Delhi on 2016-06-27

< 1% match (student papers from 05-May-2014)
Submitted to CSU, San Jose State University on 2014-05-05

< 1% match (student papers from 02-Aug-2019)
Submitted to University of Computer Studies on 2019-08-02

< 1% match (student papers from 21-Nov-2006)
Submitted to University of Toronto on 2006-11-21

< 1% match (student papers from 16-Dec-2015)
Submitted to University of Warwick on 2015-12-16

< 1% match ()
http://ethesis.nitrkl.ac.in/6534/

< 1% match (Internet from 31-Mar-2020)
http://wiredspace.wits.ac.za/bitstream/handle/10539/21650/Thesis.pdf?isAllowed=y&sequence=1

< 1% match (Internet from 14-Jan-2019)
https://eprints.qut.edu.au/93723/1/Wanvimol_Nadee_Thesis.pdf

< 1% match (Internet from 16-Nov-2019)
https://www.juet.ac.in/ConfWorkshop/Conference/ITBI-2013/ITBI-2013.pdf

< 1% match (Internet from 31-May-2020)
http://docplayer.net/4145680-Data-hiding-current-status-and-key-issues.html

< 1% match (Internet from 19-Jan-2020)
https://link.springer.com/content/pdf/10.1007%2F978-3-540-72079-9.pdf

< 1% match (Internet from 30-Aug-2019)
http://www.isle.org/~langley/papers/apa.jair04.pdf

< 1% match (Internet from 14-Apr-2018)
https://tel.archives-ouvertes.fr/tel-01135312/document

< 1% match (Internet from 12-Apr-2016)
https://boa.unimib.it/retrieve/handle/10281/70560/104693/PhD_unimib_%20%09732150%20.pdf

< 1% match (Internet from 19-Oct-2018)
https://repositorio.ufpe.br/bitstream/123456789/2654/1/arquivo5808_1.pdf

< 1% match (Internet from 09-Jan-2014)
http://repository.tudelft.nl/assets/uuid:f7d3977e-f191-40d4-8f27-784a32902a55/thesis_yueshi.pdf

< 1% match (Internet from 28-Aug-2017)

http://summit.sfu.ca/system/files/iritems1/16625/etd9734_YWu.pdf

< 1% match (Internet from 27-Mar-2020)
http://lexicometrica.univ-paris3.fr/jadt/JADT2018/actes-jadt18.pdf

< 1% match (publications)
Qiang Yang. "EigenRank", Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval - SIGIR 08 SIGIR 08, 2008

< 1% match (publications)
Varaprasad Rao M, Vishnu Murthy G. "chapter 3 DSS for Web Mining Using Recommendation System", IGI Global, 2017

< 1% match (publications)
Cappella, J. N., S. Yang, and S. Lee. "Constructing Recommendation Systems for Effective Health Messages Using Content, Collaborative, and Hybrid Algorithms", The Annals of the American Academy of Political and Social Science, 2015.

< 1% match (publications)
Lars Eldén. "Numerical linear algebra in data mining", Acta Numerica, 2006

< 1% match (publications)
Zheng Pei. "Handling linguistic web information based on a multi-agent system", International Journal of Intelligent Systems, 05/2007

< 1% match (publications)
Eduardo R. Soares, Eduardo Barrére. "Automatic Topic Segmentation for Video Lectures Using Low and High-Level Audio Features", Proceedings of the 24th Brazilian Symposium on Multimedia and the Web - WebMedia '18, 2018

< 1% match (publications)
Sreenivas Gollapudi, Rina Panigrahy. "Exploiting asymmetry in hierarchical topic extraction", Proceedings of the 15th ACM international conference on Information and knowledge management - CIKM '06, 2006

< 1% match (student papers from 08-May-2020)
Submitted to University of Reading on 2020-05-08

< 1% match (student papers from 05-May-2011)
Submitted to Asian Institute of Technology on 2011-05-05

< 1% match (student papers from 08-Jun-2015)
Submitted to Higher Education Commission Pakistan on 2015-06-08

< 1% match (student papers from 13-May-2014)
Submitted to iGroup on 2014-05-13

< 1% match (student papers from 13-Sep-2018)
Submitted to University of Warwick on 2018-09-13

< 1% match (student papers from 03-May-2013)
Submitted to University of Bath on 2013-05-03

< 1% match (student papers from 09-Dec-2013)
Submitted to University College London on 2013-12-09

< 1% match (student papers from 27-Apr-2010)
Submitted to Nottingham Trent University on 2010-04-27

Scalable Recommendation System using Hybrid Approach: Content and Collaborative Filtering Project report submitted to Visvesvaraya National Institute Of Technology, Nagpur In partial fulfillment of the requirements for the award of the degree Bachelor of Technology In Computer Science and Engineering Chaitya Chheda Ayush Singh Deval Mudia Sadneya Pusalkar under the guidance of Dr. S.R. Sathe BT16CSE016 BT16CSE098 BT16CSE060 BT16CSE076 Department of Computer Science and Engineering Visvesvaraya National Institute Of Technology Nagpur 440 010 (India) 2020 © Visvesvaraya National Institute Of Technology (VNIT) 2009 Department of

Computer Science and Engineering Visvesvaraya National Institute Of Technology, Nagpur DECLARATION We hereby declare that this project work titled Scalable Recommendation System using Hybrid Approach: Content and Collaborative Filtering is carried out by us in the Department of Computer Science and Engineering of Visvesvaraya National Institute of Technology, Nagpur under the guidance of Dr. S.R. Sathe. The work is original and has not been submitted earlier whole or in part for the award of any degree/diploma at this or any other Institution/University. Sr.No. Enrollment No Names Signature 1 BT16CSE016 Chaitya Chheda 2 BT16CSE098 Ayush Singh 3 BT16CSE060 Deval Mudia 4 BT16CSE076 Sadneya Pusalkar Date: Department of Computer Science and Engineering Visvesvaraya National Institute Of Technology, Nagpur CERTIFICATE This is to certify that the project work entitled- “Scalable Recommendation System using Hybrid Approach: Content and Collaborative Filtering”, is a bonafide work done by Mr. Chaitya Chheda, Ms. Sadneya Pusalkar, Mr. Deval Mudia and Mr. Ayush Singh under the guidance of Dr. S.R. Sathe in the Department of Computer Science & Engineering, Visvesvaraya National Institute of Technology, Nagpur, for the partial fulfilment of the requirements for the award of the degree of Bachelor of Technology in “Computer Science &Engineering”. The work is comprehensive, complete, and fit for final evaluation. (Dr. U.A. Deshpande) Head Of Department Department Of Computer Science and Engineering, VNIT, Nagpur (Dr. S.R. Sathe) Project Guide Department Of Computer Science and Engineering, VNIT, Nagpur Date: ACKNOWLEDGMENTS We would like to express our sincere thanks and gratitude to our guide, Dr. S.R. Sathe for giving us the opportunity of working on a project that is widely applied in most online businesses, guiding us thoughtfully and giving us an opportunity to work at our own pace. We would also like to thank all the teaching and non-teaching faculty of the Computer Science and Engineering Department for supporting us and providing necessary facilities at all times. Last but not least we would like to acknowledge the contribution of those who have constantly supported and encouraged us which helped in the successful completion of our project. ABSTRACT A recommendation system helps users find compelling content in large corpora. In a very general way, recommender systems are algorithms aimed at suggesting relevant items to users, to ensure he/she consumes the content of specific interest and relevance. There are two major paradigms of recommender systems: collaborative and content-based methods. Collaborative filtering systems recommend items to a user that are preferred by other users with similar tastes. Content-based systems examine the properties of the items to recommend other similar items. However, these technologies by themselves are not sufficient and suffer from common problems like the “cold start problem”: impossible to recommend anything to new users or to recommend a new item, sparsity: feedback data is sparse and insufficient for identifying neighbours and popular bias: frequently rated items get a lot of exposure while less popular ones are under-represented In this thesis we discuss our implementation of a hybrid recommender system that leverages both content and collaborative data. We have chosen the base model for our scalable collaborative filter to be Singular Value Decomposition (SVD) since recommender systems based on SVD have proved to be versatile and efficient. However, building the SVD model suffers from being an extremely slow process. Therefore we used an incremental SVD algorithm based on “fold-in” technique that could append new users so they receive their recommendations on the spot instead of waiting for the model to be rebuilt. For content filtering, we represented each item as a set of descriptors or terms from the metadata available and used cosine similarity to calculate a numeric quantity that denotes the similarity between two items. To overcome the shortcomings of both the approaches we have implemented hybrid filtering by taking the weighted average of the content recommendations and collaborative recommendation. The evaluation of this system we chose predictive accuracy metrics. We evaluated the three approaches using two metrics : Mean Absolute Error (MAE) and Root Mean Square Error (RMSE). The MAE and RMSE values for the hybrid recommender system turned out to be 0.745 and 0.994 respectively. We found that hybrid filtering gave better results than content based filtering and collaborative filtering. CONTENTS Acknowledgments V Abstract VI 1 Introduction 1.1 Motivation . . . . 1 1.2 Project Aim . . . . 2 1.3 Organisation of Thesis . . . . 3 2 Literature Survey 2.1 Methods of Recommendation . . . . 4 2.2 Categorisation of Recommender Systems . . . . 2.3 Utility Matrix . . . . 2.3.1 Collecting data for Utility Matrix . . . . 2.4 Collecting data for Feature Representation . . . . 2.5 Content Based Filtering . . . . 2.5.1 Feature Representation . . . . 2.5 .1.1 Vector Space Model . . . . 2.5 .1.2 TF - IDF . . . . 2.5 .1.3 Term Frequency . . . . 2.5.1. 3 Inverse Document Frequency . .. . 2.5.2 Similarity . . . .

Information of Toy Story 4.1 Summary of Results Chapter 1 Introduction Imagine the following scenario, you are currently in the mood for a new movie or TV series to begin watching. However, there isn’t anything particular in mind. So what are the available options? One option is to visit those "Top 10 movies/TV series" blogs or websites and choose one at random to watch. But the problem is that if those movies or series were really popular, you might have already seen them by now, since these lists tend to be generic. Another option could be asking your friends whom you share common interests with. They have always given you good recommendations in the past due to shared interest, so maybe they might recommend something good to you again! That is exactly where the main idea for recommender systems originated from. Basically seeking others with similar interests and formulating personal recommendations tailored to the user based on other like-minded users. Not only are recommender systems more beneficial to helping the end user find what he or she requires by giving a pleasurable browsing experience but also they have an extremely high potential for increasing sales by exposing users to items more tailored to their needs. Recommender systems are such a big concern to businesses that even Netflix once held a 1 Million dollar prize for whomever can decrease their error score metric by just 10% [8]. 1.1 Motivation In the age of unlimited digital content and due to the rise of E-commerce businesses, we are no longer restrained to the physical capacity of the shelves but to the receptive capacity of the end user. For example, while physical book stores may only be able to offer a limited number of books that only their inventory can accommodate, there are online stores such as Amazon that can offer millions of books to their customers. The explosive growth in the amount of available digital information and the number of internet users created a challenge of information overload which hinders timely access to items of interest available online. Search engines such as Google have partially solved this problem but prioritization and personalization (where a system maps available content to user’s interests and preferences) of information were absent. Fortunately, Recommender systems were designed to assist the user in consuming content of specific interest and relevance to the user; hence, recommenders solve the problem of information overload. Recommendation systems help users find and select items from the huge number available on the web or in other electronic information sources. Given a large set of items and a description of the user's needs, they present to the user a small set of the items that are well suited to the description. The recommender should suggest the items the user might be interested in. Since, if the recommender system isn’t on par with the user’s expectations then the user will most likely not pay much attention to it’s recommendations and all future recommendations will be neglected or overlooked. Which is why the quality of the recommendation the recommender system produces is vital; however, the recommendation quality is only half of the problem, the other half is the scalability of the system. The recommender system should be able to handle large amounts of data and produce recommendations on the spot for new and existing users, even if the base model consists of millions of user items. 1.2 Project Aim The focus of this project is to implement a hybrid recommender system that is based on the combination of content based filtering and collaborative filtering methods. The scalable collaborative filter is based on Singular Value Decomposition (SVD) and supports incremental updates while retaining the recommendation quality. So when new users enter the system they can be "folded in" directly and receive their recommendations instantaneously with good accuracy that is comparable to the base SVD model. Instead of originally waiting till the rebuilding of the whole model from scratch at the specified server time. In content based filtering each item is represented by a set of features or attributes in the vector space model which characterize that item and use cosine similarity to calculate the similarity scores between the items. The content filtering and collaborative filtering are combined by taking the weighted average of the content recommendation and collaborative recommendation. The main advantage of the Hybrid Recommender systems is that they combine two or more recommendation techniques to realize performance with fewer of the drawbacks of any of them. The hybrid recommendation system not only provides improved recommendation accuracy but also handles the cold start problem and sparsity of data. 1.3 Organization of Thesis This Thesis is divided in to the following chapters: Chapter 2, Literature Survey: This chapter outlines and explains the terminologies and concepts for understanding how recommenders function. Any background knowledge or concept needed to understand this thesis is provided in this chapter. Chapter 3, Methodology and Implementation: This chapter explains the approach and implementation of the hybrid recommender system . Chapter 4, Results and Evaluation: This chapter introduces the different error metrics and datasets we used for evaluation. It also contains a discussion of our achieved results. Chapter 5, Conclusion and Future Work: This chapter concludes the thesis and final discussion of the project and includes what additional features and ideas to include and examine in the next iteration of the project. Chapter 2 Literature Survey In this chapter we shall introduce the motivation, terminology and concepts needed to understand generally how recommender systems work then we will explain how recommendations are made, and we will demonstrate how classic collaborative filtering and content filtering is utilised in order to outline the difference between the classic recommendation technique and our hybrid model. 2.1 Methods of Recommendation There are multiple ways recommendations could be produced such as: 1. Hand curated content: Recommendation content could be hand curated by a human staff; however, this has the problem of putting too much load on the human workforce and might be lacking user feedback hence it won't be personalised. 2. Aggregate lists: A simple aggregate list like "top 10" or "most popular items" (like Youtube's trending/most popular videos for example); however, this suffers from being too generic and not relatable to most users, hence those with more refined tastes tend to get overlooked. 3. Recommender systems: Recommenders could generate personalised recommendations specifically tailored to the user, just like that friend from chapter 1, the system knows what content the user consumes and what content he or she prefers based on previous feedback and interactions. 2.2 Categorisation of Recommender Systems Generally, Recommender systems are categorised into three categories based on how recommendations are made: 1. Content-based filtering: Makes recommendations based on the user's past preferred item features (meta-data) for example such as text and/or item attributes (no other user ratings are required) from past highly rated items by the user, usually the advances of information retrieval are utilized here [1, 11]. 2. Collaborative -based filtering: Makes recommendations based on the user's past ratings in addition to the other user's past ratings to find items of similar interest to the user [9]. a. Memory-based: Utilizes statistical techniques to find neighbors then it computes a prediction derived from the weighted combination of the neighbor's rating [13, 11]. b. Model-based: Utilizes Data mining or machine learning algorithms to create a model and recognize patterns based on pure data to make intelligent predictions based on the learned models [16]. 3. Hybrid: Combines both of the content-based and collaborative- based approaches [1]. There are several strategies for hybrid recommenders; however, the most popular are weighted recommenders which combine the scores outputted by different recommender algorithms. Also switching recommenders which switch between recommender algorithms according to the best expected results in a particular con- text [5]. A simple example on switching recommenders is using content based filtering at the start when there are not enough users or ratings for the user but then switching to collaborative filtering when there are enough users or ratings for the user. 2.3 Utility Matrix Recommender systems have been usually studied under their most common form of "rating prediction" [11, 1] where the recommendation problem has been simplified to just predicting ratings for items not seen /rated by the user, if the ratings are high enough, then the items get recommended to the user based on those predicted ratings [1]. The corresponding rating data is represented in the form of a "utility matrix" which is basically a user- item pair containing values from an ordered set (ex: integers from a certain range representing stars given) denoting the degree of the user's preference to that item [1, 9] , it can be represented formally , Utility matrix u : users × items → ratings where users is the set of all users while items is the set of all possible items that can be recommended , and ratings are the values coming from an ordered set (values of a specific range) [1]. An example utility matrix is shown in Figure 2. 1. Figure 2. 1: The utility matrix which consists of utility matrix u : user × item → ratings where each cell rui corresponds to the rating of user u for item i, and it is required to find the rating of current user a which is rai [11] As can be seen from Figure 2.1 not all items are rated by most users hence the matrix can be called "sparse" meaning most entries are "unknown" (there are blanks in the matrix) [9, 11], which formally means the utility u is not defined on the whole users × items space but only a subset of it [1]. These "not rated" or even "unseen" items form the pure essence of the recommender system problem, where we must "extrapolate" [1] or predict the missing ratings then use the predicted ratings or utility to formulate a list of items that if the user saw them would of possibly gave them a close high rating. Thus the recommender system problem is to predict the missing ratings ;however, predicting every blank entry is not the main objective of a recommender, the main objective is finding the few highly valued items to the user and recommending them [9]. 2.3.1 Collecting data for utility matrix As can be deduced from the above section 2.3 for recommender systems to be able to recommend an item there must be a utility matrix with some initial user ratings; acquiring data for the utility is out of scope for this thesis ;however, for the sake of completeness it will briefly be mentioned here. Ratings are collected either explicitly or implicitly for example some websites explicitly ask users to give ratings, while others can create a certain formula that deduces ratings based on inferences from the user behavior for example buying a product or several products of the same type or even watching a movie for a specific duration on certain websites. [9]. 2.4 Collecting data for feature representation Using keywords to model items is an important step for many recommender systems. But extracting keywords of an item is also a difficult problem, especially in the media field, because it is very hard to extract text keywords from a video. Expert tagging systems like Pandora for music and Jinni are present for tagging the items. Let's take Jinni as an example, the researchers of Jinni defined more than 900 tags as movie gene, and they let movie experts make tags for them. These tags belong to different categories, including movie genre, plot, crew, cast, credits and keywords. Figure 2.2 shows the tags for the movies generated by the experts at MovieLens. As we can see from the figure, the tags from each movie are divided into many categories: Title, Cast, Director, Keywords and Genres. These tags contain all aspects of movie information, which can describe a movie very accurately. Figure 2.2: The metadata of the different movies which consists of tags of various categories including title, genres, crew, cast and keywords. 2.5 Content-based filtering Content based filtering is deeply based on the well established researches in the information retrieval and information filtering fields [1] since a popular method in content based filtering is using the item features as the knowledge base and the user's profile or preference as an information retrieval query which then produces a recommendation to satisfy that query [1, 11]. These methods are best suited to situations where there' s known data on an item (name, location, description, etc.), but not on the user. In this system, keywords describe the items and a user profile is generated to suggest the type of item this particular user may like. In a more generic sense, these algorithms attempt to recommend items that are similar to what a user liked in the past, or is examining in the present. Figure 2.3 Content based filtering system chooses items similar to those for which the user has already expressed a preference. Another alternative to using information retrieval for collaborative filtering is treating the problem as a classification task and using one of the popular classification algorithms in machine learning such as naive Bayes classifier, k-nearest neighbor, decision trees [11]. 2.5.1 Feature Representation In content based filtering the first step is to identify keywords for representing the items. In order to avoid indexing useless words, a text retrieval system often associates a stop list with a set of items. The irrelevant words are called stop words (the, of, for, with, etc). There is a requirement that the information retrieval system has to identify groups of words, where groups are small syntactic variants of one another and collect only the common word stem per group. A group of various words may share an equivalent word stem. For example apple and apples share a common word stem. 2.5.1.1 Vector Space Model Before being deep into feature representation, it is very important to get the idea of document representation. Assume there are several documents and each document consists of one single sentence. Then we can represent the document as a model in Figure 2.3, which is called the Vector Space Model. Consider each feature of a movie as a term, then a feature can be represented by this model. Fig 2.4 Vector Space Model of Documents 2.5.1.2 TF-IDF TF-IDF is short for term frequency-inverse document frequency that is a common weighting technique for information retrieval and text mining, which reflects how important a word is for a document [23]. The importance of a word increases proportionally to the times the word appears in the document, but it also decreases inversely proportional to the frequency the word appears in the whole corpus. TF-IDF consists of TF and IDF, which are Term Frequency and Inverse Document Frequency respectively. TF represents the frequency a word appears in the document. The main idea of IDF is: if a term appears more in other documents, the term will be less important. 2.5.1.3 Term Frequency Term frequency is about how many times a term ti appears in document dj, which can be represented by TF(tij). In the condition of removing stop-words, the more ti appears in the document, the more important term ti is for the document. It can be defined as: N (tij ,dj) T F (tij) = N (dj) (2 .1) N(ti,dj) is the number of times ti appears in dj and N(dj) is the total number of terms in document dj. 2. 5.1.4 Inverse Document Frequency In order to understand inverse document frequency, let us see what document frequency is. Document frequency is how many times term ti appears in all documents C, which is represented by N(ti , C). The more term ti appears in all documents C, the weaker term ti can represent document dj. Inverse document frequency means that the represent ability of term ti for document dj and its amount in all documents N(ti , C) is inverse proportion, which is represented by IDF(ti): IDF (ti) = log NN(t(iC,C)) (2.2) N(C) is the total amount of documents, IDF(ti) decreases with the increase of N(ti , C). The less N(ti , C) is, the more representative ti is for dj. 2.5.2 Similarity Calculating the similarity between items is the objective of content-based recommender systems, with the assumption that if a user likes a certain item like a movie, news article or a product, he/she may like similar items in future. 2.5.2.1 Cosine Similarity The usual similarity metric to compare vectors is the cosine similarity which measures the cosine of the angle between two vectors that are projected in a multi-dimensional space. Using the Euclidean dot product formula the cosine for two non-zero vectors can be derived as: A · B = ||A|| ||B|| cos (2 .3) Given two vectors of attributes, A and B, the cosine similarity, cos(θ), is represented using a dot product and magnitude as: b similarity = cos(θ) = ||AA||·||BB|| ∑ AiBi = i=1 √i∑=n1 Ai2 √i∑=n1 Bi2 (2.4) Where Ai and Bi are components of vectors A and B respectively. The range of the similarity is between -1 and 1. -1 means that the direction of the two vectors are totally opposite and 1 means they are in the same direction. The cosine similarity is 0 if the two vectors have no relationship. For text matching, it is obvious that the weights are non-negative, so the range should be 0 to 1 in our case. Here is an example for illustrating cosine similarity [24]. Given two sentences: • A: I like watching TV, but I don't like watching films. • B: I don't like watching TV and films. How can we calculate the similarity between the two sentences? The basic idea is: the more similar the words used by the two sentences are, the more similar the sentences are. First step: Word segmentation. • A: I / like / watching / TV, but / I / don't / like / watching / films. • B: I / don't / like / watching / TV / and / films. Second step: List all the words. • I, like, watching, TV, but, don't, films, and Third step: Word frequency calculation. • A: I 2, like 2, watching 2, TV 1, but 1, don't 1, films 1, and 0. • B: I 1, like 1, watching 1, TV 1, but 0, don't 1, films 1, and 1. Fourth step: Get word frequency vector. • A: [2, 2, 2, 1, 1, 1, 1, 0] • B: [1, 1, 1, 1, 0, 1, 1, 1] Then we can calculate the cosine of the two vectors by Equation 3.15. cos() = 2×1+2×1+2×1+1×1+1×0+1×1+1×1+0×1 √22+22 +22+12+12+12+12+02×√12+12+12 +12+02+12+12+12 The value is 0.85 so that the two sentences are much similar. 2.6 Collaborative-based filtering Collaborative filtering deduces recommendations based on the user's past ratings and other like minded user's past ratings to find items of similar interest to the user [9, 13]. As stated before, Collaborative filtering is divided into Memory -based which utilize statistical techniques for deducing neighborhoods which are used for producing recom- mendations [13, 11] and Model-based which utilize data mining and/ or machine learning techniques to create the models which are used for producing recommendations [16]. Figure 2.5 Content based filtering system chooses items similar to those for which the user has already expressed a preference. 2.6.1 Memory-based: K-Nearest Neighbor Collaborative Filtering (k-NN) The most traditional type of memory based collaborative filtering is K-Nearest Neighbor; It was the first automated collaborative filtering method and forms the basis to current collaborative filtering recommender systems [5]. Additionally it is also known as "user- user collaborative filtering" [5] , at its base definition it is simply finding a subset of other users whose past rating behavior is similar to the current user and then use those weighted (based on similarity to the current user) ratings to predict new items that the current user will value [5, 11] . 2.6.1.1 Calculating Predictions In order to predict ratings we first need to compute the similarity using a certain similarity function (explained in the next subsection 2.6.1.2) to know how similar certain users are and to be able to gather a neighborhood N ⊆ U where U is the space containing all of our users. Then the predicted rating simply becomes a "weighted average of the neighboring users' ratings which uses the similarity as weights" [5] as formulated in the following: pu ,i = ru + v∈N ∑ simi (u,v)(rv,i − rv) (2.5) ∑ |simi(u,v)| v∈N Where N is the neighbors of user u (subset of the user base which are from the neigh- borhood of u) and v is the other user which is neighbor to our current user u, simi(u,v) is the "similarity function" which computes how similar current user u is with user v which adds the weights, i is the current required item's rating , pu ,i is the current user' s predicted rating for item i[5]. Notice that the average user rating rv gets subtracted from rv,i to somewhat normalize the user's rating bias since a user could be a hard or easy rater , also the ru is being added for the same reason, to compensate the user's rating bias so the predicted rating is close to the real rating [5, 9]. 2.6.1.2 Computing Similarity To be able to compute predictions we first need to determine how similar other users are to our current user , hence we need a "similarity function" to know how similar another user's "taste" is to our current user's , we can use one of several similarity functions which is either based on statistics or linear algebra. • Pearson correlation: Computes the statistical correlation (Pearson's coefficient) the value is between [-1,1] inclusive where -1 is a negative correlation (opposing similarity therefore not similar) , 0 no correlation (no similarity) and 1 is a positive correlation (strong similarity) ∑ (ru,i − ru)(rv,i − rv) simi(u, v) = i∈Iu∩ Iv (2. 6) √ ∑ i∈ Iu ∩ Iv (ru,i − ru)2 √ ∑ i∈ Iu ∩ Iv (rv,i − rv)2 Where item i is the commonly rated item between user u and user v as denoted formally by i ∈ Iu ∩ Iv [5,11] • Cosine similarity: Measures the cosine distance (The vector's closeness (angle)) [5, 9] since the cosine similarity is based on the cosine's angle therefore the cosine can only take values from [-1,1] inclusive where -1 means the vectors are exactly opposite in direction , 0 means vectors are orthogonal to each other , and 1 means the vectors are exactly same direction. simi(u, v) = cos(u, v) = cos(θu,v) = ||rruu|·|||rrvv|| ∑ ru,i rv,i = i √∑i ru,i2√∑i rv,i2 (2.7) From the below figure 2.6 it can be shown that users can be represented as vectors in a two dimensional space where each axis represents an item. Figure 2.6: user u and user v represented as vectors in a 2d item space with angle θ in between which signifies the cosine distance between those two vectors since the angle translates to similarity (cosine similarity) 2.6.1.3 Computing a movie rating using k-NN Given user × movie matrix in figure 2.6 we want to find User C’s predicted rating for movie "Equilibrium" (pc,e). We will be using Pearson correlation as our similarity function and a neighborhood of size 2. Fig 2.7 An Example user x movie matrix on a 5 point rating scale [5] The mean rating of user C is rc = 3.667, only two users have rated "Equilibrium" therefore only two candidates for the neighborhood even though some might not be similar to our user but to meet the neighborhood quota we must take them into consideration. simi (c,a)= 0.832 and simi (c,d)=-0.515 as can be seen d isn’t similar to user c but it will be taken into consideration to meet the neighborhood quota ;however, it’s weights will reflect it’s dissimilarity to user c, to get pc,e we can just apply equation 2.1 [5] pc,e = rc + simi(c,a)|(sriam,ei(−c,raa))| ++ s|siimmii((cc,,dd))(|rd,e − rd) (2.8) = 3.667 + 0.832 · (|05.−8342)| ++ −|00.5.51155| · (2−3) = 4.667 2.6.2 Model-based: Singular Value Decomposition (SVD) Singular Value Decomposition is a dimensionality reduction technique with deep roots and origins in information retrieval where it is usually called latent semantic indexing (LSI) ,this information retrieval technique was patented by Deerwester et al. [4, 12, 16]. It was mainly used to solve the problems of synonymy which is having several words containing the same meaning and polysemy which is having words containing more than one meaning [12, 4]. 2.6.2.1 SVD Equation Definition SVD is a factorization technique that factors a m × n matrix into three matrices that retain the essence of the original matrix as shown in the following equation: Amn = U mr Σrr V Trn (2.9) where A is the original utility matrix, UTU = I and VTV = I which means both V and U have to be orthogonal matrices and Σ is a diagonal matrix containing the square roots of Eigenvalues from U or V in descending order [2] . So for clearer visualization of the U,V, Σ formalization: U = Orthonormal{EigenVectors(AAT)}, V = Orthonormal{EigenVectors(ATA)}, (2.10) Σ = Diagonal{Descending < Sqrt[EigenValues(AAT OR ATA)] >} The main idea of SVD is that we can represent the principle factors (dimensions) of a matrix using only a couple of the dimensions while the rest is just a linear combination of those principal axes.Which is why after getting UΣVT we truncate the dimensions to at least the rank of the matrix; however, we can even go further and reduce it to the most significant k factors which would be less than the rank of the matrix as shown in figure 2.7 and table 2.1. So Ak ≈ UmkΣkkVTkn is the best rank-K approximation to the original kn matrix A since Ak minimizes the "Frobinius norm" which is the measure of error of how close the lower rank matrix Ak is to the real matrix A [5, 10], where the Frobinius norm would be mathematically represented as |F| = Σij(Aij − Akij)2 [10] where F is the Frobinius norm and A the original matrix and Ak the resulting approximation of A that occurs from multiplying √ the reduced UkΣkVk. Which basically means A ≈ AK, where matrix Ak is the best rank-k approximation for matrix A [5]. If the k is chosen correctly, this new reduced dimensional space will show the true essence of the data without any noise [12, 10, 2, 3]. Ak Best rank-k approximation to matrix A U User vectors Σ Singular values (latent factors) V Item vectors m Number of users n Number of items k Number of factors (reduced factors) r Rank of matrix A Table 2.1: Interpretation of SVD components in recommender setting if the original matrix A contained the rows as users and columns as items. Figure 2.8: Shows matrix Ak being broken down into three matrices UmkΣkkVTkn , notice that matrix U contains the users vector as a row and matrix V contains item vector as columns while Σ contains the "concept strength" as shown in the above table 2.1 [3] 2.6.2.2 Calculating Predictions using SVD A user’s item prediction can be computed by applying the following algorithm formulated in Sarwar et al.’s research paper [12]: 1. For any empty entries, fill them in using the item average rating (Sarwar et al. stated that item average performed better than user average) , Since SVD is a mathematical operation so as with any mathematical operation , all values must be present , if there are missing values then SVD cannot be computed therefore in step1 we filled in with the product average rating this method is called "imputation" [8, 5]. 2. Normalize by subtracting user Average from each rating (Sarwar et. al states nor- malizing by this method works better than using the Z-score) since some user rate higher/lower than others. 3. We can now factor the matrix into UΣV using SVD then reduce the matrices’ dimensions to k. 4. Compute and let Q = Uk√Σk and Z = √ΣkVTk (Note that in other references such as [5], doing Q = Uk and Z = ΣkVkT worked as well.) 5. To compute prediction of the uth user for the ith item, just do the dot product to Qu,_ and Z_,i which is the uth row in Q and the ith column in Z then add back the average user rating, more formally denoted by this equation: pu,i = ru + (Qu,− · Z−,i) (2.11) Notice that what occurred in the above equation is the same as doing P = UkΣkVkT then the predicted rating pu,i would be just P u,i + ru 2.6.2.3 Intuition on why SVD works Intuitively what SVD does is that instead of imposing the different categories or latent factors of the movies like action, comedy, drama, oriented towards males, oriented towards females, serious , escapist ...etc. It deduces these categories or latent/hidden factors using the ratings patterns given by the users for the movies [8]. Basically it can be considered as a data driven categorization method for both the movies and users, the secret lies in the singular matrix Σ, which contains the weights of the latent factors [10]. Each movie and user is a linear combination of the basis unit vectors where these basis unit vectors are the categories. But these categories must be different or independent from each other thus they are orthonormal meaning they are vectors of unit length and their dot product equals zero which means they are perpendicular and independent (since they are used for being bases for linear combination). Figure 2.9 illustrates a simplified categorization that occurs when multiplying the reduced dimension user matrix Uk with Σk then graphing it and reduced dimension movie matrix Vk with Σk then graphing it [3]. Also figure 2.10 shows the clustering that occurs on each latent factor vector which acts as the base axis for the movies and users (can only visually show a 2D plot with 2 factor vectors as the base axis). Figure 2. 9: A simplified illustration of the latent factors, which characterizes both users and movies on some deduced latent factors [8]. Figure 2.10: "The first two vectors from a matrix decomposition of the Netflix Prize data. Selected movies are placed at the appropriate spot based on their factor vectors in two dimensions. The plot reveals distinct genres, including clusters of movies with strong female leads, fraternity humor, and quirky independent films." [8]. 2.6.2.4 Interpretation example on SVD Given the simple user × movie matrix in figure 2.11 from source [10], we want to use the information from subsection 2.6.2.3 to intuitively understand the components of svd and the latent factors. Figure 2.11: user × movie → rating utility matrix M [10]. After doing SVD via numpy we arrive at figure 2.12, in this contrived example the matrix is of rank=2 (number of linearly independent vectors) so the k=2 which means this is the full SVD formulation so U ΣVT effectively equals matrix M and doesn't approximate it in this instance. U contains the "user-to-concept" (user to latent factor) matrix, V contains the "movie-to-concept" (movie to latent factor) matrix (so VT is "concept-to-movie"), and Σ is the concept strength (latent factor scale or weight) [10]. Figure 2.12: Full SVD for matrix M [10]. From figure 2.11 and Σ it's easy to see that there are two concepts or latent factors present, the "Sci-fi" factor and the "Romance" factor, this can be deduced by looking at the columns of VT since we know that Matrix, Alien, Starwars are scifi and we can see that they are not zeroes in the 1st row (dimension) but are zeroes in the 2nd row so their scifi factor weight corresponds to 12.4. While Casablanca and Titanic are non zero on the other row (dimension) so their romance factor weight is 9.5. For users, using the same thought process as above, it can be seen that column 1 in U is the scifi latent factor dimension and column 2 is the romance latent factor dimension, if we multiply U by Σ (user-to-concept multiplied by concept weight) we project the users on the latent factor axis and could plot them, it can be seen where each user falls into place and on which latent factor vector such as shown in figure 2.13. Notice that Joe and Jane rated their movies low so they had small factors in U in their respective rows, this is also reflected in the graph. Figure 2.13: Graph of U multiplied by Σ which produced projection of users on the two latent factor vectors which act as axes. 2.7 Hybrid Filtering To address the shortcomings of both content-based and collaborative-based approaches, often recommendations are made using a combination of the two techniques. There is a large variability on the hybrid methods – there is no standard hybrid method. Table 2.2: Some of the common hybrid filtering methods. 2.7.1 Weighted Hybrid A simple approach for building hybrid systems is using weights: SA (p) is the predicted rating for product p computed by algorithm A SB (p) is the predicted rating for product p computed by algorithm B The hybrid rating can be calculated as follows: SH (p) = ɑ SA (p) + (1- ɑ)SB (p) (2.12) The score of a recommended item is computed from the results of all of the available recommendation techniques present in the system Example 1: a linear combination of recommendation scores n Example 2 – many recommender systems: treats the output of each recommender (collaborative, content based and demographic) as a set of votes, which are then combined in a consensus scheme. The implicit assumption in this technique is that the relative value of the different techniques is more or less uniform across the space of possible items. 2.7.1.1 Weighted Example This is an example from the movie recommendation system that integrates item-to-item collaborative filtering and information retrieval [25]. Information retrieval component: W eb (i, q) = (N + 1 − ki) N (2.13) where N are the items returned by the query q and ki is the position of movie i in the results set (example q = "arnold schwarzenegger"). Movies highly ranked by the IR component (low ki) have a Web(i,q) value close to 1. Item-to-item collaborative filtering: Auth(i, u) is the score of item i for user u. Movies similar to those highly ranked by the user in the past get a high Auth(i, u) score. Final rank: MADRank(i,q,u)= ɑ Auth(i,u) +(1- ɑ)Web(i,q) (2.14) 2.7.2 Switching In a switching recommendation system, some criterion is used to switch between recommendation techniques. If the content-based system cannot make a recommendation with sufficient confidence, then a collaborative recommendation is attempted. Example: The DailyLearner [26] system uses a content/collaborative hybrid in which a content-based recommendation method is employed first This switching hybrid does not completely avoid the ramp-up problem, since both the collaborative and the content-based systems have the "new user" problem. 2.7.3 Mixed In this method, recommendations from more than one technique are presented together. The mixed hybrid avoids the "new item" start-up problem: n since the content-based approach can be used for new items although it does not get around the "new user" start-up problem: both the content and collaborative methods need some data about user preferences to start up. It is a good idea for hybridizing two different kinds of recommender (e.g. demographic and collaborative) and most importantly it introduces diversity in the recommendation list. 2.7.4 Cascade In cascade hybrid recommendation technique, one recommendation technique is employed first to produce a coarse ranking of candidates and a second technique refines the recommendation from among the candidate set. Example: EntreeC uses its knowledge of restaurants to make recommendations based on the user's stated interests: The recommendations are placed in buckets of equal preference (equal utility) and the collaborative technique is employed to break ties. Cascading allows the system to avoid employing the second, lower-priority, technique on items that are already well differentiated by the first but it requires a meaningful and constant ordering of the techniques. 2.7.5 Feature Combination The Feature combination method of hybrid filtering achieves the content/collaborative merger treating collaborative information (ratings of users) as simply additional feature data associated with each example and use content-based techniques over this augmented data set. Example: The inductive rule learner Ripper can be utilised for the task of recommending movies using both users' ratings and content features [27]. The feature combination hybrid lets the system consider collaborative data without relying on it exclusively, so it reduces the sensitivity of the system to the number of users who have rated an item p The system has information about the inherent similarity of items that are otherwise opaque to a collaborative system. Figure 2.14: Feature Augmentation Chapter 3 Methodology and Implementation In this chapter we shall explain our approach and implementation of the hybrid recommender system based on the principle mentioned in Chapter 2 using Incremental SVD technique that updates the SVD model with new users instead of building from scratch and popularity based content filtering that models the movies into the vector space and calculates similarity using cosine similarity to recommend critically acclaimed movies of user's preference. 3.1 Dataset All the movie data we used is from The Movies Dataset by Kaggle. The dataset consists of movies released on or before July 2017 with metadata for all 45,000 movies listed in the Full MovieLens Dataset. Data points include cast, crew, plot keywords, budget, revenue, posters, release dates, languages, production companies, countries, TMDB vote counts and vote averages. For the perspective of the recommender system, a movie can be described by a collection of features, which can be genres, actors, directors and so on. • Crew: The directors, producers, editors etc. For our purpose, we will be considering the directors only. Most of the movies only have one director, but some of them have two or more. • Actor: A movie normally has a lot of actors, but most of them are useless for the recommender system and bring disadvantageous effects. So we only get three main actors for one movie. • Keywords: Contains the movie plot keywords for our movies. Available in the form of a stringified JSON Object. • Genres: Theme of the movie that describes movies in a different perspective than keywords. Eg. Sci-Fi, Romance, Thriller, Horror etc. • Ratings: The set of ratings from different users to the movies. 3.2 Collaborative Filtering To address some of the limitations of content-based filtering, collaborative filtering uses similarities between users and items simultaneously to provide recommendations. This allows for serendipitous recommendations; that is, collaborative filtering models can recommend an item to user A based on the interests of a similar user B. Furthermore, the embeddings can be learned automatically, without relying on hand-engineering of features. We have created a 2D matrix using the given data which contains the explicit users' feedback which specifies how much they liked a particular movie by providing a numerical rating (in our case a number between 0 and 5). The collaborative nature of this approach is apparent when the model learns the embeddings. Suppose the embedding vectors for the movies are fixed. Then, the model can learn an embedding vector for the users to best explain their preferences. Consequently, embeddings of users with similar preferences will be close together. Similarly, if the embeddings for the users are fixed, then we can learn movie embeddings to best explain the feedback matrix. As a result, embeddings of movies liked by similar users will be close in the embedding space. 3.2.1 Matrix Factorisation To create these embeddings we have used Matrix Factorisation. Matrix factorization is a simple embedding model. Given the feedback matrix A∈ Rm×n, where m is the number of users (or queries) and n is the number of items, the model learns: • A user embedding matrix P ∈ Rm ×d, where row i is the embedding for user i. • An item embedding matrix Q ∊ Rn×d, where row j is the embedding for item j. The embeddings are learned such that the product PQT is a good approximation of the feedback matrix A. The (i,j) entry of P.QT is simply the dot product ⟨Ui,Vj⟩ of the embeddings of user i and item j, which we want to be close to Ai,j. The prediction r ui is set as: 𝑝 𝑟̂j = 𝜆 +𝑏i +𝑏u + 𝑝𝑇i 𝑝u (3.1) Where, 𝜆 is the global mean, bi is the item bias, bu is the user bias, q is the item matrix, p is the user matrix. To estimate all the unknown, we minimize the following regularized squared error: ∑ (𝑝𝑢𝑖−𝑝 𝑢̂𝑖2 + 𝜆 (𝑏i2 + 𝑏u2 + || 𝑝𝑖 ||2+|| 𝑝𝑢 ||2) (3.2) ui ∊ train The minimization of this error function is performed by a very straightforward stochastic gradient descent. 3.2.2 Folding in a new user When a new user arrives his user embedding has very few ratings. To recommend movies to the new user we cannot compute the matrix factorization all over again since it is an expensive computation. We use the error function mentioned above and the gradient descent algorithm. But this time we keep the item matrix fixed and try predicting values only for the new user embedding instead of the complete user matrix, thereby reducing computation drastically. 𝑏𝑢 ←𝑏𝑖 + 𝛾 (𝑒𝑢𝑖 −𝜆 𝑏𝑢) pu ←pi + 𝛾 (𝑒𝑢𝑖 −𝜆 pu) 3.2.3 The Algorithm (3.3) (3.4) Figure 3.1 shows the architectural flow for our implementation. Below are the steps taken by our algorithm. 1. Load the data set using Pandas into dataframes. 2. We convert the ratings dataframe into a utility matrix where the rows are the user ids and the columns are the movie ids. 3. Then we perform the matrix factorisation which converts our utility matrix to 2 matrices p and q where p contains the user embeddings and q contains the item embeddings. 4. The matrix factorisation is performed considering the error function: ∑ (𝑝𝑢𝑖−𝑝 𝑢̂𝑖2 + 𝜆 (𝑏i2 + 𝑏u2 + || 𝑝𝑖 ||2+|| 𝑝𝑢 ||2) (3.2) ui ∊ train 5. To get the estimated rating for a user u for a movie i we simply take the dot product of the embeddings. The prediction r ui is set as: 𝑝 𝑟̂j = 𝜆 +𝑏i +𝑏u + 𝑝𝑇i 𝑝𝑢 (3.1) Where, 𝜆 is the global mean, bi is the item bias, bu is the user bias, q is the item matrix, p is the user matrix. Figure 3.1: Architectural Flow diagram to algorithm to find the estimated rating of a movie by a user. 3.3 Content Filtering The standard recommendation systems built purely on movie popularity and genre end up recommending movies that are more popular and more critically acclaimed. This model does not give personalized recommendations based on the user. For example, this is the list of Top 15 Romance movies according to the popularity rating (wr). Figure 3.2: Similar movie recommendations that don't take personalisation into account Consider a person who loves Dilwale Dulhania Le Jayenge, My Name is Khan and Kabhi Khushi Kabhi Gham. One inference we can obtain is that the person loves the actor Shahrukh Khan and the director Karan Johar. Even if s/he were to access the generic romance chart like Fig 3.x, s/he wouldn't find these as the top recommendations. 3.3.1 Our Approach: Extending Content Filtering with Popularity As discussed in Section 2.5 .1, a document can be represented as a set of features in the vector space model. Each movie in our case is a document, represented by the Vector Space Model and each feature for the movie is a term in the document. Then we use cosine similarity discussed in Section 2.5.2 to calculate similarity for each movie. To personalise the recommendations more, we suggest movies that are most similar to a particular movie that a user liked but still takes into consideration the popularity of the movies so that bad movies are not recommended to any user based on his/her taste. 3.3.2 The Algorithm Figure 3.4 shows the architectural flow for our implementation. Below are the steps taken by our algorithm. 6. Load the data set using Pandas into dataframes. 7. To pre-process the keywords, calculate the frequency counts of every keyword that appears in the dataset. Keywords that occur only once are not used and can be safely removed. 8. Convert every word to its stem using SnowballStemmer so that words such as Dogs and Dog are considered the same. 9. Merge movie's metadata(name,id) dataframe with the crew, cast and the keywords dataframe information thus creating a metadata dump or soup for every movie. - Lesser known actors and minor roles do not really affect people's opinion of a movie. Therefore the top 3 actors that appear in the credits list are chosen - From the crew, only the director is picked as the feature since the others don't contribute that much to the feel of the movie. For example, consider the movie 'Toy Story'. Table 3.1 shows the metadata information that will be merged to form a dump shown in Fig 3.3 Director(Crew) John Lasseter Cast Tom Hanks, Tim Allen, Don Rickles Keywords jealousy, toy, boy, friendship, friend, rivalry, boynextdoor, newtoy, toycomestolife Genres Animation, Comedy, Family Table 3.1 Information of Toy Story ['jealousy toy boy friendship friend rivalry boynextdoor newtoy toycomestolife tomhanks timallen donrickles johnlasseter Animation Comedy Family'] Figure 3.3 The representation of Toy Story 10.To convert a movie to the vector space model, use sklearn's CountVectorizer to convert this combined metadata soup of all the movies into a matrix of token counts. 11 .Compute the cosine similarity between a given movie and all the movies represented in the count matrix from Step 4 using sklearn's linear_kernel. This returns similar movies regardless of ratings and popularity. For example, it is true that Batman and Robin has a lot of similar characters as compared to The Dark Knight but it was a terrible movie that shouldn't be recommended to anyone. 12.To ensure popular movies with good critical response are returned, take the top 25 movies based on similarity scores from 6 and use IMDB's weighted rating formula: Weighted Rating (WR) = (v+ vm · R) + (vm+m · C) (3.5) where, • v is the number of votes for the movie • m is the minimum votes required to be listed in the chart • R is the average rating of the movie • C is the mean vote across the whole report to construct a chart of the most popular movies from those 25 movies. 13.To determine an appropriate value for m, the minimum votes required to be listed in the chart, use 75th percentile as our cutoff. In other words, for a movie to feature in the charts, it must have more votes than at least 75% of the movies in the list. Figure 3.4: Architectural Flow diagram to algorithm to find the top 'n' popular movies that are similar to the given movie. 3.4 Hybrid Filtering As mentioned in chapter 2 Collaborative recommenders rely on data generated by users as they interact with items. In the context of a movie recommender, collaborative filters find trends in how similar users rate movies based on rating profiles. There are some drawbacks to collaborative filters such as the well-known cold start problem. It is also difficult for collaborative filters to accurately recommend novel or niche items because these items typically do not have enough user-item interaction data. Content filters tend to be more robust against popularity bias and the cold start problem. They can easily recommend new or novel items based on niche tastes. However, in an item-to-item recommender, content filters can only recommend items with features similar to the original item. This limits the scope of recommendations, and can also result in surfacing items with low ratings. In the context of a movie recommender, a collaborative filter answers the question: "What movies have a similar user- rating profile?", and a content filter answers the question: "What movies have similar features?" Hence we have created a hybrid recommender that recommends movies that other users rated in a similar manner, while still making on-topic recommendations based on the features of that movie. 3.4.1 Combining Recommendations An approach to combine collaborative and content-based filtering is to make predictions based on a weighted average of the content-based recommendation and the collaborative recommendation. The rank of each item being recommended could be a measure for the weight. In this way the highest recommendation receives the highest weights. 3.4.2 The Algorithm 1. We combine the outputs of collaborative and content filtering to produce recommendations. 2. Given a user u for whom we need to compute the recommendations first we get 'n' recommendations from the collaborative filtering. 3. Using these 'n' movies we get 'm' movies from content filtering. 4. Now for these 'n+m' movies we calculate a hybrid score for all these movies using the formula : α x Collaborative Score + β x content score (wr) Where, β = 1 - α Weighted Rating (wr) = (v+vm · R) + (v+mm · C) where, v is the number of votes for the movie m is the minimum votes required to be listed in the chart R is the average rating of the movie C is the mean vote across the whole report (3.6) (3.5) 5. We sort the recommendations on the basis of this hybrid score and recommend movies. 3.5 Demonstration and visual validation Chapter 4 Results and Evaluation In this chapter, we shall discuss our achieved results when running our algorithm on MovieLens dataset. We will also analyze the distribution of the datasets and the optimal number of latent dimensions. Also we will initially introduce the class of evaluation metrics we have chosen for evaluating our recommender system. 4.1 Evaluation Metrics In order to claim that a recommender is better than another recommender system, one needs a standard evaluation metric which can be the same for all recommender systems. Since different datasets can give different accuracy readings for the same algorithm [5]. There are a lot of factors that can affect the error score of a recommender, possible examples can include: 1. Sparsity: How empty is the dataset? Some recommenders can work well with dense data while others perform poorly on sparse data, and vice versa. 2. Rating scale: Is the dataset on a scale from 1-5 or 1-10? An error score in one scale could be good while in another scale it could be bad. 3. Portion used for testing: Which portion of the dataset was used for training and which for testing? The parts of the data used must be consistent for all when evaluating a recommender. This is usually true in competition like the Netflix challenge. 4.1.1 Predictive Accuracy Metrics The class of metrics we decided to use for evaluation are the predictive accuracy metrics. These metrics measure how accurate or close the predicted ratings are to the real user ratings [6, 7]. They were chosen due to their consistent use in academic papers and their inclusion in the Netflix competition, thus proving relevancy. 4.1.1.1 Mean Absolute Error (MAE) The MAE measures the average deviation between the predicted and real user rating [7] Σ| pu,i − ru,i| MAE = u,i n (4.1) Where n is the number of elements held out for testing and pu,i is the predicted rating for the real hidden user ratings ru ,i. So if MAE=0.75 then that means the algorithm was off by 0.75 stars on average therefore less is better [5]. 4.1.1.2 Root Mean Square Error (RMSE) The RMSE is similar to MAE; however, it places more emphasis on large errors. For example a system is penalized more if it's off by 2 stars in 1 single prediction, than for being off 1/4 of a star in 8 predictions [5]. Therefore less is also better here. RMSE has been popularized by the Netflix prize since it was the chosen error metric to be decreased by just 10% for the 1 million dollar prize [5, 6]. RMS = √ Σ(pu,i − ru,i) 2 u,i n (4.2) Where n is the number of elements held out for testing and pu,i is the predicted rating for the real hidden user ratings ru ,i. 4.2 MovieLens Dataset The dataset we have tested our recommender system on was The Movies Dataset on Kaggle1. Dataset contains “45 ,000 ratings from 700 users on all the 45,000 movies”. The rating scale for this dataset is from 1 to 5. Data points include cast, crew, plot keywords, budget, revenue, posters, release dates, languages, production companies, countries, TMDB vote counts and vote averages. In the upcoming subsections we will discuss the user distribution, finding optimal weights given to each method for calculating hybrid score, and the achieved results. 4.2.1 User Distribution We examined the user distribution to get a clearer image of the data. So we implemented a simple algorithm in python that counts the number of rated items that each user had rated. Then it counted the frequency of the number of occurrences of each rating per user count. Then it proceeded to plot a histogram for visualization. Figure 4.x shows the distribution of the number of ratings per user count. This is important because it means that the dataset has different types of users, i.e, the users which rated only a few movies and the ones who rated a lot. Figure 4.1 : Distribution of number of ratings to the count of users 4.2.2 Evaluating our Hybrid model versus the standard model - our model vs collaborative alone, our model vs content alone The RMSE value calculated for our hybrid model and the standard collaborative and content model alone are summarized in the table 4.x. It is observed that the error value is least for the hybrid model. Algorithm RMSE MAE Collaborative Filtering 1.0123370322183025 0.7452834149039007 Content Filtering 1.2962940839324248 1.0821815179168959 Hybrid Filtering 0.9938265116368672 0.7452834149039007 Table 4.1 : Summary of Results 4.2.2.1 Division of data into Training and Testing datasets We divided the dataset into training and testing. The training set is the data that the SVD will be built from, which is used for the collaborative filtering and the testing test is used to calculate the root mean square error. The rating dataset is divided such that 75% of it is used as training dataset and the remaining 25% is used as testing dataset. 4.2.2.2 Calculating the RMSE value for the testing dataset The training data of rating is used to train the collaborative model so as to predict the rating for the given movie and user and content model use movie data to calculate the rating. The RMSE is calculated for the testing data, i.e, 25% of the total ratings data. For each movie rated by a user in a testing dataset, a rating is predicted by our hybrid by taking a weighted average of values from collaborative and content models. RMSE value is calculated for all predicted ratings against the actual rating in the dataset. 4.2.2.3 Finding optimal value of alpha The rating of a movie predicted by the hybrid model is equal to the weighted average of rating predicted by the collaborative model and the content model. The weight given to the rating predicted by the collaborative model is taken as alpha and the rating predicted by the content model is taken as (1 - alpha). In order to calculate the optimal value of alpha to get the least RMSE value, its value is varied from 0 to 1 and corresponding RMSE value is calculated for the testing dataset as shown in Figure 4. x. We get the minimum RMSE value for alpha as 0. 81. Figure 4.2 : Optimal Value of alpha for least Root Mean Square Error 4.3 Discussion In this section we shall discuss our findings for the optimal alpha and Error scores in the dataset. 4.3.1 The Optimal alpha As discussed in section 4.2.2.3, we calculated the RMSE value by varying alpha. It is found that for the value of alpha equal to 0.81, RMSE value is the least. This implies that the score from the collaborative model has relatively more weightage than the content model in the hybrid model. This optimal alpha value may vary depending on the item for which the rating is calculated. Here the item considered is a movie which has attributes such as crew, cast, genre, etc which are used for calculating the similarity in movies. But for some other items (say food products) for which a similar recommendation model is to be developed , the attributes will vary and will not be the same as that for movies, hence the optimal value for alpha will be different. 4.3.2 Discussing and Validating RMSE and MAE The RMSE and MAE values calculated for our model are summarized in section 4.2.2. The RMSE of the hybrid model is 0.994 this means that on average, it’s value is deviated from actual value by 0.994 stars, for a 1-5 scale and the MAE of the hybrid model is 0.745 this means that on average, it’s wrong by a value of 0.745 stars, for a 1-5 scale. The RMSE value is less for the hybrid model than for the collaborative and content model individually. Hence by combining the results from both, we get better results. Chapter 5 Conclusion and Future Work 5.1 Conclusion In this project, our main directive was to implement a hybrid recommender system that leverages both content and collaborative data using an incremental SVD based algorithm that could incrementally update itself and a popularity based content filtering algorithm that personnalises the recommendations removing bad movies from the result. The incremental SVD approach would solve the scalability problem of frequent model construction in a short time frame to remain relevant and enable new users to receive recommendations instantly which is usually not the case with the base SVD model recommender, since new users will need to wait for the scheduled rebuilding of the model to receive their recommendations. The popularity based Content Filtering approach would suggest movies that are most similar to a particular movie that a user liked but still takes into consideration the popularity of the movies so that bad movies are not recommended to any user based on his/her taste. 5.2 Future Work We have planned a continued development, adding additional features to this project. The new proposed features for continued development are as follows: 5.2.2 Parallelisation 5.2.1 Development of a working web application 5.2.2 Support Folding in the new user as well as a new item simultaneously When folding in new users and the user had rated a new item not originally in the matrix. We made the algorithm truncate the new items and Fold-in the user without those new items. Since those items should either be in the model, or be Folded-in initially before Folding-in a new user with a new item at the same time. This is not a serious issue; however, it does lead to some irritations in testing when the testing set contains new items that weren't folded in or part of the SVD model. Appendix Appendix A Lists SVD MAE RMSE Singular Value Decomposition Mean Absolute Error Root Mean Square Error References [1] Gediminas Adomavicius and Alexander Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. IEEE transactions on knowledge and data engineering, 17(6):734–749, 2005. [2] Kirk Baker. Singular value decomposition tutorial. Ohio State University, 2005. [3] Michael W Berry, Susan T Dumais, and Gavin W O'Brien. Using linear algebra for intelligent information retrieval. SIAM review, 37(4):573–595, 1995. [4] Scott Deerwester, Susan T Dumais, George W Furnas, Thomas K Landauer, and Richard Harshman. Indexing by latent semantic analysis. Journal of the American society for information science, 41(6):391, 1990. [5] Michael D Ekstrand, John T Riedl, Joseph A Konstan, et al. Collaborative filtering recommender systems. Foundations and Trends® in Human–Computer Interaction, 4(2):81–173, 2011. [6] Asela Gunawardana and Guy Shani. A survey of accuracy evaluation metrics of recommendation tasks. Journal of Machine Learning Research, 10(Dec):2935–2962, 2009. [7] Jonathan L Herlocker, Joseph A Konstan, Loren G Terveen, and John T Riedl. Evaluating collaborative filtering recommender systems. ACM Transactions on In- formation Systems (TOIS), 22(1):5–53, 2004. [8] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. Computer, 42(8), 2009. [9] Jure Leskovec, Anand Rajaraman, and Jeffrey David Ullman. Mining of massive datasets, chapter 9: Recommendation Systems, pages 307–341. Cambridge Univer- sity Press, 2014. [10] Jure Leskovec, Anand Rajaraman, and Jeffrey David Ullman. Mining of massive datasets, chapter 11: Dimensionality Reduction, pages 405–437. Cambridge Univer- sity Press, 2014. [11] Prem Melville and Vikas Sindhwani. Recommender systems. In Encyclopedia of machine learning, pages 829–838. Springer, 2011. [12] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Application of dimensionality reduction in recommender system-a case study. Technical report, DTIC Document, 2000. [13] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Item-based collab- orative filtering recommendation algorithms. In Proceedings of the 10th international conference on World Wide Web, pages 285–295. ACM, 2001. [14] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Incremental sin- gular value decomposition algorithms for highly scalable recommender systems. In Fifth International Conference on Computer and Information Science, pages 27–28. Citeseer, 2002. [15] Rob Speer, Kenneth Arnold, and Catherine Havasi. Divisi: Learning from semantic networks and sparse svd. Python in Science Conf.(SCIPY 2010), 2010. [16] Xiaoyuan Su and Taghi M Khoshgoftaar. A survey of collaborative filtering tech- niques. Advances in artificial intelligence, 2009:4, 2009. [17] Ricardo Baeza-Yates, Berthier Ribeiro-Neto, et al. Modern information retrieval, volume 463. ACM press New York, 1999. [18] G Salton, A Wong and C.S. Yang. A Vector Space Model for Automatic Indexing [19] R.D.Simon, X. Tengke, and W. Shengrui, "Combining collaborative filtering and clustering for implicit recommender system," in Proc. 2013 IEEE 27th Int'l Conf. on. Advanced Information Networking and Applications, pp. 748-755, March 2013. [20] Z.Zheng, H. Ma, M. R. Lyu, et al., "QoS-aware Web service recommendation by collaborative filtering," IEEE Trans. on Services Computing, vol. 4, no. 2, pp. 140-152, February 2011. [21] Rocchio, J.: Relevance Feedback in Information Retrieval. In: G. Salton (ed.). The SMART System: Experiments in Automatic Document Processing. NJ: Prentice Hall (1971) 313- 323 [22] Salton, G. Automatic Text Processing. Addison-Wesley (1989) [23] Juan Ramos. Using tf-idf to determine word relevance in document queries. In Proceedings of the first instructional conference on machine learning, 2003. [24] Anna Huang. Similarity measures for text document clustering. In Proceedings of the sixth new zealand computer science research student conference (NZCSRSC2008), Christchurch, New Zealand, pages 49–56, 2008. [25] Chang Seok

Park, Byeong Man Kim, Qing Li, Si Gwan Kim. A new approach for combining content-based and collaborative filters. [26] Daniel Billsus and Michael J. Pazzaniu. User Modeling for Adaptive News Access. [27] Chumki Basu, Haym Hirsh, William Cohen. Recommendation as Classification: Using Social and Content-Based Information in Recommendation