FUNDAÇÃO GETULIO VARGAS
ESCOLA DE ADMINISTRAÇAO DE EMPRESAS DE SÃO PAULO

ANDRE URATSUKA MANOEL

Understanding Consumer Preferences Through Latent Spaces

SÃO PAULO
2017

ANDRE URATSUKA MANOEL

Understanding Consumer Preferences Through Latent Spaces

Monografia apresentada à Escola de Administração de Empresas de São Paulo da Fundação Getulio Vargas, como requisito para a obtenção do título de bacharel em Administração de Empresas.

Orientador(a): Prof. Dr. Gustavo Mirapalheta

SÃO PAULO

2017

Uratsuka Manoel, André.

Understanding Consumer Preferences Through Latent Spaces / André Uratsuka Manoel – 2017.

68f.

Orientador: Gustavo Mirapalheta

Monografia (bacharel) – Escola de Administração de Empresas de São Paulo da Fundação Getulio Vargas.

1. Machine Learning. 2. Consumer Preferences. 3. Marketing. I. Mirapalheta, Gustavo. II. Monografia (bacharel) – Escola de Administração de Empresas de São Paulo. III. Título.

ANDRE URATSUKA MANOEL

Understanding Consumer Preferences Through Latent Spaces

Monografia apresentada à Escola de Administração de Empresas de São Paulo da Fundação Getulio Vargas, como requisito para a obtenção do título de bacharel em Administração de Empresas.

Data de aprovação:

___/___/___

Banca examinadora:

_____
Prof. Dr. Gustavo Mirapalheta (Orientador)
FGV-EAESP

_____
Prof. Dr. João Luis Chela
FGV-EAESP

ACKNOWLEDGEMENT

Many people contributed to make this work possible. First and foremost, my wife, whose patience and support made all the effort bearable, and who gave me advice and encouragement, and my parents Claudia and Quintino, who gave me life and put me on the path. Bianca Bueno provided insights about some obscure movies that she knew like no one. I'd also like to thank my advisor Prof. Gustavo Mirapalheta, who helped me shape the final work and showed me how to finish. Also fundamental was the instruction of Prof. Eduardo Francisco, who showed me the way in a previous version of this work that finally turned into this piece. The author is also indebted to the GroupLens group at the University of Minnesota and to my colleagues and friends who made life interesting.

RESUMO

A técnica de Espaços Latentes possui diversas aplicações em áreas como o processamento de linguagem natural (PLN), reconhecimento e geração de imagens, interpretação e geração de textos, reconhecimento de desenhos vetoriais e tradução poliglota. Essa técnica permite o uso de vetores de *embedding,* i.e., vetores em um espaço vetorial k-dimensional que representam os objetos de estudo, em modelos de aprendizado cada vez mais sofisticados, servindo de base para novas áreas inteiras de aplicações. Esses vetores capturam características semânticas dos objetos estudados e podem ser reaproveitados em outros modelos, assim reduzindo o tempo de desenvolvimento e permitindo a transferência de informações entre áreas diferentes. Um exemplo em PLN são os vetores Google Word2Vec, ou Facebook FastText, amplamente utilizados em mecanismos de tradução, geração e interpretação de textos para representar palavras. A técnica de espaços latentes é explorada neste trabalho como forma entender preferências de usuários através um mecanismo de recomendações a partir da base de dados pública MovieLens da Universidade de Minnesota. Dessa base de dados foram utilizadas apenas uma sequência de triplas *(Id de usuário, Id de filme, nota)*, representando notas atribuídas por usuários a filmes Com a técnica de fatoração de matrizes, foram gerados dois espaços vetoriais k-dimensionais representando respectivamente usuários e filmes. A implementação utilizou a biblioteca *tensorflow* e fez uso de técnicas de Machine Learning como SGD. Diferentes tamanhos para os vetores foram comparados. Foi avaliada a capacidade dos espaços assim gerados de captar características não-triviais: foram escolhidos alguns filmes base, e os filmes mais similares foram avaliados qualitativamente para verificar a capacidade de generalização do modelo. O resultado desse estudo foi positivo: espaços latentes de diferentes dimensões são capazes de sugerir filmes subjetiva ou objetivamente relacionados. A qualidade dos resultados apresentou melhoria quando a dimensionalidade dos espaços vetoriais aumentou, mas o uso da fatoração de matrizes para estimação de gostos de um usuário com base em poucos exemplos teve performance baixa. Finalmente, um modelo alternativo baseado em redes neurais foi implementado, demonstrando positivamente que outras técnicas também podem ser empregadas para geração de espaços latentes. Os vetores gerados pela fatoração capturam características complexas, podendo ser reaproveitados no futuro para novos modelos, assim abrindo espaço para novos modelos.

Palavras-chave: Fatoração de Matrizes, Preferências, Tensorflow, Redes Neurais, GPUs, Espaços Latentes, Sistemas de Recomendação, Embedding, Comportamento do Consumidor, Aprendizado de Máquina, Modelos Preditivos..

ABSTRACT

The Latent Spaces technique has several applications in areas as diverse as Natural Language Processing (NLP), image recognition and generation, text interpretation and generation, recognition of vectorized drawings, face recognition and multi-language translation. It permits the use of embedding vectors – vectors in a k-dimensional vector space that represent objects of study – in increasingly advanced learning models, forming the basis for whole new areas. Those vectors can capture semantic characteristics of the objects of study and once trained can be reused in other models, which allows reuse of training time and transfer of knowledge, as with the Google Word2Vec and Facebook FastText set of pre-trained word vectors, widely used to enable the most advanced machine translation, text generation and interpretation systems. Latent Spaces techniques are explored in this work to understand preferences through an implementation of a recommendation mechanism on top of the MovieLens dataset from the University of Minnesota. From that data base we extracted solely a sequence of triples *(userId, movieId, rating)*, representing ratings given by users to particular films. Through Matrix Factorization, we generated two k-dimensional Latent Vector Spaces representing film characteristics and the corresponding preferences of users for those characteristics. Our implementation used *Google Tensorflow* and made use of typical Machine Learning techniques like SGD. We compared the mean-square error of our method depending on the dimensionality k of the Latent Spaces. We then evaluated the capacity of the Latent Spaces to abstract non-trivial information about films: we compared some films with those with small cosine distance to them, and found that our system could indeed find similarities that would be considered subjective related according to criteria that would be difficult to code for. Even though our Mean Squared Error decreased as the number of dimensions increased, we found that estimating individual user vectors had in average low performance when the number of ratings used for estimation was low, reaching a peak of errors as the number of dimensions matched the number of ratings. Finally, we quickly explored an alternative to generate user vectors through neural networks. The techniques exposed here support the case that Machine Learning techniques like Latent Spaces can and should be used in Business.

Keywords: User Modelling, Marketing, Latent Spaces, Matrix Factorization, TensorFlow, Machine Learning, Recommendation Systems, User Preferences, Embeddings, Latent Variable Models, Consumer Behavior

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

AI – Artificial Intelligence

CPU – Central Processing Unit

FA – Factor Analysis

GP – Gaussian Process

GPLVM – Gaussian Process Latent Variable Model

GPU – Graphical Processing Unit

SGD – Stochastic Gradient Descent

SVD – Singular Value Decomposition

LSI – Latent Space Indexing

MF – Matrix Factorization

ML – Machine Learning

MT – Machine Translation

PCA – Principal Component Analysis

RBM – Restricted Boltzmann Machines

RNN – Recurrent Neural Network

TABLE OF CONTENTS

1. INTRODUCTION

1.1. The Proliferation of Choice

The exponential advance of technology has brought about an increase in the number of options that any person is subject during a given day. While in earlier decades a typical person would have two or three options for films that they'd want to watch, with the increasing speed of broadband connections and new business models, a computer connected to the Internet can give that same typical person the option of thousands of films to be watched on demand. Similarly, thousands of options for clothes, books, cars, pictures, dishes, schools, or options to spend one's time are available at any time. As costs for storage and logistics are reduced, and with the proliferation of digital goods for which such costs are minimal, market structures have been disrupted and whole industries have been changing with ever increasing speed. Those changes have affected the way people interact with markets and how companies and their products are structured to serve their clients. Consumers have more options, companies have more selections to offer, because storage costs are not as much of a problem (long tail), and there are more ways to deliver products. The growth of e-commerce has removed the previous limitation of choices to what can be physically available at a Point of Sale. In October 2017, Amazon sold close to 600 million products (Scrapehero.com) which it can deliver in a matter of days or sometimes just hours from its depots. In 2017 retail ecommerce was already responsible for 2.3 trillion dollars according to Emarketer (2017).

Such breathtaking increase in the number and variety of options taxes the strategies developed for a world of limited choice, when it was possible for a consumer to browse through a menu of dozens of options or to make use of the recommendations from friends or acquaintances. The mere complexity of searching a space of thousands or hundreds of thousands of items causes "information overload" and generates a cost for the person doing the search and makes it so that the person will have a search cost that makes it unlikely that the optimal option will be chosen. The difficulty of finding a good choice causes what is called the "Paradox of Choice" (Schwartz, 2002), in which such increases in the number of options, while allowing for better choices, at the same time are utility-reducing due to the time spent and the feeling of frustration caused by the incapacity of the person to find the optimal option.

The difficulty in choosing is often a result of the difficulty for the agents to describe or even understand their own preferences. While it is usually possible for a person to compare several different options and order them based on preference, such preferences are usually not based on clear and easily describable rules. Instead, choices are often based on intuition and

rationalized later. This is not to say that choices are not rational, but instead that their reasoning can be very complex and their choices can use product characteristic that are not apparent, even to them.

The impact is even more important when it comes to entertainment where films and music are now often consumed digitally. While in the past consumers would choose between a fixed number of TV and radio channels or obtain physical disks or tapes, they will now download films and music nearly instantly. As media consumption is not restricted by a physical object, all catalog can be made available to everyone at a much lower cost. The result is that as of 2017, 10 of the top 20 open companies by market capitalization have new digital technologies at the core of their offerings, as can be seen on Table 1.

Top 100 global companies 1-20

Company name	Nationality	Industry	Rank +/-	31 March 2017		31 March 2009	
				Rank	Market Cap ($bn)	Rank	Market Cap ($bn)
Apple Inc	United States	Technology	32	1	754	33	94
Alphabet Inc-Cl A	United States	Technology	20	2	579	22	110
Microsoft Corp	United States	Technology	3	3	509	6	163
Amazon.Com Inc	United States	Consumer Services	-	4	423	NA	31
Berkshire Hathaway Inc-Cl A	United States	Financials	7	5	411	12	134
Facebook Inc-A	United States	Technology	-	6	411	-	-
Exxon Mobil Corp	United States	Oil & Gas	-6	7	340	1	337
Johnson & Johnson	United States	Health Care	0	8	338	8	145
Jpmorgan Chase & Co	United States	Financials	19	9	314	28	100
Wells Fargo & Co	United States	Financials	45	10	279	55	60
Tencent Holdings Ltd	China	Technology	-	11	272	-	13
Alibaba Group Holding-Sp Adr	China	Consumer Services	-	12	269	-	-
General Electric Co	United States	Industrials	11	13	260	24	107
Samsung Electronics Co Ltd	South Korea	Consumer Goods	39	14	259	53	61
At&T Inc	United States	Telecommunications	-8	15	256	7	149
Ind & Comm Bk Of China-A	China	Financials	-12	16	246	4	188
Nestle	Switzerland	Consumer Goods	-2	17	239	15	129
Bank Of America Corp	United States	Financials	69	18	236	87	44
Procter & Gamble	United States	Consumer Goods	-9	19	230	10	138
China Mobile Ltd	Hong Kong	Telecommunications	-15	20	224	5	175

Table 1 - Top 100 Companies by Market Capitalization. Source: PwC (2017, p. 35)

One alternative to deal with the problems caused by this "Paradox of Choice", “Overchoice” or “Information Overload” is for agents to give up on the expectation for an optimal solution that doesn't take into account the search costs, a strategy described as “Satisfacing” as opposed to “Optimizing” (Schwartz et al, 2002). An agent who adopts a satisficing strategy will search only until the point in which a good enough option is attained even if that option is not what would be the optimal solution absent search costs. From an economic standpoint, though, what is lost is a certain amount of market efficiency as a consequence of even small search costs, under certain conditions, is the appearance of monopoly pricing as demonstrated by Peter Diamond (1948). For consumers, the increase in choice has led to a swelling number of options, more difficult decisions, more time lost choosing, less efficient results and less satisfaction. The dissatisfaction comes from the

realization that even though there are plenty of available options, the amount of effort required to make choices increases proportionally, resulting in a phenomenon called the paradox of choice, overchoice or information overload. In the limit, while theory would predict more choice resulting in better choices and more satisfaction, the excess cognitive cost required in choosing leads to suboptimal choices and limits the generation of value.

1.2. Recommendation Systems

Companies can alleviate the excessive choice problem by understanding their complex preferences and offering recommendations to their customers, thus reducing the number of choices that they are required to make. Recommendation systems usually make use of a combination of two basic approaches: content-based, and collaborative filtering, along with hybrid systems (Adomavicius and , 2015). A content-based approach would identify user characteristics such as gender, age, location, stated preferences, and membership to groups to predict a user's preferences.

The content-filtering approach on the other hand works by comparing user tastes based on their purchase decisions and not on other characteristics. Typically, such a system would film users with similar tastes and make inferences based on those users tastes. In all those approaches, what is being accomplished is the development of an understanding of the structure of user preferences. The content-based approach is the more familiar one as there are several typical steps like using multiple regressions to evaluate the contribution of each of the chosen factors to user preferences, or factor analysis and clustering to concentrate the amount of information in fewer numbers and to identify actions that companies can take to improve their bottom-line by working on those factors. The collaborative filtering technique, though, works by ignoring such knowledge and instead focusing on using the set of user preferences to gain insights into other users' preferences. While a content-based approach would try to discover user characteristics and how they affect each other, a collaborative-filtering technique would identify similar tastes through a technique like latent variable modeling, or clustering on previous choices. At the same time as it tries to identify what to offer customers, both techniques, in different ways, help understand preferences, and can help understand how they change.

1.3. Machine Learning and Latent Space Embedding

The increasing popularity of machine learning and AI, many new tools and resources are becoming available. Currently there are dozens of new frameworks that permit

working on Machine Learning areas specially in the R and python languages. Such frameworks typically nowadays can make use of multiple processors, and sometimes multiple computers, besides the new breed of Graphical Processing Units (GPU), capable of massively parallel vector computation. New techniques have brought about a spring of new techniques, especially the ones based on Neural Networks and Deep Learning, enabled by free tools like Tensorflow, Keras, PyTorch, Cafe, SciKit-Learn and others. Such toolkits make it feasible to develop models that were previously too time-consuming or too expensive to try. Advances in the training of neural networks turned them into the workhorse of machine learning and made breakthroughs in machine learning possible in areas like image recognition, translation, autonomous driving, Go playing, style transfer, among others.

One invaluable technique that grew out of widely used techniques like PCA and MF but since then gained new life is Latent Space Embedding, which assigns (or embeds) to objects, such as images, phrases, or words to vectors in a multidimensional space. Those embedded vectors capture and encode information about the space, the objects and relationships between them, and so can be used in complex models that may require that information. They have been used in areas as diverse as NLP, translation, image recognition and generation, and recommendations.

1.4. Problem formulation and research goals

With those new technologies, it should be now possible to devise tools that can simultaneously reduce the cognitive load for users while they make choices. At the same time, the same techniques should also help companies understand users and markets. In this work, we wanted to pursue the following research problem: "**Can we use Latent Space Embedding and other new Machine Learning techniques to understand user preferences?"**

To research the problem, we structured this report in 5 parts:

1. Statement of the problem and context: The Paradox of choice, Recommendation systems; Latent Space Embeddings, and Problem formulation
2. Theoretical Basis: we study recommendation systems as they are seen in the literature; then see the technical requirements for this work: Latent Variable Models; the familiar techniques of Factor Analysis and PCA; Embeddings and Latent Variable and Matrix Factorization as a way to obtain Vector Embeddings.
3. Methodology: Our Model; Training Process: Gradient Descent Regression; avoiding overfitting and local minima; Activation Functions; Dimensionality; The

Tensorflow Library; Predicting user ratings from previous ones without retraining the model; and analysis of the structure of the space through similarities between films;

4. Analysis: Description of data and its preparation; analysis of performance; description of user and movie embeddings; results of incremental predictions; study of model performance according to dimensionality; and other models.
5. Conclusion: we revisit our research problem and analyze our results, limitations, our contribution to the research problem and future research.

2. REVIEW OF PRIOR LITERATURE

2.1. Recommendation Systems and the Paradox of Choice

Recommendation systems come from many different areas and encompass a wide range of areas of study, emphasizing different goals and premises. That diversity of points of view led to a definition and nomenclature problem with overlapping and at times contradicting names for the concept. While reviewing studies on Recommendation Systems, Li & Karahanna (2013, p. 73) point out that terms like "Recommendation Systems" may sometimes be confused with, among others, "recommendation agents", "recommenders", "customization", "personalization" and "interactive decision aid systems", proceeding to define those terms on the table reproduced in Figure 1. According to their definitions, "customization" systems would be ones in which customers can choose between preexisting set of options without recommendations; "interactive decision aid" are more comprehensive systems, that ask customers for their preferences and present them with recommendations based on those elicited preferences; "personalization" systems use existing user information to adapt user experience to their needs; "recommendation agents" would be systems containing avatars that present the user with options often with voice and visual personifications; "recommendation systems" would be web-based systems that make use of collected and inferred user preferences to present the user with recommendations.

Terms	Definition	System's key features	System operationalization	Sample studies	Included in review?
Customization	Producing goods and services to meet individual customer's needs with near mass production efficiency (Jiao & Tseng, 2001)	Consumers proactively specify their needs. No recommendations.	Customization systems allow consumers to choose or search what they like from a pre-determined list of products (e.g., Dell.com).	Dewan et al. (2000), Thirumalai and Sinha (2009).	No.
Interactive decision aid system	An interactive tool that helps consumers to search for product information and make purchase decisions (Häubl & Trifts, 2000)	The system explicitly asks consumers for their preferences.	To elicit users' preferences, a user-aid dialogue was used to simulate the dialogues between a consumer and the decision aid system. Based on the elicited preferences, recommendations are then provided to the consumer (Wang & Benbasat 2009).	Häubl and Trifts (2000), Wang and Benbasat (2009).	Partially (include studies on a specific type of interactive decision aid—the recommendation agent).
Personalization	Personalization is the use of technology and customer information to tailor electronic commerce interactions between a business and each individual customer (Personalization Consortium, 2003)	Provide products and services that are tailored to an individual consumer's preferences.	An example includes providing real-time weather reporting based on the customer's location and alerting the customer to serious weather conditions (Sheng, Nah, & Siau, 2008).	Tam and Ho (2005, 2006), Sheng et al. (2008), Liang, Chen, Du, Turban, & Li (2012), Lavie, Sela, Oppenheim, Inbar, & Meyer (2010), Xu, Luo, Carroll, & Rosson (2011).	Mostly (exclude studies on personalization that is not focused on recommendations).
Recommendation system	A Web-based technology that collects a consumer's preferences and recommends tailored e-vendors' products or services accordingly.	Provide product or service recommendations based on an individual consumer's preferences.	The system uses data on purchases, product ratings, and user profiles to predict which products are best suited to a particular user (Fleder & Hosanagar 2009).	Schiaffino and Amandi (2004).	Yes.
Recommendation agent	A tool to facilitate users' decision making by providing advice on what to buy based on user-specified needs and preferences (Wang & Benbasat, 2005)	User animated avatar and human voices to present recommendations.	The interface for the agent technology providing options for changing the loudness, pace, range of frequency, and word emphasis with the text-to-speech (TTS) engine and for creating agent gestures and movement (Hess et al., 2010).	Hess et al. (2009), Qiu & Benbasat (2009).	Yes.

Figure 1 - Different terms and their meanings (Source: Li & Karahana (2015, p. 75)

Adomavicius and Tuzhlin (2005) proposed a model for recommendation systems that has three phases: the first phase is user preference elicitation where information about users is collected implicit or explicitly and a model of their preferences, or user profile, is built. Such profile would capture binary information on user interests, but could also contain more complex information. In the second phase or the delivery phase, user preferences are matched with available options and presented to users. Such matching was often made through rules written by specialists but now tend to use recommender systems, borrowing tools from data mining, statistics, machine learning, human-computer interaction and information retrieval.

Those tools belong to three categories depending on their recommendation approach: content-based, collaborative and hybrid, to which Li & Karahana (2013) add a fourth type: social network-based. According to algorithmic technique, those systems can also be classified as heuristic-based or model-based. Model-based systems build models of users and use those to construct a recommendation while heuristic-based system use rules, such as finding users with similar tastes and suggesting their favorite options. The third phase comprises a measurement of the impacts of the recommendations and adaptations to the system. That measurement is missing in many systems and is usually based on a measurement of the accuracy of the recommendations with limited use of more expansive metrics such as life-time value, loyalty and customer return metrics.

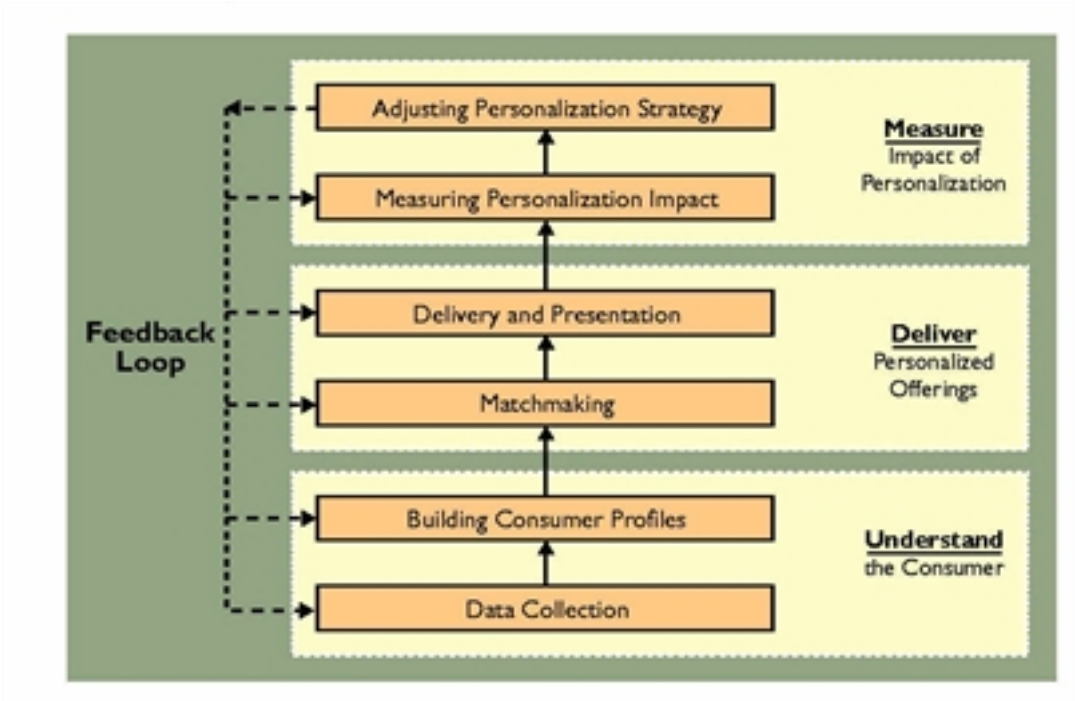

Figure 2 - Personalization Process (Source: Adomavicius and Tuzhlin, 2005, p. 87)

It should be mentioned that the meaning intended for Recommendation Systems in this work corresponds to the bottom three parts of the Adomavicius and Tuzhlin model depicted on Figure 2, encompassing Data Collection, Building Consumer Profiles and Matchmaking parts, i.e., the lower level systems that make the visual and interactive components that users interact with possible. The parts with which users interact should be assumed when required.

The use of Recommendation Systems changes consumer behavior and thus has impacts in the market. The first impact is the reduction in what is the reason for the creation for recommendation systems: the cognitive overload due to increased choices. The subject of cognitive load was studied, among others, by Todd & Benbasat (1992) who demonstrated that subjects acted as if they attributed a cost to cognitive effort and thus took effort minimization

into consideration when making decisions. That result is consistent from findings in the field of psychology from Schwartz et al (2002) in which, starting from the behavioral model of Simon (1955), two strategies used by subjects are identified and tested: some users are maximizers, trying to obtain the best possible results in any given situation; while others are satisfiers while satisfiers seek to obtain results that satisfy their minimum requirements without spending extra efforts to obtain the best possible. Their results indicate that adopting a satisficing strategy leads to higher life satisfaction and happiness, which indicates that the cognitive costs of choice persists and extends to other aspects of one's life. The fact that the effort required to deal with a larger number of options might lead to lower levels of happiness is what he calls the "Paradox of Choice".

Other researches have found that choice still has a net positive effect on happiness. Brynjoffson et al (2003) performed an empirical study of the impacts of increased variety in markets due to lower costs and found that electronic markets increase customer satisfaction even before dealing with the "Paradox of Choice" problem. But increased choice and recommendation systems which alleviate that problem seem to lead to even higher satisfaction. Brynjoffson & Simester (2011) have found that Recommendation Systems have the effect of increasing variety and leading to a long tail effect.

It should be noted too, that some voices, especially Pariser, have warned that that Recommendation Systems for information, based on collaborative filtering have the counter-effect of reducing variety in choice due to the creation of information bubbles, as subjects are less exposed to alternative viewpoints, which he described in a talk during TED 2011 (Pariser, 2011a) and in his book (Pariser, 2011b).

2.2. Latent Variable Models, Factor Analysis and PCA

A common technique for recommendations is the use of so called Latent Variable Models in which data is transformed in some way to extract hidden or "latent" variables, which explain the data better under some criterion – often, they explain more of the variance – than the original data. The idea has been extremely successful and has appeared under different names in different areas of science and has been rediscovered several times throughout the decades. According to Bartholomew et al. (p. 12), "*Latent variable models and factor analysis are among the oldest multivariate methods, but their origin and development lie almost entirely outside of mainstream statistics. For this reason topics such as latent class analysis and factor analysis had separate origins and have remained almost totally isolated from one another. This means that they have been regarded as separate fields with virtually no cross-fertilisation*."

Both Factor Analysis and PCA are in common use to obtain latent variables and allow the representation of objects of study as vectors in spaces where the component of the vector in each direction represent a new variable that explains data better than what was possible before.

PCA was introduced by Pierson (1901) and then rediscovered and named by Hotelling in 1933, although with some differences in interpretation and use. Pierson had a more geometric view while Hotelling's was variance-based, according to Jolliffe (p. 59). PCA can be explained by finding new representations of data through orthogonal transformations such that each new basis vector explains variation in increasing proportion the lowest its ordinal number, and such that each vector is uncorrelated with all the others. By ignoring but the first few dimensions, it's possible to describe most of the information in the original data more compactly, which is a very useful method when studying data sets and allows for easier data visualization, in a technique called Dimensionality Reduction.

Factor Analysis models data X as being composed by the product of a matrix of loadings Λ and a matrix of factors f, which, adding an error term e becomes:

$$X = \Lambda f + e \qquad (2.3.1)$$ (Jolliffe, p. 151),

Thus, solving a factor analysis problem, i.e., finding two matrices Λ and f that satisfy this formula is equivalent to the problem of matrix factorization.

The terms PCA and Factor Analysis are sometimes confounded with other area-specific terms and are other times used interchangeably, but one important difference between the two models is that, when in factor analysis some factors are dropped to reduce the dimensionality of the model, the whole model changes, including the factors that were kept, which is the opposite of PCA, where a change in the number of components won't affect the ones that remained. Another difference is that while Factor Analysis follows the model (2.3.1), PCA does not necessarily follow any model. As such, PCA can represent non-linear relationships between factors and data. Factor analysis also doesn't provide unique solutions. It's possible to find other equally valid new solutions, for instance, by rotation, which is often done in a way that will be useful for the objectives of the analysis. (Jolliffe, p. 151).

2.3. Embeddings and Latent Spaces

Latent Variable Models have evolved as its usage increased. In some areas, they soon became the state of the art. Information Retrieval had been an early adopter of the idea of vector spaces to represent documents since the word frequency vectors proposed by Salton (1975). But Deerwester et al. (1990), soon proposed a new technique called **Latent Space**

Indexing (LSI) which used a PCA technique on a modified vector of frequency of words to find relationships between documents and between documents and search terms. Such method didn't require exact matches between search terms and documents, opening the way to modern search methods. Similarly, Wei; Xin; Yihong (2003) also proposed the use of non-negative matrix factorization method, similar to the factor analysis method.

While PCA and factor analysis are two of the most used techniques that can produce a vector that can meaningfully represent an object; many other technologies have been introduced in the literature. An interesting one, that can learn information about objects unsupervised is the one presented in Hinton & Salakhutdinov (2006), where two neural networks of a type called Restricted Boltzmann Machines (RBM), an encoder and a decoder, were combined and trained together into what is called an **Autoencoder Neural Network**, an unsupervised technique that produced both a vector representation of an image and a probability distribution on input images. Images were encoded with the encoder part of the network, resulting in a vector. That vector would then be used as input to the encoder that would reconstruct the image from just that vector, thus forcing both networks to learn together an efficient encoding for the images. Such encodings provide surprising results as they are capable of representing similar objects with similar vectors. By using this system on a set of handwritten digits, the network was capable of finding vectors that placed representations of same digits together, without any previous information on the meaning of each image. On Figure 3, Hinton and Salakhutdinov compare the placement of images of handwritten digits on a vector space obtained by using the autoencoder to those obtained with a standard PCA approach.

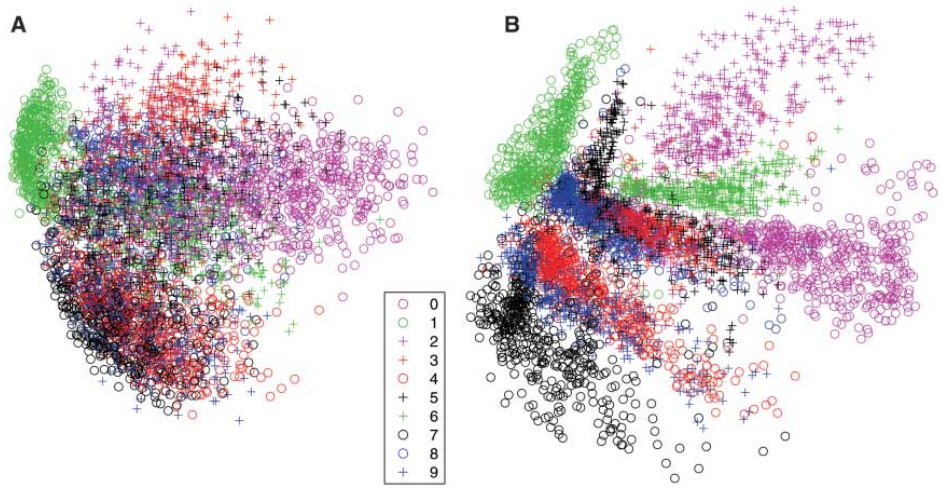

Figure 3 - Comparison of results from classification of digit images with PCA(A) and with Autoencoders(B). Source: Hinton & Salakhutdinov (2006, p. 506)

The autoencoder technique was also used successfully in other areas, such as Document Retrieval, where an unsupervised autoencoder was capable of "learning" relationships between unmarked documents through an efficient encoding of the collective frequency of appearance of words in documents.

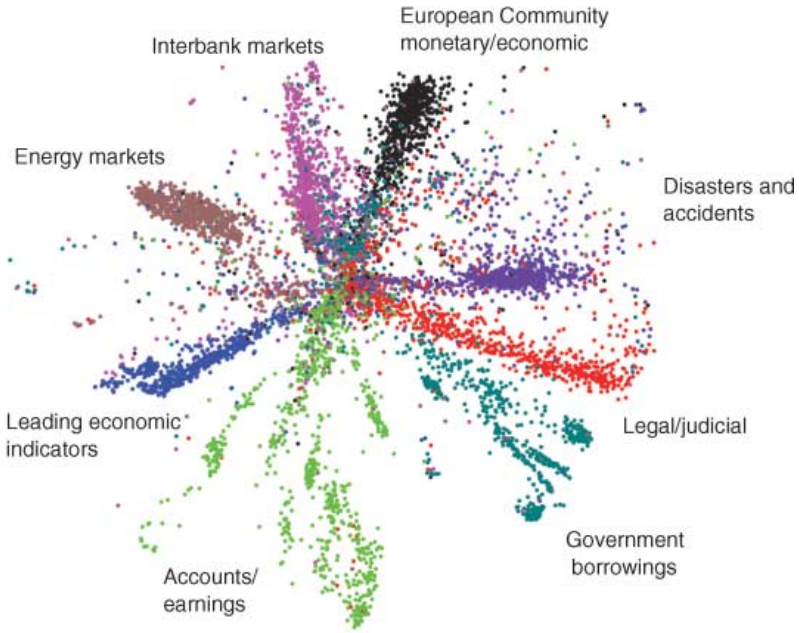

Figure 4 - Codes produced by a 2000-500-250-125-2 autoencoder for document retrieval by comparing the cosine distance between query and document vectors. Source: Hinton & Salakhutdinov (2006, p. 506)

Yet another interesting use of Latent Spaces was described by Baxter & Anjyo (2006), in which the authors translated doodles, i.e. line drawings, into a Latent Doodle Space. The coordinates could then be used to regenerate the given doodles, or to create new ones with slight variations. They describe three methods to generate those doodles (p. 481): PCA with regeneration based on thin-spline Radial Based Functions (RBF); PCA with regeneration based on a Gaussian Process (GP); and a Gaussian Process Latent Variable Model (GPLVM).

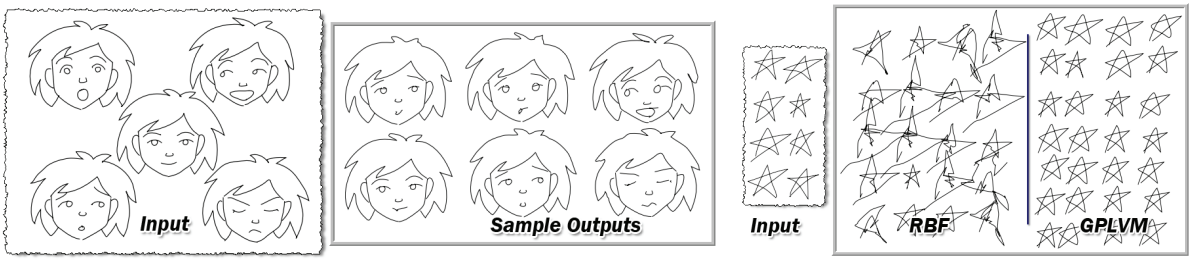

Figure 5 - Doodle generation from latent space encoding. Left images were drawn by humans. Right images were generated by the program from the vector representations. Source: Baxter & Anjyo (2006, p. 481)

Probably the most successful use of Latent Space Embeddings comes from the work of Mikolov (2013a, p. 749), where he created a vector representation of words with a recurrent neural network model whose goal was to, given a word, predict words that would surround it in a large corpus of text. It was found that the vector representations exhibited a structure and emerging properties through each some word relationships between words would be marked by similar offsets. The vector difference between the vectors for the words "King" and "Queen", for instance, would be about the same as the difference between the vectors for the words "Man" and "Woman", and the difference between the vectors for the words "Uncle" and "Aunt", thus capturing the concept of gender and the relationship between the two genders. Similarly, the vector difference between the words "Kings" and "King" would be close to the vector difference between the words "Queens" and "Queen", thus capturing the relationship of grammatical plural. That relationship is demonstrated on Figure 6.

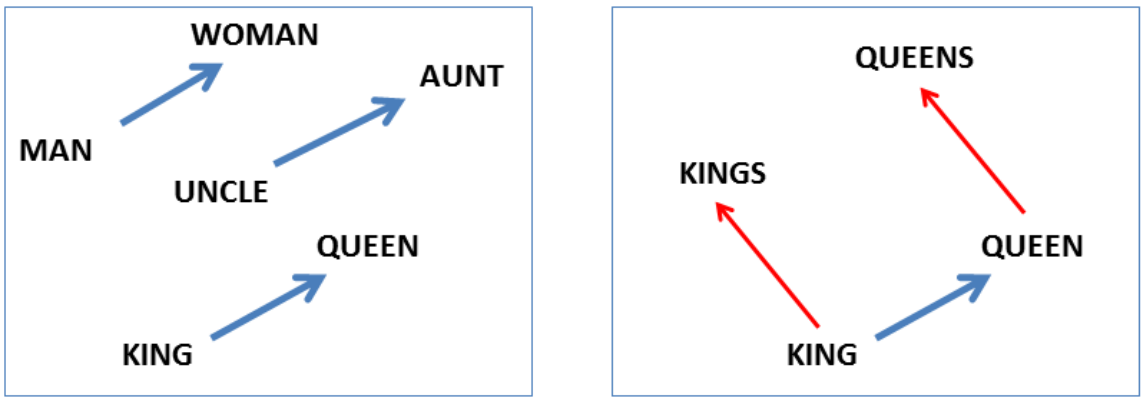

Figure 6 - Left Panel: Gender relationship between words in a projection of latent space. Right panel: number relationship between words in latent space. Source: Mikolov (2013c, p. 749)

It's important to note that the corpus of text was not previously annotated. The relationship between words was apprehended solely due to the frequency with which each word was used in relationship to the closest surrounding words in the corpus. Mikolov et al. (2013c) went further and with a more efficient model, trained a larger corpus and demonstrated that the

embeddings captured not only morphologic relationships between words such as gender and number, but also more complex semantic relationships, like the relationship between countries and capitals demonstrated on Figure 7.

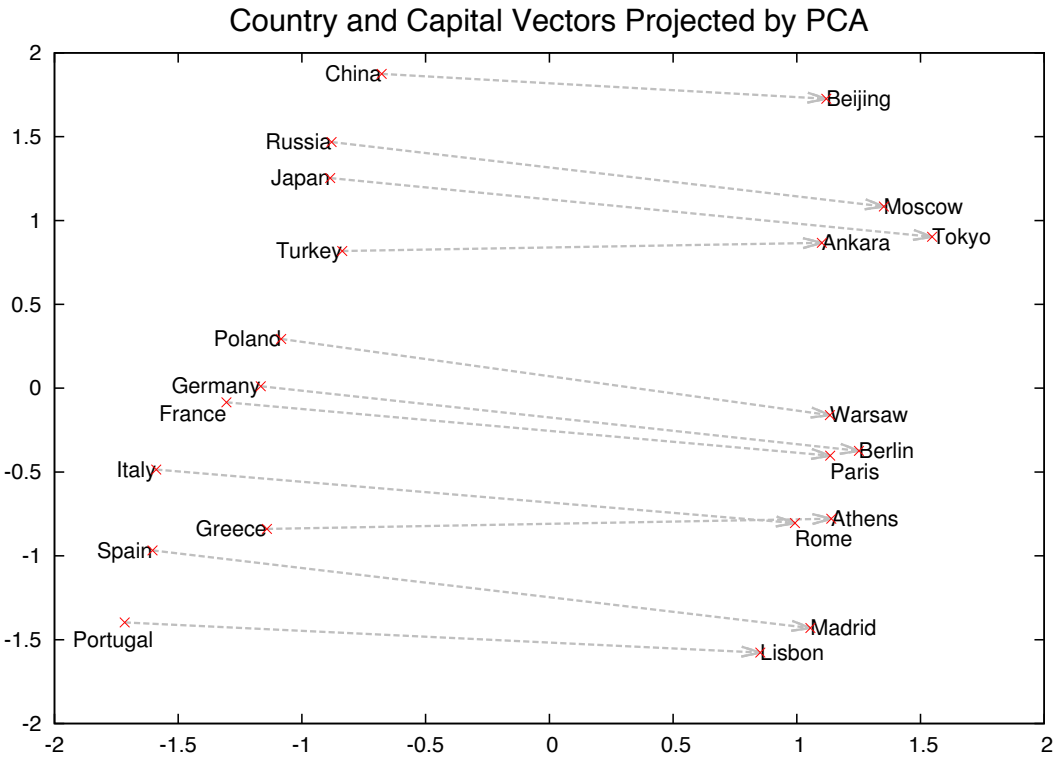

Figure 7 - Country-capital relationship encoded in word vectors. Source: Mikolov et al. (2013, p. 4)

The success of the embedding model led to widespread use of such vectors, specially after Mikolov (2013b), which published the techniques and the code used to create them, along with a set of pre-trained vectors, which were then used as a component for more complex models. The use of so called word embeddings were very important for the subsequent advances in other areas like Machine Translation (MT), image captioning, text interpretation and question answering.

In Machine Translation, Johnson et al. (2017) not only used word embeddings as the representation of words for a complex neural network model capable of translating between several different languages, it also hinted at the possibility of language-independent semantic spaces for meaning, as demonstrated on Figure 8.

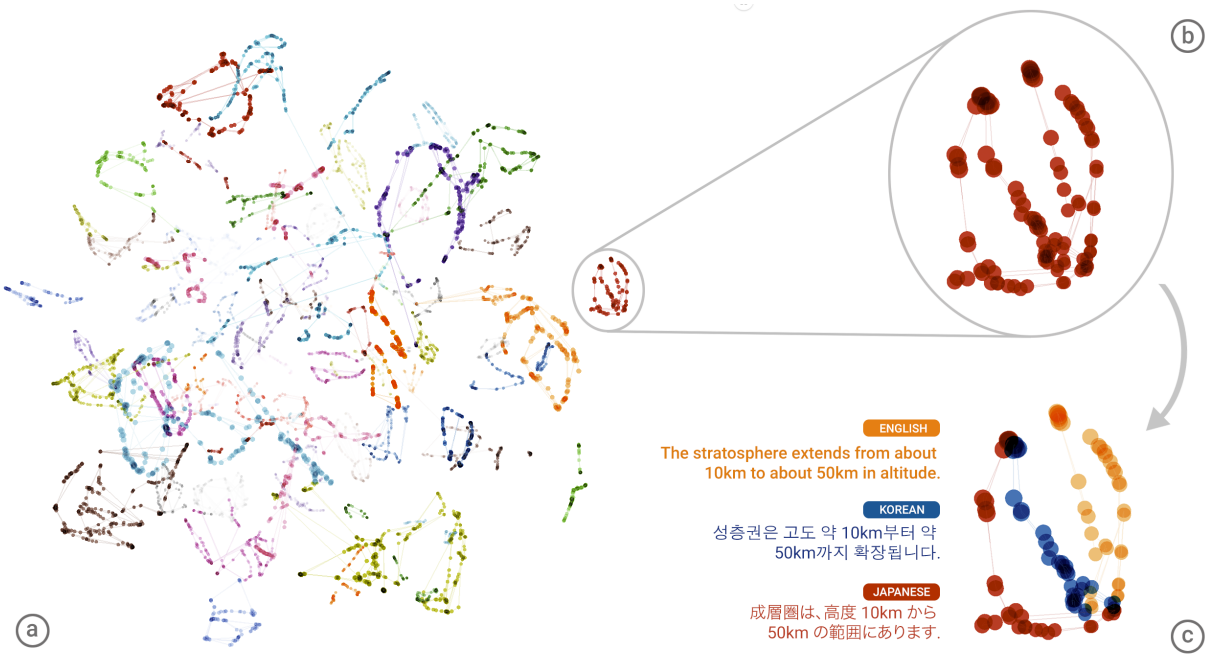

Figure 8 - Representation of context vectors of a machine translations model trained on English-Japanese and English-Korean examples. (a) projection, using t-SNE, of embeddings of variations of similar phrases, showing a clustering their meanings. (b) Variations of the phrase "The estratosphere extends from about 10km to about 50km in altitude". (c) Same image as (b), but coded by language. Source: Johnson et al. (2017, p.12)

2.4. Matrix Factorization for the Netflix Recommendation Prize

The Netflix Prize was introduced in 2006 and sought to advance the field of Recommendations. Netflix offered a dataset of 100 million film ratings by 140 thousand users and offered a one million dollar prize for any team that could offer recommendations that would reduce in more than 10% the Root Mean Square Error (RMSE) in recommendations compared to their existing Cinematch model. The prize generated a flurry of activity in the field of recommendations, leading to a series of papers on the subject. The winning team, BellKor Pragmatic Chaos published three papers: Koren (2009), Töscher; Jahrer (2009) and Piotte; Chabbert (2009) explaining their approach to solving the challenge. Instead of using a single technique, the BellKor Pragmatic Chaos team made use of several different techniques, which were combined to make predictions. One of the most important techniques, described in Koren; Bell; Volinsky (2009) and followed by this work was the Matrix Factorization technique. This technique is equivalent to finding latent factors in a Factor Analysis model, as described in equation (2.3.1). The factorization divides the rating matrix into a matrix of user tastes and a matrix of movie characteristics. That technique has been very successful as it can generalize well over missing values and can make use of a large set of tools.

3. METHODOLOGY

Our goal is to demonstrate the construction of a model-based collaborative filter recommendation system according to the classification Li & Karahanna following the Matrix Factorization technique with bias described in (Koren; Bell; Volinsky, 2009).

We use the MovieLens dataset, produced by University of Minnesota. The dataset containing ratings given by anonymous users to a set of films. Those ratings were given as numbers from 0.5 to 5.0. Our goal is to create a model that would predict the rating that a given user would likely give to a film they have not yet seen using a Latent Space Embedding to attribute a vector to each user and to each film such that such vector would represent preferences and features, thus capturing semantic information about the films, like genre and mood and about user preferences.

3.1. Model

The whole set of ratings can be understood as a 224,753 x 33,670 matrix $R = r_{ij}$, where r_{ij} is the rating attributed by the user i to the movie j. Such a matrix would contain 8.3 billion cells, but only 22.6 million data points were given, less than 1% of the data. We want to be able to predict those missing data points by creating a generalization of the user and movie data, combined in a way that approximates R.

The Matrix Factorization technique with bias was chose, because that technique is easy to understand with standard statistical knowledge, quick to train, and can supposedly be done incrementally. It is not the only technique that would generate a useful latent space; a classification or regression deep learning network, for instance, would provide similar results, as we will show on section 4.9.

The Matrix Factorization technique works by choosing a suitable number of dimensions k and assigning to each user i a k-dimensional vector u_i and to each film j a k-dimensional vector m_j. Defining U as the matrix whose i-ths rows are the vectors u_i and M as the matrix whose j-th columns are the vector m_j, we can have a model such that our matrix approximant R' is given by the first matrix form:

$$R' = UM$$

or:

$$R'_{ij} = u_i \cdot m_i$$

Each component of the vectors m_j represent a level of intensity by which a film possesses some previously undefined latent characteristic, and each component of the vectors u_i represent the intensity of user i's taste for that characteristic.

That model can be improved by applying some normalization to the data, by adding a bias scalar b_i^u to each user i and a bias parameter b_j^m to each movie j, representing characteristics beyond those that were evaluated by this model such as user generosity or stinginess (i.e., the degree by which each user would give a higher or lower rating *ceteris paribus*) and some measure of universal movie popularity. This allows us to write the matrix with the following formula:

$$R'_{ij} = u_i \cdot m_j + b_i^u + b_j^m \qquad (1)$$

Or in our second matrix form:

$$R' = UM + B^u + B^m \qquad (2)$$

If we extend the vectors u_i and $m_j$into $u_i{}' = (u_i, b_i^u, 1)$ and $m'_j = (m_j, 1, b_i^m)$, then we can again approximate R as the product of two matrices in our third matrix form :

$$R' = U'M'$$

Finding the U' and M' matrices is a matter of factorizing the matrix R, for which several techniques exist. Since the rank of matrix R is at most the largest of the numbers of rows and columns, and that the rank of matrix R' is at most k, R' cannot usually capture all variability in R and can thus, except in special cases, just approximate it.

While factorizing a matrix is easy with standard methods like Eigenvectors, SVD or LU factorization. To apply a standard SVD to the data, it would be necessary to impute missing data with zeros or averages, increasing the amount of work and adding errors to our results.

3.2. Training process

The simple technique that has been used most frequently on Machine Learning recently is an iterative optimization through Gradient Descent (GD). For this technique, the k-dimensional embedding vectors u_i and m_j and the bias vectors b_i^u and b_j^m are initialized randomly and then from each tuple (i, j, r) from the dataset representing the user i, the movie j and rating r given by them, a rating is predicted using equation (1).

This procedure is usually executed in batches, thus instead of single vectors we aggregate them into matrices representing several samples at a time. For each batch, we have a matrix $\dot{U}$of the user vectors for this batch as rows, a matrix $\dot{M}$ of movie vectors of the batch

as columns, the vector of $\dot{B}^u$ of user biases for the batch, the vector $\dot{B}^m$ of movie biases for the batch. The prediction for the ratings thus becomes:

$$\hat{R} = \dot{U}\dot{M} + \dot{B^m} + \dot{B^u}$$

The difference between the result vector $\hat{R}$ thus obtained and the vector of the actual ratings for the sample $\dot{R}$ is the error of our model. Our goal is to reduce such error by changing the values of the vectors u_i, m_j and the scalars b_i^u, b_j^m repeating the operation until we find a minimum.

Following the standard least squares method, it's more convenient to minimize the square of the error:

$$SqError = \left(\hat{R} - \dot{R}\right)^T (\hat{R} - \dot{R})$$

After each step, we calculate the gradient, i.e, the vector containing all the partial derivatives of the function we want to minimize (our loss function) in relation to each of our variables, thus providing the degree by which each variable affects our objective function. We update each variable in the direct proportion by which it affects the loss function:

$$\dot{U} \leftarrow \dot{U} - \alpha \frac{\partial loss}{\partial \dot{U}}$$

$$\dot{M} \leftarrow \dot{M} - \alpha \frac{\partial loss}{\partial \dot{M}}$$

$$\dot{B^u} \leftarrow \dot{B}^u - \alpha \frac{\partial loss}{\partial \dot{B^u}}$$

$$\dot{B}^m \leftarrow \dot{B}^m - \alpha \frac{\partial loss}{\partial \dot{B}^m}$$

where α is what is called the learning rate, i.e., the speed with which we update our model. The process was repeated with batches of samples from the training data in sequence. Each time all the samples in the training data were thus processed, we were said to complete an epoch, and the results were evaluated and then repeated or stopped.

Without loss of generality, we fixed movie vectors to be non-negative by applying an extra step that transformed each component into its absolute value. It doesn't cause problems because each component in the movie vector is multiplied by a corresponding component in the user vector. Both could be either positive or negative, but if both components are negative or both are positive, the result of the multiplication will be a positive number. By forcing the movie component to be positive, the variation in sign still exists, but comes from the user vector exclusively, thus simplifying our analysis.

3.3. Avoiding overfitting

In a model with enough parameters, it's possible to incur into the problem of overfitting, by which the model fits the samples so well that it is incapable of generalizing its results to data which it has not seen yet. Several techniques can be used to avoid overfitting. The first one, which we applied, was the use of regularization, in which we augment our loss function, i.e., the function we intend to minimize with a regularization parameter:

$$Loss = SqError + \lambda reg$$

where λ is the regularization constant, a hyper-parameter indicating the strength of our regularization. We used the regularization formula:

$$reg = \frac{u^T \cdot u}{k \, . \, b} + \frac{m^T \cdot m}{k . b}$$

where k is the number of dimensions in our model, and b is the number of ratings in each batch. By adding a regularization to our loss function, we favor models with low slopes, thus reducing the chance of overfitting. The regularization constant imposes a tradeoff between adjusting the system to the training data and keeping the low-valued parameters that are more likely to generalize to unseen data.

A second technique that we used was early stopping, by which we divided data into a training set and a validation set. After each batch, the current calculated variables were used to calculate the value of the loss function on each batch from the training set, and our parameter variables were accordingly updated. After each epoch, the value of the loss function for all the samples in the validation set was calculated using the parameter variables that had been calculated on the training set. When the validation-set loss stopped decreasing, our training would be interrupted, as further reductions in the training set loss were more likely to come from overfitting than from generalizable learning.

3.4. Avoiding Local Minima

Because both u_i and m_j are being updated, our model is not convex. Thus, there is the a that our model may find local minima. To avoid this problem, we used a modified version of the technique called Mini-batch Stochastic Gradient Descent (SGD). SGD adds a stochastic component to the model by performing variable updates at each step and reducing the size of those steps, while standard Gradient Descent would perform updates after each epoch. As we decrease the size of the steps, the optimization process becomes less deterministic and thus more likely to explore the parameter space, reducing the chances of convergence to a local minimum. More sophisticated SGD variants can also change the rate of update from by

adding a momentum to the learning rate (Momentum method), vary the learning rate for each parameter (AdaGrad), or variations. Our implementation combined a standard SGD with rule-based changes in the learning rate α and in the batch size b. We started with a large batch size (on the order of 100,000) and at each epoch, we monitored the loss on the validation set and reduced the learning rate and batch size when we seemed to be approaching (or past) a minimum, and increased it by a very slow amount (around 1%) otherwise. This reduced the effort necessary to finding suitable parameters for α and b. We did several tests using a standard SGD, our method, Adaptive Momentum (Adam), and Root Mean Square Propagation (RMSProp). Our method converged faster than the other techniques and was chosen. We didn't perform a detailed comparison of each technique as it would be out of the scope of this work.

3.5. Activation functions

The matrix multiplication model we describe is what could be described as a single-layer model. Many of the recent advancements in Machine Learning were due to the stacking of several layers of models that are formed by matrix multiplications, augmented by the application of an activation function. An activation functions is a function $f: \mathbb{R}^m \rightarrow \mathbb{R}^m$ that alters the output of a neural network layer of parameters W and bias b with input x turning it into an application of the following formula:

$$y = f(Wx + b)$$

The most typical activation functions are linear, tanh, sigmoid, ReLU and Softmax. A linear function is in fact an identity function, passing along the results of the matrix multiplication. Sigmoid and Tanh are often used because of their very convenient derivatives, which are useful for gradient calculations. They also limit their outputs to the [0,1] and [-1,1] images, which is convenient for some types of data. ReLU and its variations Leaky ReLU and Parametric ReLU are quasi-linear models, combining two linear segments with different slopes. They are often used in deep learning since they are easy to calculate and don't suffer from some pathologies that afflict exponential models like tanh and sigmoid. Softmax, finally, is often used for probability modeling as its output is a vector whose elements follow the probability axioms.

For our model, we slightly deviated from our matrix factorization model by using the following activation function:

$$f_{clipped}(x) = \begin{cases} 5 \; if \; x \geq 5 \\ x \; if \; 0.5 \leq x < 5 \\ 0.5 \; if \; x < 0.5 \end{cases}$$

This clipped linear activation function helped our model deal with situations in which predicted values were too high due to several factors applying to a given movie. This activation is roughly equivalent to a sigmoid activation function $\sigma(x) = \frac{1}{1+e^{-x}}$, after appropriately scaling our target ratings by a constant. It's interesting to note that using a sigmoid activation function would have turned our model into a kind of logistic (instead of linear) regression, although that wouldn't have had affected our results.

3.6. Embedding Space Dimensionality

The most important parameter in the matrix factorization model is the dimensionality k of the user and movie embedding vector spaces, i.e., the number of dimensions of each user and movie vector. To understand the quality of our factorization, we trained the model several times, varying the number of dimensions in each run. The mean square differences between predicted and actual values was compared for each dimension, thus allowing for the comparison of its effects.

3.7. Tensorflow Implementation of Model

The model was implemented using the Tensorflow framework in Python. Tensorflow is a machine collection of libraries and tools developed by Google, Inc., that interprets data as tensors – multi-dimensional data arrays–, and computations as graphs whose edges represent data flow and whose vertices as operations on those tensors. Graphs can be compiled as code specific to different hardware, including CPUs to GPUs. By using tensorflow for this model, we were able to take advantage of cloud-based GPU-enabled computing instances, which increased our processing speed by 100 to 1000 percent. The model, is reproduced on the APPENDIX A – Python code for MF model, and is represented visually in Figure 9, which was generated by the Tensorboard tool from the actual model used.

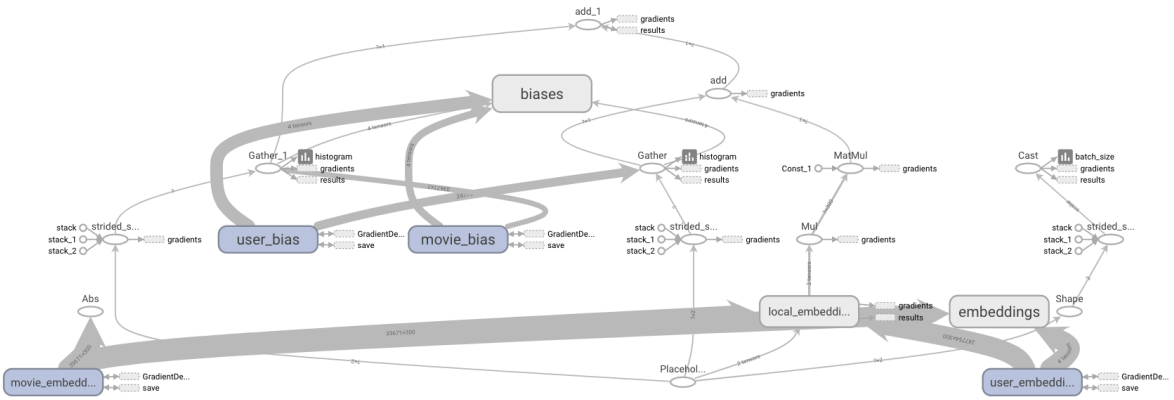

Figure 9 - Tensorflow Model for Matrix Factorization generated by the Tensorboard tool. Variables to be trained are represented in blue and arcs representing operations on them.

3.8. Testing the incremental predictive power of the model

After training, our model attributes a k-dimensional vector to each user and to each film, placing users and movies in the latent spaces of users and films, where vector components could be interpreted as representing film features and user intensity of preference for those features. If we have a new user and intend to estimate that user's vector, so that we can predict its preference, we could note the ratings that they attribute to films whose vectors are known. Supposing the movie vectors are known, by our matrix-factorization model it would be possible to find partial approximations to user vectors, using a simple regression based on the Roger-Penrose Pseudo-inverse:

$$\dot{u} = \left(\dot{M}\dot{M}^T\right)^{-1}\dot{M}^T\dot{r}$$

where $\dot{M}$ is a matrix whose columns are the movie vectors corresponding to the ratings $\dot{r}$ given by the user to those films. The vector $\dot{u}$ is the Least Squared Error-approximation to the vector.

We studied the quality of those approximations by using our test set, which contained users not previously seen in neither the training nor the validation sets. Our goal was to estimate the ratings given by each user in the test set by first calculating a user vector $\dot{u}$ from a varying number of ratings given by them. Those results were compared with the error obtained by using the average movie score as the predicted value.

3.9. Analysis of similarity of movies

One of the most important characteristic of embedding spaces is the ability to capture some form of meaning from objects and their relationships. We opted to study it through a test of similarity of films. We have already made calculations that give us a measure of how well our model can predict previously unseen ratings based on the mapping of users and movies into embedding spaces. We'd like to analyze the structure of spaces generated through similarities. We have thus chosen several different films and found films that have the lowest cosine distance:

$$cosine\ dist(v, w) = 1 - \frac{v \cdot w}{\|v\|_2 \|w\|_2} = 1 - \frac{v \cdot w}{\sqrt{v \cdot v + w \cdot w}}$$

. We analyzed those films qualitatively to verify if we could find films that would be similar, sharing characteristics like genre and mood. A latent space embedding model should be able to identify shared characteristics in films the way they can identify meaning in words just by the frequency with which words appear together. We repeated that analysis for different dimensionalities of the embedding vector spaces, focusing on 10 and 300 dimensions, so as to gain an insight into how an increase in the dimensionality of vector spaces affects the ability of the model to capture such meaning.

The results of this analysis are available on section 4.5, with further data on APPENDIX B – Lists of films with high degree of similarity (300 dimension latent space)

4. DATA ANALYSIS

4.1. Description of the Data

For this work we used the MovieLens Latest Dataset (Harper & Konstan, 2015), downloaded in October 2016. This dataset is the result of the GroupLens project of the University of Minnesota. That group runs an online application that recommends films to users and collects ratings given by users to those films. The dataset contains 22,884,377 ratings given by 247,753 distinct users to 33,670 distinct films. The ratings are in the 0.5 to 5.0 range, in increments of 0.5. Besides the ratings, the dataset contains tags attributed by users to films, and their classification. The files provided in the dataset are described in Table 2. Given our goals, we used the ratings, contained in the file **ratings.csv,** described on Table 4. For our reference we also used the names of the films, contained in the file **movies.csv**.

File Name	Description	Entries	File size (bytes)
ratings.csv	Ratings provided by users to films	22,884,377	620,204,630
movies.csv	General information about films	34,208	1,729,811
genres.csv	Normalized genres attributed to films	66,668	1,304,401
tags.csv	Normalized tags attributed to films	586,994	21,904,823
links.csv	Ids of the film on other databases	34,208	725,770

Table 2 - List of files on the MovieLens Dataset

4.1.1. movies

The movies table, provided in the file **movies.csv**, has the structured described on Table 3. This table was not used during training, being used only to permit the subsequent analysis of results.

Column Name	Description	Number of unique entries
movieId	Unique integer referencing a unique identification number for the film on the MovieLens Dataset	34,208
Title	The name of the film, including sometimes the year, in parenthesis, or some other name of the film	34,208
genres	A vertical bar-separated list of the genres the film has been assigned to, from the following list: Action, Adventure, Animation, Children, Comedy, Crime, Documentary, Drama, Fantasy, Film-Noir, Horror, IMAX, Musical, Mystery, Romance, Sci-Fi, Thriller, War and Western. If no genres match, the film is assigned as (nogenreslisted).	1446

Table 3 - The structure of the movies.csv file

4.1.2. ratings

The ratings table was the only data used during the Matrix Factorization process to train our model. It consists on the rating given by users, identified only by numbers, to movies, also identified by numbers. The structure of the file is the one depicted on Table 4.

Column Name	Description	Number of unique entries
userId	Unique integer referencing a specific user of the MovieLens App	247,743
movieId	Unique integer referencing a specific film on the MovieLens App	33,670
ratings	A score from 0.5 to 5.0 in increments of 0.5 given from the given user to the film	10
timestamp	The time in which the rating was given by the user, given in seconds, following the standard Unix format.	17,764,339

Table 4 - Description of ratings.csv

The ratings given by users has some interesting properties. Scores were given in increments of half, but due to a bias in the way users score films, whole numbers are considerably more likely than half scores. That can be seen on Figure 10. That effect distorts the distribution of scores, which could be expected to be mono-modal if not for it.

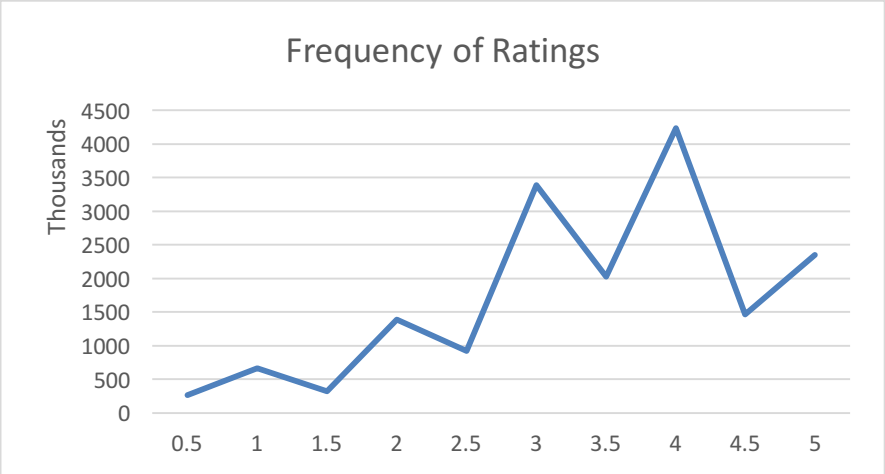

Figure 10 - Frequency of Scores

Mean of Ratings: 3.526

Standard Deviation of Ratings: 1.061

The distribution of ratings per user and per movies had present great differences. The number of ratings per movie presented a clear power-law relationship with a monotonically decreasing frequency and a very long tail with an average of 92.37 ratings per film as can be

visualized on the histogram of Figure 11, the distribution of ratings per movie, despite presenting a similar relationship, had fewer than expected ratings per user for very low numbers of ratings with smaller modes on 1, 5, and 10 and a larger mode on 15 ratings, as demonstrated on Figure 12. We attribute that strange distribution to the possible organization of the software used to obtain data.

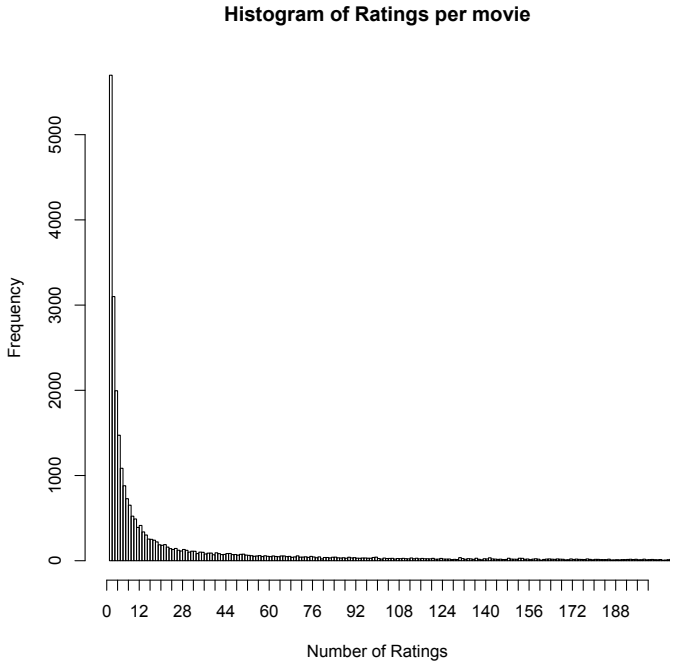

Figure 11 - Histogram of Ratings per Movie

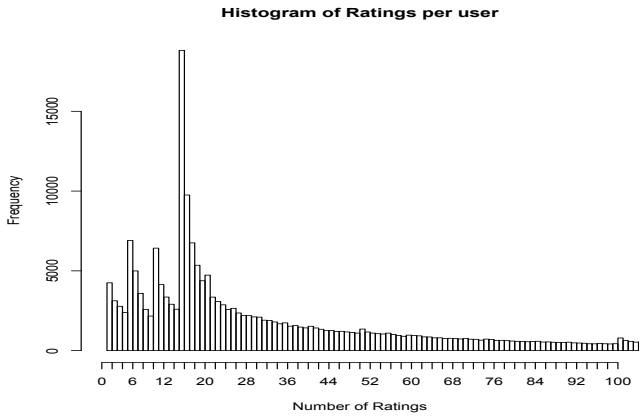

Figure 12 - Histogram of Ratings per User

4.2. Data preparation

Our first step while working with data was to clean up the data set. Some movieIds and userIds that were theoretically among the ones we would use didn't appear in the dataset.

Since we would put data in vectors, that would create “holes” in our vectors. For that reason, we renumbered movieIds on the ratings and movies tables.

We then split out data set into three subsets, extracting all the ratings for 10% of the users, to use for later tests of the incremental performance while estimating datasets. The resulting data was then randomly split into a training and a validation set. With the sizes given on Table 5.

Data set	Size (entries)	Percentage
Training Set	18,494,574	80.8%
Validation Set	2,054,954	9.0%
Test Set	2,334,850	10.2%

Table 5 - Distribution of data among the three sets used for training, validation and testing

The resulting data is exemplified on Table 6, which shows a sample of the data from the rating.csv file.

userId	movieId	rating	timestamp
132220	1356	4.0	859334955
132220	1367	4.0	859335028
132221	1	5.0	1319318876
132221	47	3.5	1319318897
132221	216	3.0	1319317535

Table 6 - Sample of rating data from the middle of the rating.csv file. The file was ordered by userId and movieId.

While processing the file, the Timestamp column was left unused and when loading the dataset, the ratings file was shuffled so that any given set of data would have enough randomness to avoid artifacts.

4.3. Training Performance

To train the system we used a cloud computer with an i7-6700 with 4 physical cores and 8 threads, running at 3.5 GHz and 64 GB of RAM. We compared the run time until convergence of the model while varying the number of dimension of the feature space. Training time varied from around 1 minute for dimension zero (mere calculation of bias), 4 minutes for one dimension, up to 1h35m for 300 dimensions. The same models were also trained on an Amazon P2 instance with one NVidia K80 GPU with 61 GB of RAM. By making use of the

GPU times it was possible to have lower and more consistent run times on the order of 60% as the number of dimensions grew, with run times that varied from around 1 minute for zero dimensions to 39m25s for 300 dimensions as can be seen on Figure 13 - Time to train the linear model, depending on the number of dimensions on the latent space The upper line corresponds to a 4 core CPU-only machine and the lower line corresponds to a machine with 1 Nvidia K80 GPU. Training time for GPUs were more consistent, but we believe that is due to the fact that while training with CPUs, there was more competition for cores, including from visualization and development tools, while the GPU on our servers were used only for the training task.

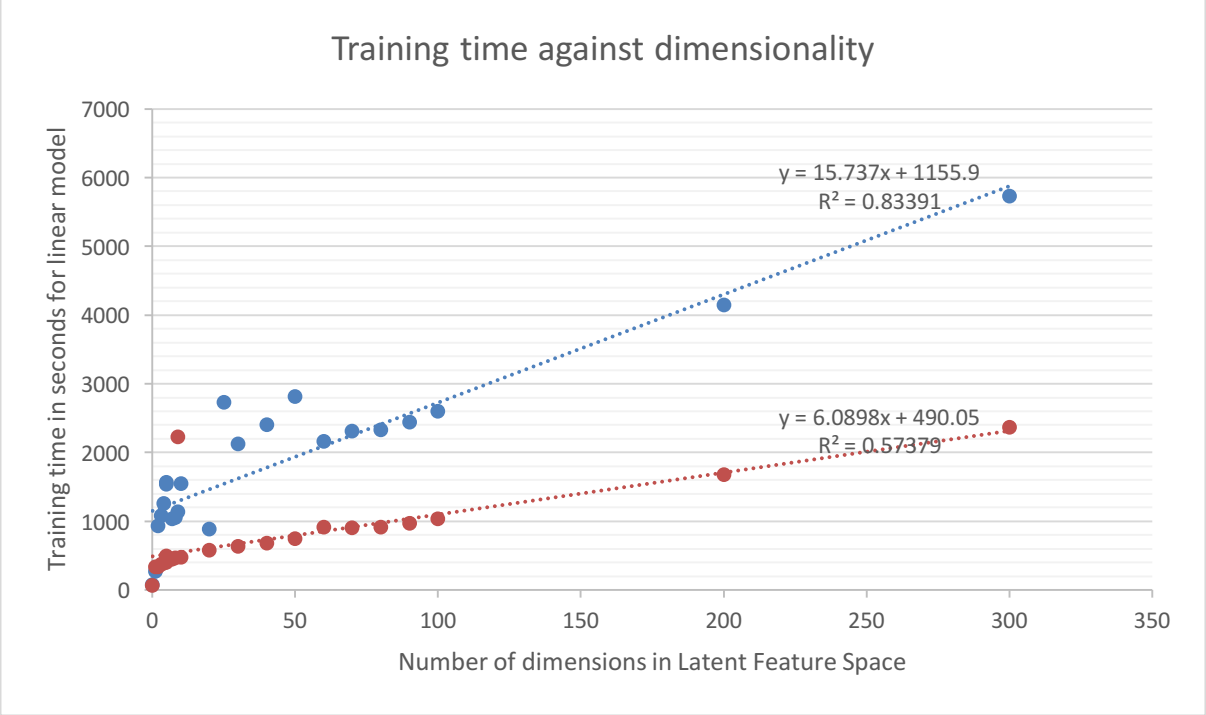

Figure 13 - Time to train the linear model, depending on the number of dimensions on the latent space The upper line corresponds to a 4 core CPU-only machine and the lower line corresponds to a machine with 1 Nvidia K80 GPU.

Model training showed a remarkably linear complexity as the number of dimensions increased. That was the best that could be hoped for as the number of floating point operations done on each step are proportional to the number of dimensions on the latent space. GPUs are particularly well suited for that kind of problem. We detected that the performance decreased quickly as batch sizes were reduced, which is why we decided to adopt an adoptive model where the learning rate and batch size were changed. At first a high learning rate and batch size would be used, but as validation set results got close a minimum, they would be cut in the hopes of breaking through local minima.

4.4. User Embeddings

After training models, each user is represented as a k-dimensional vector. That multivariate distribution can be understood as representing each user's preference for each of the k latent film characteristic that we track in our model. Since users are anonymous and the only information we have about them is the rating that they gave to some particular films, we decided to focus our attentions on the movie embeddings, for which we can bring outside information.

In the 1-dimensional case, the user vector is a single number, that indicates the user's sensitivity to an unidentified feature that allow people to differentiate movies. We believe that that feature may be an amalgamation of several features that happen to commove for large enough group of users. The distribution has mean -0.073 and standard deviation 0.921 although at first sign it may look like it follows Gaussian distribution, it fails the normality tests with a D'Agostino and Pearson p-value of 0.000.

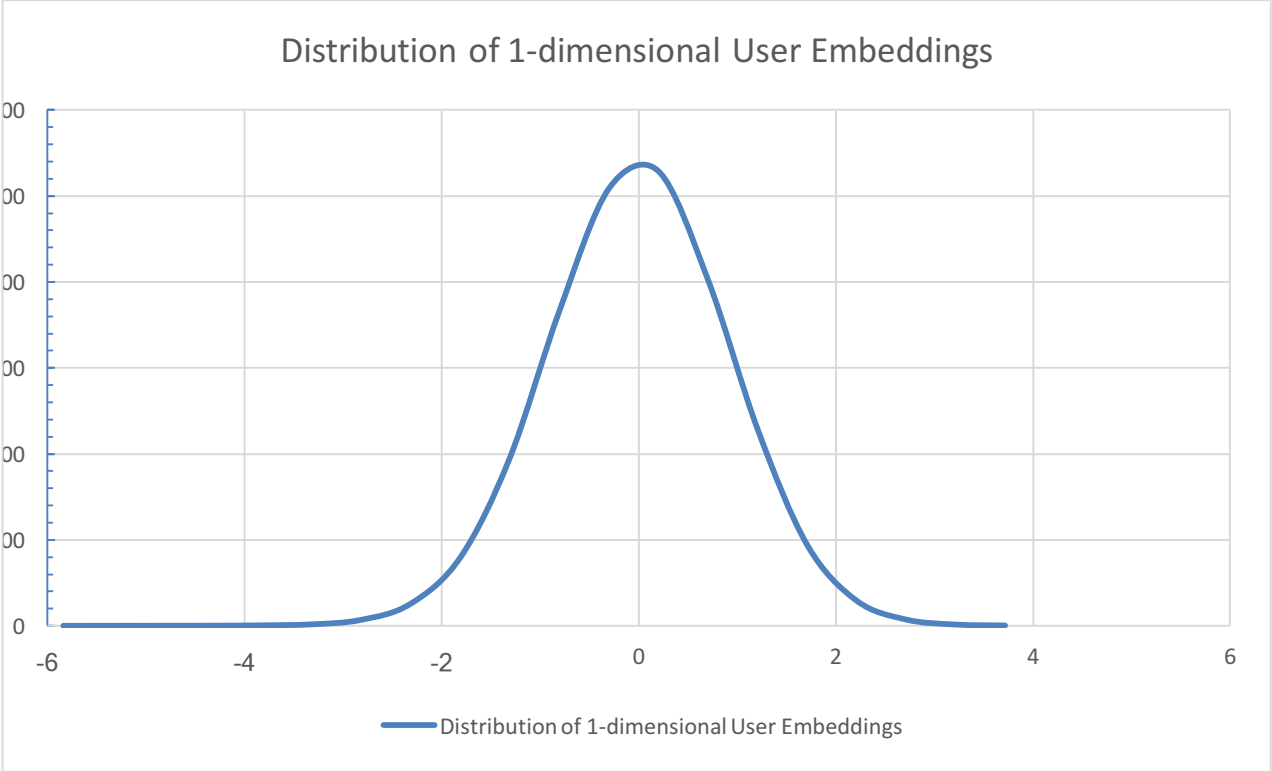

Figure 14 - Distribution of values on 1-dimensional User Embeddings

Increasing the number of dimensions, we find a multi-variate distribution that has the same shape as the 1-D version. For 300 dimensions, we find that the average user vector has a mean on each direction of 0.05774 ± 0.0032. It also fails tests of normality. It's possible to have an idea of the distribution through a projection of user vectors into yet another space,

obtained through a PCA transformation. The visualization of the top three vectors for the 300-dimension user vector embeddings is available on Figure 15.

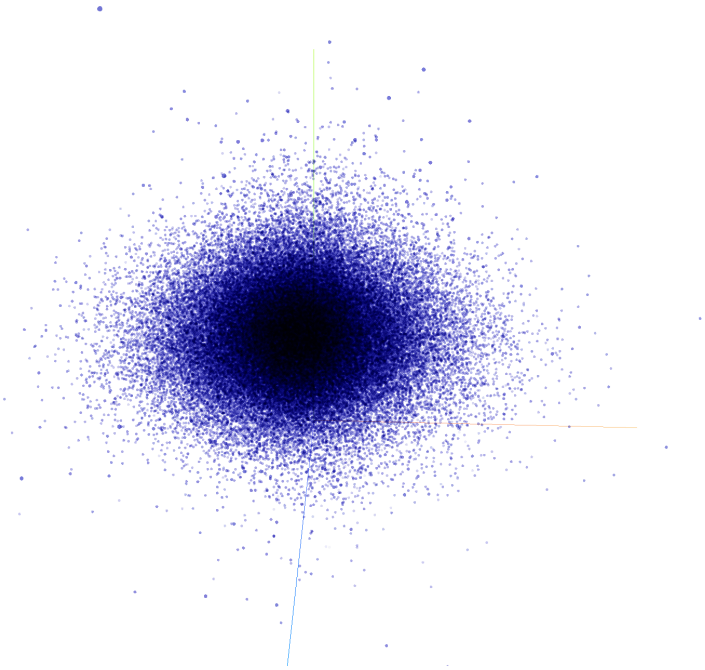

Figure 15 - Distribution of 300-dimensional user vectors along top 3 PCA vectors

4.5. Movie Embeddings

The analysis of the movie embeddings is made easier by the fact that each movie vector corresponds to a film that is possibly known. That way, we can compare them, and analyze the relationships between them. Our goal in this work is to find the hidden structure in the space of movie vectors that makes movie recommendations possible. In the latent spaces we sought, similar films will be close in the vector space, and those vectors will have encoded different meanings, so that distinctions that make people have different attitudes are reflected in them. That structure would codify genres, moods, themes, target audiences, the presence of famous directors or actors. For each characteristic x for which a person would say “I like films with x” a well-trained model with enough dimensions would codify x as a direction in that latent vector space. The characteristic described here are more noticeable in high-dimensional spaces.

To analyze embedding spaces we loaded the list of vectors on the Tensorboard tool, which enabled us to perform the visualization of high-dimensional states, through a PCA projection of the vectors, which can then be rotated and zoomed in.

In the 300-dimensional space, whose 3 top PCA dimensions are demonstrated on Figure 16, the first dimension captured films with and without many ratings. The collection of vectors plotted on this space formed the shape of a comet, with the head representing films

with a low number of ratings (most often less than 100) while famous films are on the right, on the sparser part of the space. The visualization allows us to see that films were clustered on large categories based on style, in a way that seems to reproduce common-sense intuition.

.

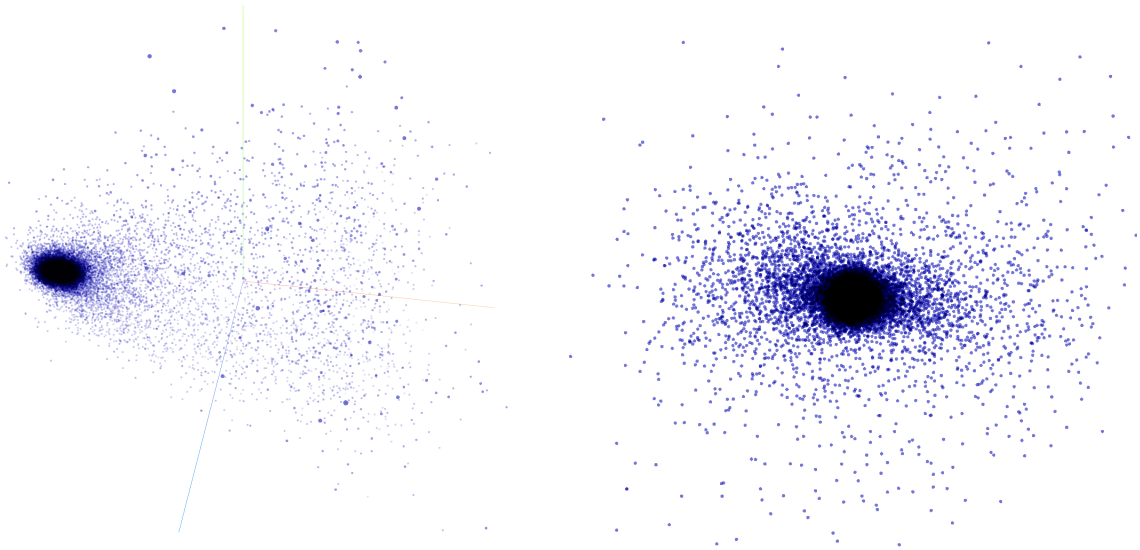

Figure 16 - Distribution of 300-dimensional Movie Vectors along 3 top PCA vectors. The shape is that a pyramid. On the left we have a side-view of the pyramid with the first vector running hotizontally. On the right we have omitted the first PCA vector and showed just the second and third, thus looking from the bottom in the direction of the head.

Using just the second and third PCA directions, and turning them into a 2D map, we selected some films with similar second and third components. We show on Figure 17(left), a series of films on the bottom of the map (small second component, negative third component): The Aristocats, Dirty Dancing, The Lion King, Snow White and the Seven Dwarves, Grease, the Little Mermaid, Aladdin, Sense and Sensibility and Beauty and the Beast. Those films may not at first seem related, but if we take into consideration the films on the right side of the map (high second component, medium to high third component) we can see a clear difference: Trainspotting, Apocalypse Now, Clerks, Taxi Driver, Twelve Monkeys, Blade Runner, Raging Bull, Crumb, City of Lost Children and Three Colors: Blue.

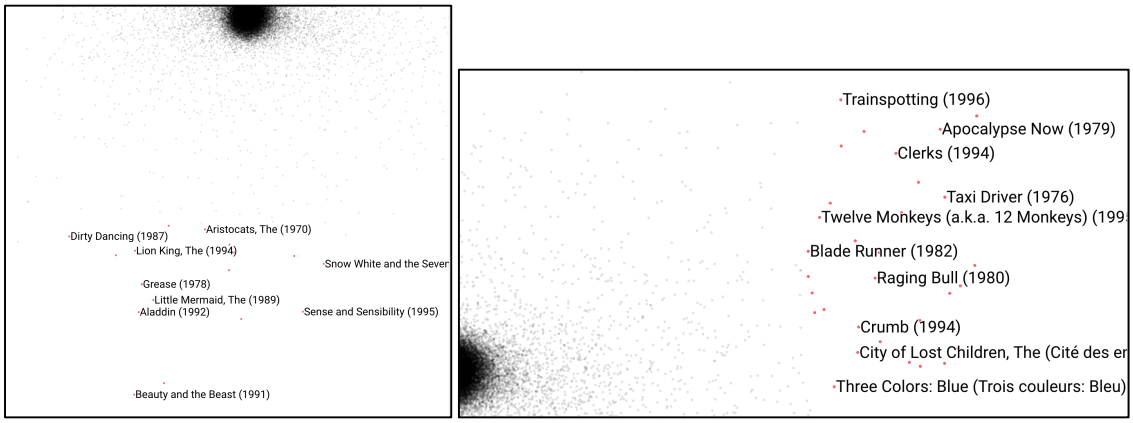

Figure 17 - Selection of films in the 2-D map using the same projection. On each panel the "head" of the pyramid is kept for reference, On the left panel, Animations and Musicals present in the lower half of the map. On the right panel, heavy, thoughtful films.

Another way to analyze the similarity between films makes use of all the components of the vector by comparing their cosine distance, i.e., the inner product of the normalized vectors. Our first experiment was to find the films that were most similar to "**The Sound of Music**, " a classic musical from 1965, shown in Figure 18. The result included, in decreasing order of similarity: Mary Poppins (Musical, 1964); Cinderella (Disney Animation, 1950); West Side Story (Musical, 1961); The Parent Trap (Disney Comedy, 1961); My Fair Lady (Musical, 1964); The King and I (Musical, 1964); Sleeping Beauty (Disney Animation, 1959); Miracle on 34th street (Family, 1947); and Lady and the Tramp (Disney Animation, 1955). The films are not all the same: they are musicals, animations and comedies, but they wouldn't look out of place close to each other in a VHS shelf as they are nearly interchangeable options for films. All of those films are more than fifty years old, which reflects an existing split in the movie market between generations. Those are not films that would be suggested to people who watch current musicals, or Disney Animations.

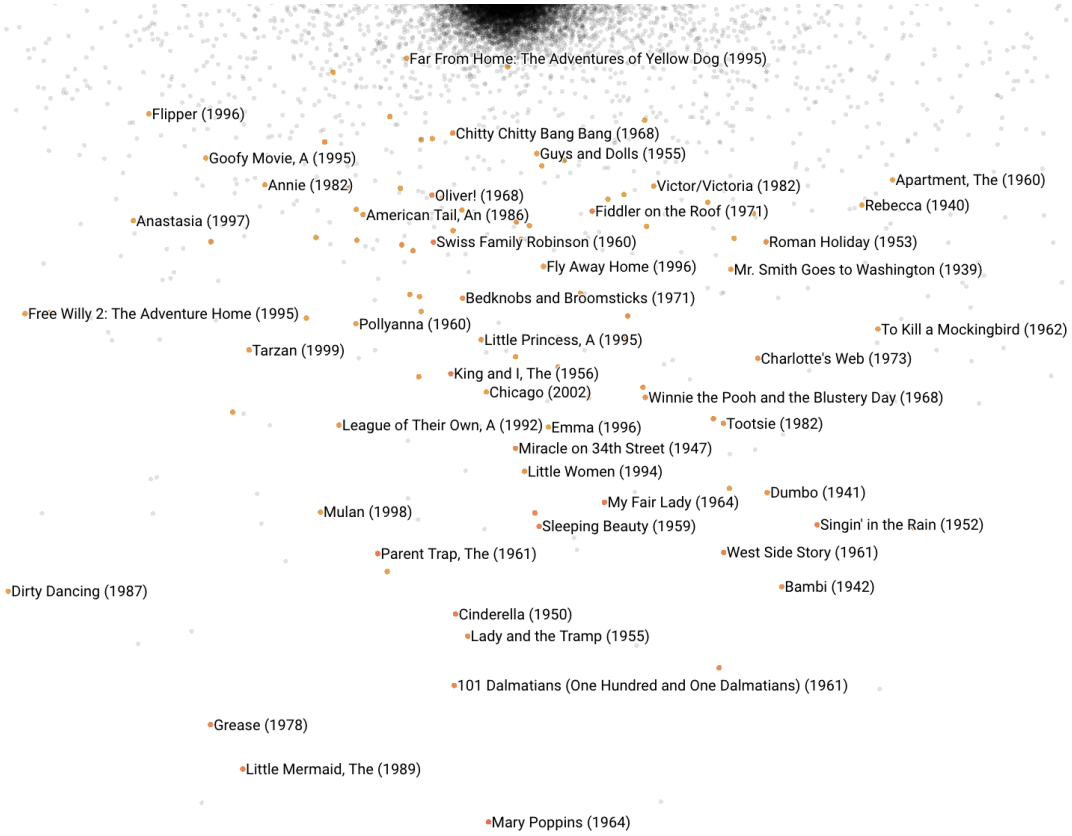

Figure 18 - Films with lowest cosine distance to "The Sound of Music" on 300-dimensional latent space. This distance calculation uses all components from the movie vectors and not just the two shown, so vectors that seem close on this map may actually be distant on the 300-dimensional vector space.

The ability of the technique to discover similarities between films based on nothing more than the ratings given by users films becomes more pronounced when the number of dimensions increase. That can be seen on Table 7, which illustrates that effect. With 3 dimensions, the first few films with high similarity wouldn't be considered similar by an external observer. On 10 dimensions, the films start to seem to cater to a similar public, one that is formed by geeks interested in fantasy and science fiction. *Exotica*, a drama, and *The Raiders of The Lost Ark* seem to be exceptions. Moving to 300 dimensions, the model is now able to have at the top of the list all films of the Star Wars trilogy, with the exception of two *Indiana Jones* films – both of which have Harrison Ford, also present on Star Wars as the main actor. A more expansive search, illustrated on **Figure 19** shows films on the general area of "boy films".

3 dimensions	10 dimensions	300 dimensions
Ferris Bueller's Day Off	Star Wars Episode V	Star Wars Episode V
Mulholland Falls	Star Wars Episode VI	Star Wars Episode VI
Planes, Trains & Automobiles	Lord of The Rings: The Fellowship of the Ring	Raiders of the Lost Ark
Garden State	Raiders of the Lost Ark	Indiana Jones and the Last Crusade
Defending Your Life	Exotica	Star Wars: Episode III
Joe's Apartment	Lord of the Rings: The Two Towers	Star Wars: Episode VII
Sideways	Lord of the Rings: The Return of the King	Star Wars: Episode II
Short Circuit 2	Battlestar Galactica	Star Wars: Episode I

Table 7 - Films most similar to Star Wars Episode IV, by latent space dimensionality

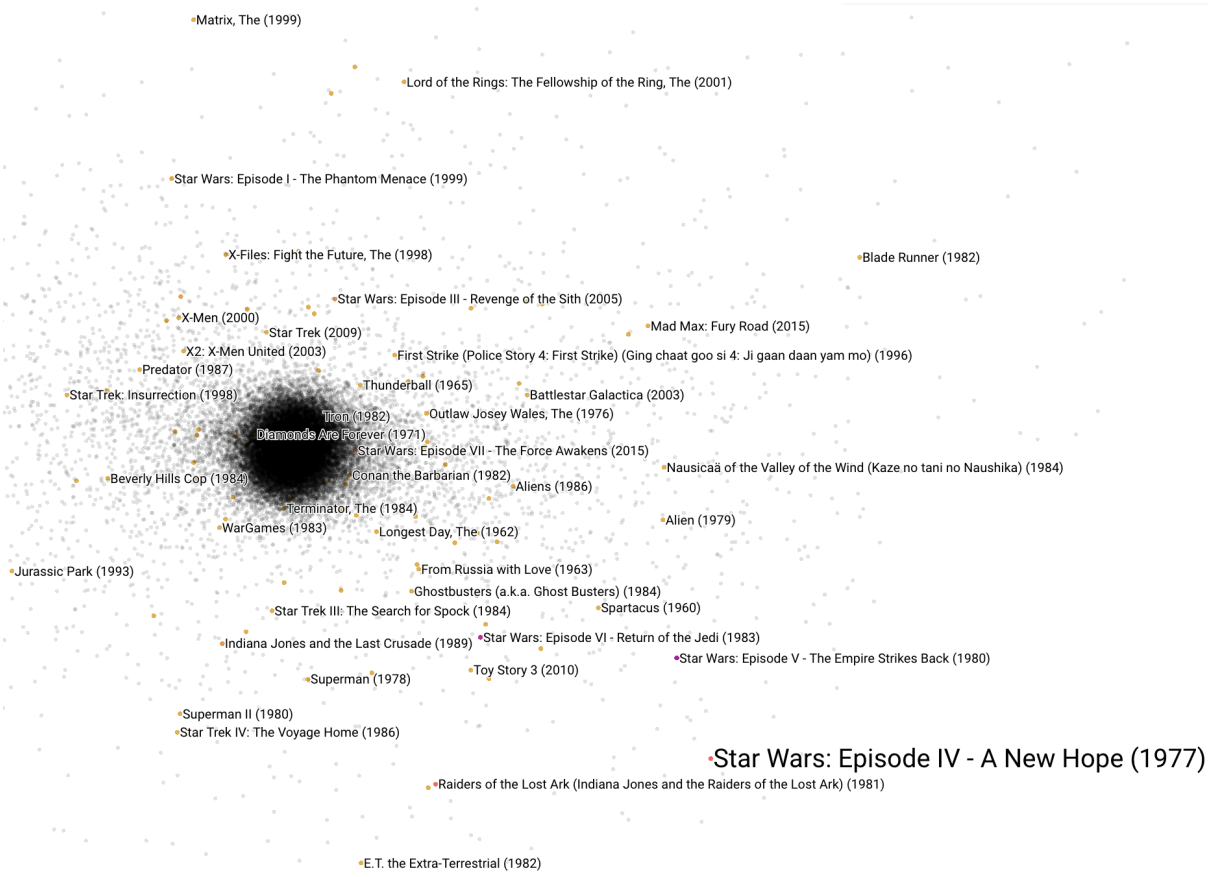

Figure 19 - Films with lower cosine distance to Star Wars: Episode IV on the 300-dimensional latent space of films. The cosine distance is calculated on all 300 dimensions, but only the second and third PCA components are displayed

The capacity of the model to detect more meaningful similarities as the number of dimensions increase is not limited to finding films from the same series. They may sometimes identify other general classifications of films. At 3 dimensions, the most similar films to Ingmar Bergman's The Seventh Seal seem unrelated. If we increase the number of dimensions to 10, we can see several European art-circuit films like Fellini's 8 ½ and La Dolce Vita; Godard's Breahless; Win Wender's Wings of Desire; Fritz Lang's 1927 classic Metropolis; François Truffaut's 400 Blows and Akira Kurosawa's Ran. Increasing the number of dimensions to 300 apparently switch to take into account not only the European sensibility, but also the claustrophobia and strong ambiance from Ingmar Bergman's Wild Strawberries; but also Kurosawa; Scorsese; Orson Welles and Hitchcock.

3 dimensions	10 dimensions	300 dimensions
Three Kings	8 ½	Wild Strawberries
Sky Captain and the world of tomorrow	La Dolce Vita	Ran
First Wives Club	Breathless	Touch of Evil
Independence Day	Wings of Desire	Strangers on a Train
City Hall	Metropolis	Last Temptation of Christ
Fahrenheit 451	400 Blows	The Third Man
Who's afraid of Virginia Wold	Ran	Paths of Glory

Table 8 - Films most similar to Bergman's The Seventh Seal

The vectors seem to encompass several different ways by which films are similar at the same time, but give more emphasis to different ways depending on the number of dimensions. As the number of dimensions increases, the criterion becomes very specific, but it can change for different films. In the examples listed on Table 9, the most similar films to Harry

Potter and the Sorcerer Stone, Tarantino's Death Proof, Mississippi's Burning and Footloose are listed. For each of those films, a different characteristic seems to be the most relevant. It can be "Belong to the Harry Potter series"; it can involve the violent and comedic nature of Tarantino's films; the activist mindset of films like Mississpi Burning; or the combined coming of age and emotional aspect of Footloose.

Harry Potter and the Sorcerer Stone	**Death Proof**	**Mississippi Burning**	**Footloose**
Harry Potter and the Chamber of Secrets	Grindhouse	Boyz N The Hood	Flashdance
Harry Potter and the Goblet of Fire	Planet Terror	The Last King of Scotland	An Officer and a Gentleman
Harry Potter and the Prisoner of Azkaban	The Wrestler	All the President's Men	Free Willy 2
Harry Potter and the Order of the Phoenix	Inglorious Bastards	Kramer vs Kramer	Three Men and a Baby
Harry Potter and the Half-Blood Prince	Kill Bill, Vol. 2	Good Morning, Vietnam	The Karate Kid
Harry Potter and the Deathly Hallows: Part 1	In Bruges	A Bronx Tale	Steel Magnolias
Harry Potter and the Deathly Hallows: Part 2	Amores Perros	My Left Foot	Father of the Bride
The Hunger Games: Catching Fire	Django Unchained	Ray	Anastasia
Chronicles of Narnia: The Lion, the Witch, and the Wardrobe	Kill Bill, Vol. 1	Awakenings	
Theme: Harry Potter / Fantasy	**Theme: Black Comedy, Violence**	**Theme: Politics, Empathy**	**Theme: Feminine / Emotion**

Table 9 - Films considered most similar using cosine distance for a 300-dimension vector.

For the film **Downfall**, the closest movies found on the 300-dimensional space were: The Lives of Others; Amores Perros; The Fighter; City of God; True Grit; All the President's Men; Boyz N' The Hood; Let the Right One In. Those films seem to mostly carry one particular theme: Downfall is a German film that tells the story of the last days of the Third

Reich in Berlin and was made famous due to repeated sharing of one of its scenes, in which the Adolf Hitler character has his last meeting with some of his top military advisors. Most of the films in this list depict adversity or oppression. They seem to come from different countries, different eras and have possibly different publics. The first film: **The Lives of Others**, could have been found by the model for being German and thus likely to be watched by the same public, obviating the use of latent factors. Downfall, received 4,943 ratings in the data set and The Lives of Others received 7,435 ratings. Of those, only 2,126 films were given by people who rated both. For those films, the correlation found was of 0.329, which seem insufficient to explain the degree of identified similarity. The existence of one or more latent factors seem more likely.

To the author, the most impressive capability of the model was the ability to detect films from the same directors or from the same series. Just as the model finds Harry Potter and Star Wars films as demonstrated on Table 7 and Table 9, but that capability seems most advanced for some prolific and remarkable directors, like Alfred Hitchcock. The 5 films most similar to “Rear Window”, for instance, are, in order: Alfred Hitchcock’s North by Northwest, Double Indemnity, The Third Man, and Vertigo; The Hustler; Alfred Hitchcock’s Notorious; Howard Hawk’s The Big Sleep; John Huston’s The Maltese Falton; Alan Pakula’s All the President’s Men; Alfred Hitchcock’s Strangers on a Train; and Billy Wilder’s Sunset Blvd.

At the same time, different levels of similarity appear to work at the same time. The films most similar tom Monty Python and the Holy Grail, and the top results are other Monty Python films. A second tier of similarities has other comedies from Mel Brooks, and finally a series of films like This is Spinal Tap, The Jerk and Airplane that seem to have in common a specific kind of humor that enjoys absurd situations.

Monty Python comedies	Mel Brooks comedies	Other comedies
Monty Python's life of Bryan		
Monty Python's The meaning of Life		
Monty Python's And Now for Something Completely Different		
Monty Python's Live at The Hollywood Bowl		
	Blazing Saddles	
A Fish Called Wanda		
		This is Spinal Tap
	Young Frankenstein	
		The Jerk
		Airplane
		Dr. Strangelove

Table 10 - Films most similar to Monty Python and The Holy Grail, in decreasing order of cosine similarity on the 300-dimensional linear latent space, classified by theme.

4.6. Incremental predictions performance

Our next step was to check if the matrix factorization model could be used to predict ratings for previously unseen users. In this work we trained both user and movie vectors at the same time. But if a user gave ratings to films with known vectors, we would be able to estimate the user vector and improve the estimation for each additional rating gave for the user. When testing the model for users in the test set, estimating user vectors from part of the data and measuring the predictions made for the rest, we discovered that our results depended strongly on the dimensionality of the space and the number of ratings used to estimate the vector. When the number of ratings used for estimations was considerably higher than the dimensions of the vector space, our errors were low. But the error in estimations hit a peak when the number of ratings used for estimation was equal to the number of dimensions in the

latent vector space, as showed on Figure 20. We haven't investigated the reason for this behavior, but among our hypothesis we have: the existence of films with too few ratings, that would require training; the overfitting of the estimated user vector to accumulated errors in the model; and insufficient randomization of ratings presented to users.

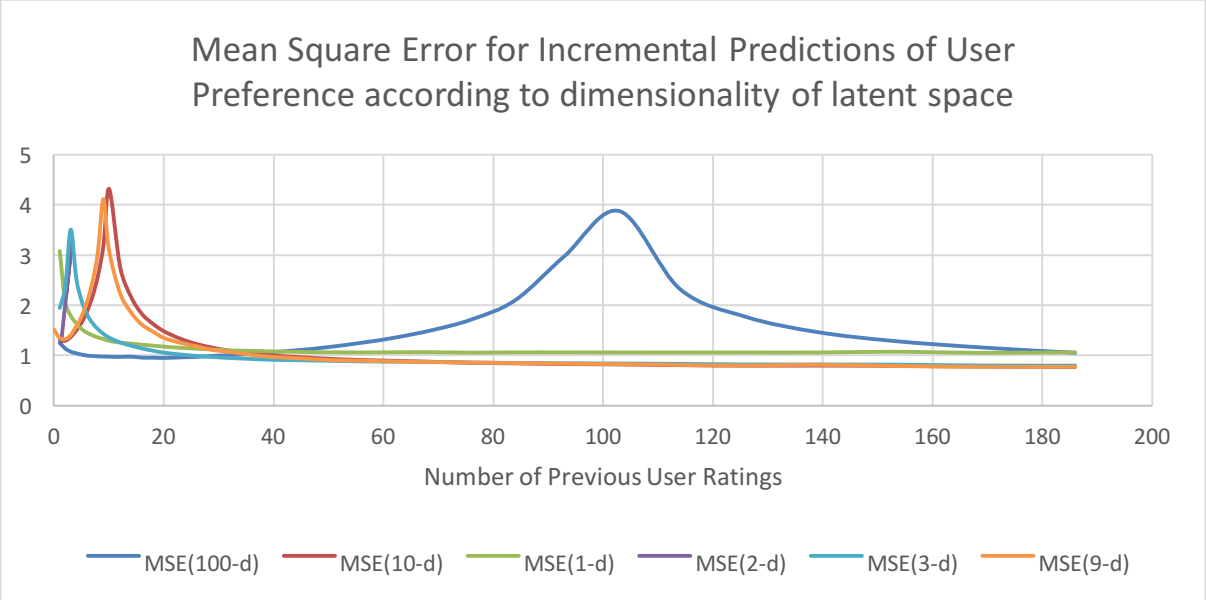

Figure 20 - Mean square error of predictions about user film preferences after user vector is estimated from varying number of scores for different dimensionalities.

Samples	Dimensionality of Feature Space										
	1	2	3	4	6	8	9	10	20	40	100
1	3,0862	2,2530	1,9443	1,4891	1,4061	1,3400	1,3572	1,3094	1,3143	1,3271	1,2483
2	2,0710	3,2664	2,3614	1,7646	1,4785	1,3494	1,3300	1,2981	1,2232	1,1955	1,1372
3	1,7958	2,2552	3,5043	2,3026	1,7024	1,4637	1,4039	1,3710	1,2074	1,1395	1,0729
4	1,6464	1,7660	2,5423	3,4869	2,0884	1,6546	1,5740	1,5020	1,2335	1,1361	1,0418
5	1,5282	1,5421	2,0817	2,5293	2,7147	1,9129	1,7743	1,6625	1,2710	1,1255	1,0141
6	1,4551	1,3809	1,8097	2,0773	3,8591	2,3020	2,0602	1,8881	1,3122	1,1320	0,9945
7	1,4045	1,2893	1,6386	1,8121	2,8425	2,9354	2,4618	2,1917	1,3746	1,1512	0,9865
8	1,3612	1,2325	1,5193	1,6476	2,3595	4,0656	3,0621	2,6108	1,4480	1,1534	0,9797
9	1,3312	1,1959	1,4269	1,5305	2,0653	3,0374	4,1172	3,2310	1,5238	1,1770	0,9756
10	1,2957	1,1601	1,3551	1,4352	1,8524	2,5014	3,1187	4,3243	1,6167	1,2027	0,9729
12	1,2610	1,1114	1,2573	1,3091	1,5958	1,9376	2,2569	2,7706	1,8522	1,2522	0,9681
14	1,2402	1,0778	1,1946	1,2362	1,4577	1,6474	1,8652	2,1708	2,1793	1,3215	0,9739
16	1,2182	1,0288	1,1380	1,1572	1,3276	1,4531	1,6189	1,8287	2,6002	1,3514	0,9542
18	1,1990	0,9940	1,0938	1,1138	1,2499	1,3302	1,4808	1,6330	3,3016	1,4283	0,9536
20	1,1799	0,9617	1,0542	1,0648	1,1872	1,2419	1,3548	1,4848	4,8131	1,4853	0,9479
23	1,1569	0,9469	1,0186	1,0260	1,1235	1,1644	1,2552	1,3364	3,0037	1,6611	0,9599
26	1,1381	0,9299	0,9906	0,9965	1,0759	1,1019	1,1857	1,2295	2,2949	1,8377	0,9694
29	1,1221	0,9212	0,9679	0,9725	1,0433	1,0457	1,1101	1,1557	1,9324	2,0661	0,9893
32	1,1037	0,9074	0,9497	0,9499	1,0159	1,0104	1,0732	1,0977	1,7096	2,3885	1,0039
36	1,0958	0,8935	0,9304	0,9322	0,9880	0,9774	1,0145	1,0458	1,4999	3,1581	1,0312
40	1,0834	0,8871	0,9126	0,9101	0,9639	0,9481	0,9738	1,0003	1,3621	5,2103	1,0643
45	1,0733	0,8767	0,9040	0,8990	0,9451	0,9263	0,9462	0,9683	1,2424	3,0353	1,1089
50	1,0673	0,8814	0,8935	0,8855	0,9246	0,9064	0,9133	0,9370	1,1548	2,3285	1,1648
56	1,0640	0,8837	0,8817	0,8753	0,9095	0,8865	0,8990	0,9126	1,0804	1,9028	1,2505

62	1,0681	0,8917	0,8763	0,8673	0,8968	0,8745	0,8849	0,8967	1,0321	1,6688	1,3518
69	1,0697	0,8861	0,8709	0,8609	0,8855	0,8612	0,8721	0,8770	0,9803	1,4638	1,5059
76	1,0603	0,8732	0,8587	0,8483	0,8694	0,8464	0,8644	0,8572	0,9434	1,3357	1,7138
84	1,0659	0,8617	0,8513	0,8463	0,8668	0,8337	0,8523	0,8456	0,9118	1,2115	2,0992
93	1,0664	0,8717	0,8436	0,8361	0,8535	0,8241	0,8407	0,8316	0,8784	1,1210	2,9811
103	1,0640	0,8754	0,8398	0,8332	0,8526	0,8200	0,8305	0,8246	0,8604	1,0679	3,8734
114	1,0633	0,8578	0,8315	0,8272	0,8400	0,8125	0,8155	0,8101	0,8393	0,9991	2,3209
126	1,0638	0,8483	0,8217	0,8181	0,8268	0,7986	0,8031	0,7996	0,8142	0,9454	1,7670
139	1,0637	0,8688	0,8202	0,8137	0,8277	0,7978	0,8185	0,8016	0,8042	0,9332	1,4641
153	1,0772	0,8497	0,8171	0,8132	0,8287	0,7945	0,7950	0,7962	0,7965	0,9175	1,2854
169	1,0570	0,8509	0,7995	0,7966	0,8074	0,7812	0,7780	0,7808	0,7750	0,8662	1,1565
186	1,0666	0,8550	0,7976	0,7938	0,8070	0,7753	0,7768	0,7739	0,7650	0,8442	1,0502
205	1,0479	0,8612	0,7879	0,7858	0,7948	0,7652	0,7907	0,7650	0,7506	0,8299	0,9824
226	1,0667	0,8321	0,7869	0,7865	0,7907	0,7634	0,7768	0,7613	0,7483	0,8127	0,9399
249	1,0649	0,8081	0,7788	0,7773	0,7834	0,7566	0,7583	0,7531	0,7352	0,7930	0,8893
274	1,0635	0,8204	0,7737	0,7750	0,7788	0,7528	0,7684	0,7502	0,7342	0,7933	0,8596
302	1,0576	0,7951	0,7602	0,7605	0,7697	0,7388	0,7408	0,7384	0,7223	0,7583	0,8245
333	1,0637	0,7957	0,7633	0,7585	0,7627	0,7386	0,7442	0,7351	0,7198	0,7667	0,8070
367	1,0614	0,8019	0,7498	0,7508	0,7511	0,7353	0,7350	0,7321	0,7108	0,7456	0,7900

Table 11 - Prediction error according to dimensions of feature space and number of samples. Highlighted in yellow is the worse performing number of training samples for a given user, for a given dimensionality. Highlighted in green are the number of training samples for a given user that outperform the naïve approach of using movie averages.

As the peak in errors occur when the number of training samples is in or around the dimensionality of the space, it would seem that the number of dimensions of the space should be chosen so as to always be much less than the average number of samples. Nevertheless, an increase in the number of dimensions leads to an improved error baseline in the asymptotic case, as can be seen shown on Table 11. The high error seems to be dominated by outliers while most of the samples are well-behaved. An alternative to decreasing the number of dimensions would be to identify those pathological cases and treat them differently.

4.7. Effects of the Dimensionality of the Latent Space

Setting the value of k, i.e., the number of dimensions of vectors u_i and m_j allowed for models that perform very differently as can be seen on **Figure 21**. When k was 0, the model made use only of biases, and thus was equivalent to attributing user preferences to just the mean of other users' preferences and had a mean squared error of 0.91. As we increased the value of k to 1, the model adds one extra degree of freedom and the error drops to 0.72 and starts decreasing. We found that the best results were obtained with a dimensionality close to 10 at an Mean Squared Error of 0.65, above which the error rate starts to increase again. The reason for this increase in error after a certain amount is apparently explained by an overfitting phenomenon, discussed on section 4.6.

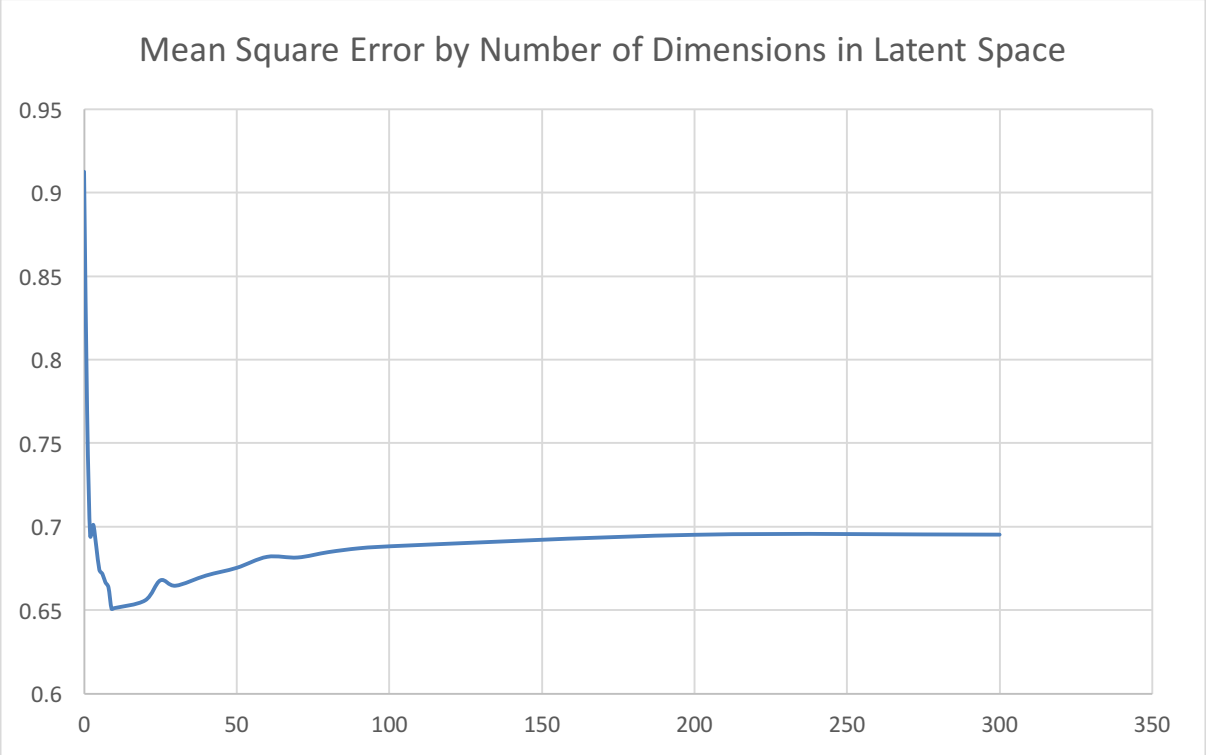

Figure 21 - Mean Square Error on Test Set by Number of Dimensions in Latent Space

4.8. Extra analysis

User vectors allow us to estimate the scores that users would give films if they had seen them. The ratings that we collected just as the ratings given by film websites are based on an average of the ratings given by only the users who contributed with ratings. Those users are a self-selected group that can be more or less generous than the general population. It is a biased estimate of the mean movie rating. A better estimate could arguably be obtained by multiplying the movie vector obtained by matrix factorization by the average of all user vectors. The best films according to average ratings are all unknown films with just one and two ratings and maximum score, so we discarded films with less than 1,000 ratings, and compared the mean of the ratings given by users and the estimated mean by all users on Table 12.

Rank	Top Films According to Simple Mean	Top Predicted Films
1	The Shawshank Redemption	The Shawshank Redemption
2	The Godfather	Black Mirror
3	The Usual Suspects	The Godfather
4	Schindler's List	The Usual Suspects

5	The Godfather: Part II	Schindler's List
6	Seven Samurai	Band of Brothers
7	Rear Window	Cosmos
8	One Flew Over the Cuckoo's Nest	The Lives of Others
9	Fight Club	Voices from the List
10	Casablanca	Twelve Angry Men

Table 12 - Films with more than 1000 ratings that have the highest mean ratings in the MovieLens dataset. On the right column films that rank higher are recorded in green and films that rank lower are recorded in red.

We also evaluated films that would have higher scores with the rest of the public than with the actual raters. Among films that seem to be disregarded by people who watched and rated them are: A Life Less Ordinary; Batman & Robin; Masters of the Universe; 9 ½ Weeks; Superman IV; The Getaway; Blind Date; Police Academy 6; and Police Academy 5.

Other films appear to be more well regarded by the public than by the public in general. The ones with the largest difference include films that are considered all time classics: Touch of Evil; Evil Dead II; Fear and Loathing in Las Vegas; Raging Bull; Brazil; Crumb; Rushmore; Eraserhead; Annie Hall; Blue Velvet; M; Sunset Blvd.; The Thing; 8 ½; Manhattan; Ed Wood; A Clockwork Orange; Chinatown; Taxi Driver; The Third Man; Night of the Living Dead; 400 blows; Duck Soup; Reservoir Dogs.

4.9. Using other models to generate embedding vectors

As mentioned, matrix factorization is not the only method that can be used to create our vectors. To demonstrate that, we also implemented a 3-hidden layer neural network using the Keras library, whose structure is displayed on Figure 22, and trained it with the same data as the Matrix Factorization model. This model used ReLU activation between layers and, instead of regularization, used Gaussian Noise to reduce the possibility of overfitting. Despite being very different and not having had any optimization effort, the results we obtained were similar to the matrix factorization method, achieving just slightly higher MSE, and also sharing the same properties, i.e., allowing us to identify similar films by a simple cosine similarity measure.

The suggestions generated by this new model are different even if still subjectively useful. For the film **Sound of Music**, for instance, the following films were identified as the most similar: The King and I (Musical, 1956); Oliver! (Musical, 1968); My Fair Lady (Musical,

1964); Gone With The Wind (Epic Romance, 1939); Going My Way (Musical Comedy/Drame, 1944); South Pacific (Romantic Musical Comedy, 1958). One possible explanation for the difference is that the latent spaces seem to have different structures, which can be seen on Figure 23.

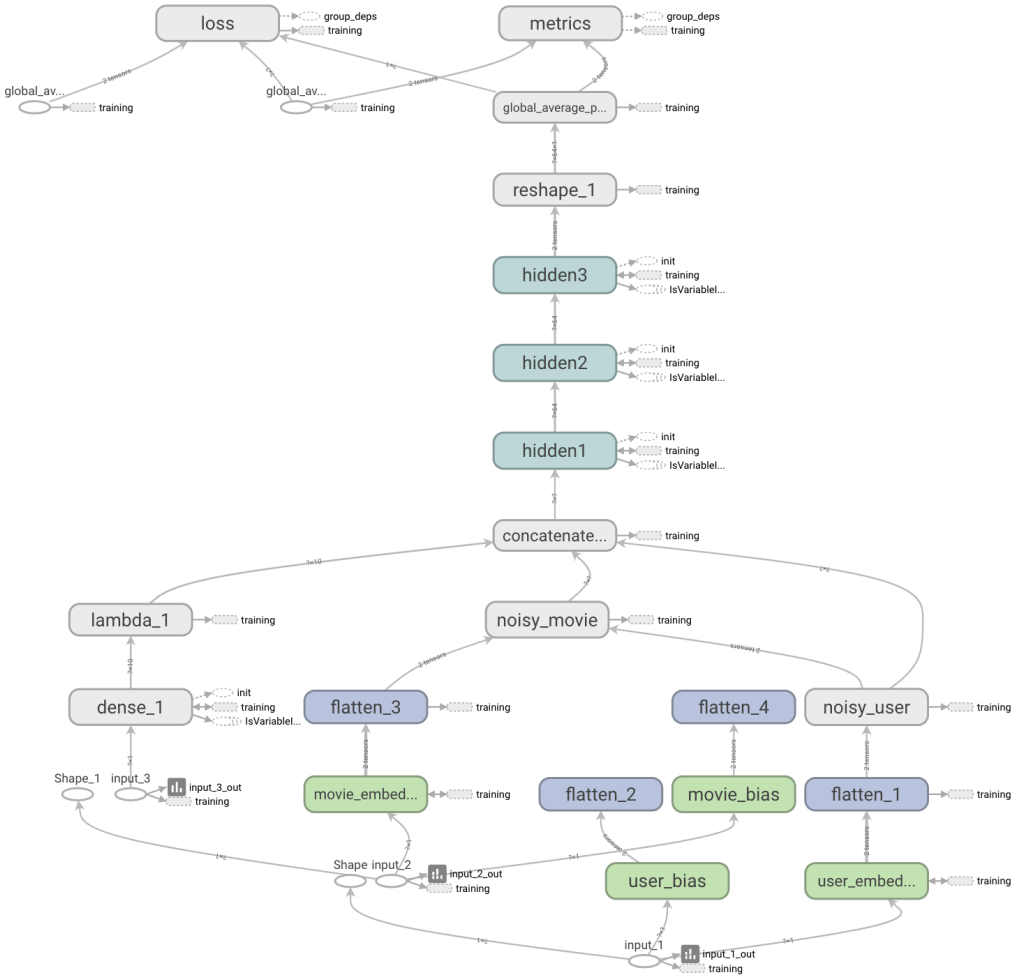

Figure 22 - Alternative Model for obtaining Film and User Embeddings using a standard 4-layer neural network with relu activations and regularization through a Gaussian Noise Layer with standard deviation 0.01.

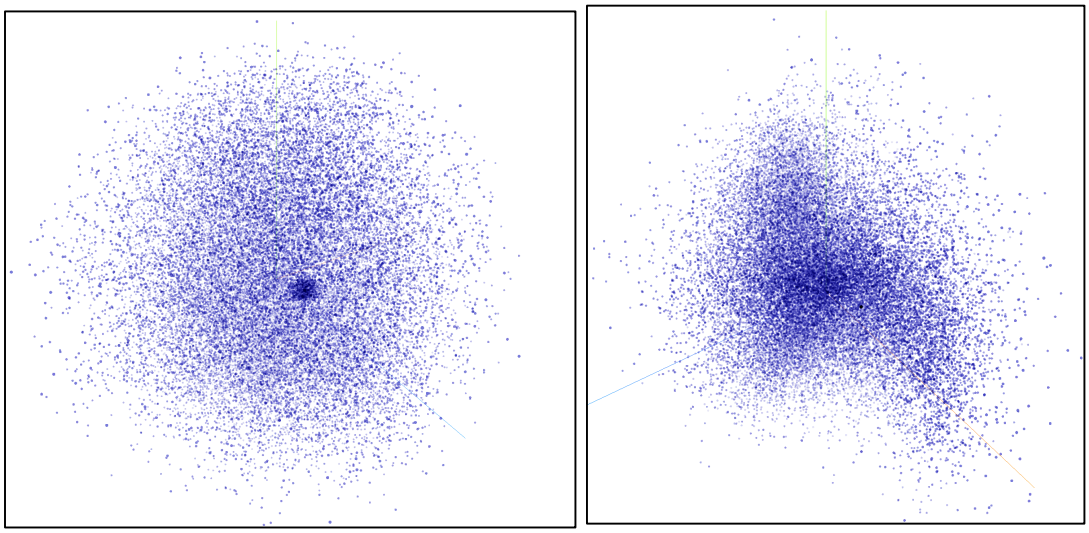

Figure 23 – Left: Projection of a Visualization of the second, third and fourth largest factors in the 300-dimensional User Embeddings generated by a 4-layer neural network. Right: Projection of the first three factors of the 300-dimensional Movie Embeddings.

5. CONCLUSION

5.1. The Research Problem and its Answer

The goal of this work is to Answer our Research Question: "**Can we use Latent Space Embedding and other new Machine Learning techniques to understand user preferences?".** Latent Space Embedding is one of the most popular Machine Learning techniques, and to answer this question, we created a model to solve the problem of User Recommendations. We first presented a background for User Recommendations, Latent Space Embeddings, Machine Learning and how they are used. We developed a Matrix Factorization model to predict movie ratings from the MovieLens database, a standard and freely available dataset assembled by the GroupLens Research group at the University of Minnesota. We successfully implemented the model using the Tensorflow framework, and found that using a form of Adaptive Stochastic Gradient Descent technique with Early Stopping and Regularization. As compared to a baseline, mean squared error (MSE) on our validation data of 0.91 our model could produce an MSE of 0.65 for 10 dimensions. The training time for our model varied from 1 minutes to 1h 35minutes, varying roughly linearly with the number of dimensions used in our model. Implementing this model with Graphical Processing Units reduced processing time in 60% on average. We found that the MSE on our trained data decreased with time

The training process attributed latent space vectors to each user and each movie in the dataset. The inner product of a user's and a movie's vector produced estimates for the ratings that the user would give to that film (minus two bias terms). That process is equivalent to Factor Analysis (FA) and just as FA can discover latent variables that help explain data, this model likewise explains users and films through latent movie characteristics and the corresponding latent user preferences thus giving us a meaningful representation for both film and movies. That representation was tested by verifying the capacity of the model to identify similarities between films through the cosine distance. By choosing films and identifying the films that had the lowest cosine distance to them, we could verify that the model extracted a deeper level of similarities than would be thought possible without information about the content of the films. By using only film scores we managed to find embeddings that placed in close proximity movies from the same series (Harry Potter films, James Bond, original Star Wars, Monty Python); films of the same rough category (musicals from the 50s and 60s); style (Tarantino and Guy Richie). We also tested one case for which we verified if the predicted similarity of two recent German films, *Downfall* and *The Lives of Others*, wasn't derived from

their being rated by the same users. We found that to be unlikely as the number of instances in which those films were rated by the same users was just around 20% of the number of ratings and the correlation of scores was too low for them to be considered most similar to each other (0.33).

We then analyzed the capacity of the model to be used incrementally to predict ratings by totally removing some users from our training set. After training the model we used a variable number of the ratings to estimate the user vector for that user, and then used that vector to estimate the rest of the ratings. Our results showed that MSE increased as the number of ratings used for the estimation approached the number of dimensions of our space and then decreased as the number of samples surpassed the dimensionality of the space. We believe that adjustments in our model could correct for this behavior, but that should be subject to further study, but it may also be a result of the nature of the space thus created for the task, which could be an explanation for the average decrease in performance after the number of dimensions pass 10 in our dataset. This should be subject to further study. Finally, we quickly demonstrated that while Matrix Factorization provides good results in understanding the spaces of user preferences, it's just one of several techniques. For that, we implemented a 4-layer neural network and found that it has a qualitatively similar performance.

Thus, we conclude that we were successful in demonstrating that user embedding can capture semantic information about user preferences through processes that are feasible and cheap.

5.2. Contributions

The fact that latent vectors trained in problems like the user recommendation problem we implemented here can encode various forms of information that are not explicitly provided to the models raise the possibility that such information may be used by different models. Since movie vectors encode a great number of films, it could be possible to train sophisticated models based on those vectors, connecting preference information with other sources. It could be possible, for instance, to relate scripts and dialogues to their acceptability, and use regression methods to forecast techniques. By using user vectors with other information about users, it would also be possible to obtain a deeper understanding of habits and tastes, allowing predictions across areas, like clothes given one's taste in films in ways. Using user vectors to learn about user and their tastes would allow new kinds of inferences that today require direct regression between disparate datasets. We believe that by demonstrating that the vectors learned with our technique encode that latent information, we help open the

way for the use of the whole toolset of Machine Learning to Marketing and User Preference modelling.

5.3. Limitations

We should note that our research has a series of shortcomings that should be taken into account. The first is that the Matrix Factorization technique that we used assumes a linear model in which preferences are strictly addictive and mutually independent. User preferences, though, may be non-linear and different users may have different concavities in their preference functions. Some movie characteristics may interact with each other. We also didn't take into consideration specific user circumstances: user preferences could vary depending on how long it was made after the movie was watched. Our technique provided us with only point estimates of film ratings and didn't track confidence on results. Also, since factoring does not generate an unique solution, we didn't explore the ways in which different solutions cold be better or worse. We focused on the assertion that latent semantic information is encoded in user and movie vectors but we didn't try to decompose those vectors to find that information. Also, the issues we identified while predicting scores from new users mean that our model could not be used without modifications for that task.

5.4. Future Research

A series of paths for future work are suggested by our results. On the model side, it would be interesting to understand the behavior with different activations and structures. An attention model could be used to emphasize important data. Convolutional Neural Network techniques like pooling could be used to allow for more complex interactions between preferences. We could also work on a Bayesian model to produce probability distribution estimates for user and movie vectors and train the model on such probabilities.

Since one of our goals was to produce embedding vectors, it would be interesting to use the vectors we created for other models: to understand tastes, predict classifications, understand tags and genres and compare with user reviews and even posters. By creating tags extra tags we could also train models that would allow us to better understand characteristics about films: directors, styles, moods and series. It would be an interesting exercise to generate descriptions of films based on film vectors. For richer datasets, the same techniques could be used to transfer understanding between different datasets. It would also be interesting to expand this model to user and product vectors to analyze receipts, thus estimating price-elasticity for never-before seen products.

REFERENCES

Adomavicius, G., & Tuzhilin, A. (2005). Personalization Technologies: A PROCESS-ORIENTED PERSPECTIVE. Communications Of The ACM, 48(10), 83-90.

Bartholomew, D. Knott, M., Moustaki, I. (2011). Latent Variable Models and Factor Analysis: A Unified Approach. 3rd Edition. John Wiley and Sons, Ltd. London.

Baxter, W., & Anjyo, K. I. (2006). Latent doodle space. Computer Graphics Forum, 25(3), 477–485. http://doi.org/10.1111/j.1467-8659.2006.00967.x

Brynjolfsson, E. & Simester, D. (2011). Goodbye Pareto Principle, hello long tail. The effect of search costs on the concentration of product sales. Management Science, 57(8), 1373-1386.

Brynjolfsson, E., Yu (Jeffrey), H., & Smith, M. D. (2003). Consumer Surplus in the Digital Economy: Estimating the Value of Increased Product Variety at Online Booksellers. Management Science, (11). 1580.

Deerwester, S. C., Dumais, S. T., Landauer, T. K., Furnas, G. W., Harshman, R. A. (1990). Indexing by latent semantic analysis. Journal of the American Society of Information Science, 41(6):391–407, 1990. Retrieved November 25, 2017 from http://lsa.colorado.edu/papers/JASIS.lsi.90.pdf

Despois, J. (2017 Feb 14). Latent Space Visualization: Deep Learning bits #2. [Weblog post]. Retrieved November 23, 2017 from https://medium.com/@juliendespois/latent-space-visualization-deep-learning-bits-2-bd09a46920df.

Emarketer. (2017, July 13th). A Brief Overview of The Global Ecommerce Market. Retrieved November 23, 2017 from https://retail.emarketer.com/article/brief-overview-of-global-ecommerce-market/59690010ebd40005284d5cc5?ecid=NL1014.

Goh, G. Decoding the Thought Vector. (2017) Blog Post. Available from http://gabgoh.github.io/ThoughtVectors/. Retrieved November 12, 2017.

Häubl, G. & Tafts, V. (2000). Consumer decision making in Online Shopping Environments: The Effects of Interactive Decision Aids. Marketing Science, 19(1), 4-21.

Hinton. G. E. & Salakhutdinov, R. R. (2006). Reducing the Dimensionality of Data with Neural Networks. Science Magazine (313), p. 504-507. DOI: 10.1126/science.1127647. Retrieved Nov 12, 20017 from https://pdfs.semanticscholar.org/7d76/b71b700846901ac4ac119403aa737a285e36.pdf.

Isinkaye F. O. (2015). Recommendation Systems: Principles, Methods and Evaluation. Egyptian Informatics Journal (2015) 16, pp. 261-273. Cairo: Elsevier.

Johnson, M. et al. (2016, November 14). Google's Multilingual Neural Machine Translation System: Enabling Zero-Shot Translation. arXiv: 1611.04558v2[cs.CL]. Retrieved November 12, 2017 from https://arxiv.org/abs/1611.04558.

Jolliffe, I. T. (2002). Principal Component Analysis. New York: Springer.

Koren, Y., Bell, R., & Volinsky, C. (2009). Matrix factorization techniques for recommender systems. Computer, 42(8), 30–37. https://doi.org/10.1109/MC.2009.263

Koren, Y. The BellKor Solution to the Netflix Grand Prize. (2009, August). Retrieved November 27, 2017 from https://www.netflixprize.com/assets/GrandPrize2009_BPC_BellKor.pdf.

Li, S. S., & Karahanna, E. (2015). Online Recommendation Systems in a B2C E-Commerce Context: A Review and Future Directions. Journal Of The Association For Information Systems, 16(2), 72-107.

Maxwell Harper, F., Konstan, J. A. (2015). The MovieLens Datasets: History and Context. ACM Transactions on Interactive Intelligent Systems (TiiS) 5, 4, Article 19 (December 2015), 19 pages. DOI=http://dx.doi.org/10.1145/2827872

Mikolov, T., Yih, W., & Zweig, G. (2013a). Linguistic regularities in continuous space word representations. Proceedings of NAACL-HLT, (June), 746–751.

Mikolov, T., Corrado G., Chen, K. Dean, J. (2013b, September 7). Efficient Estimation of Word Representations in Vector Space. arXiv:1301.13781v3 [cs.CL]. Retrieved November 27, 2017.

Mikolov, T., Sutskever, I. Chen, K., Corrado, G. & Dean J. (2013c, October 1). Distributed Representation of Words and Phrases and Their Compositionality. arXiv:1310.4546 [cs.CL]. Retrieved November 14th, 2017.

Pariser, E. (2011a). Eli Pariser: Beware online filter bubbles [Video File]. Retrieved from https://www.ted.com/talks/eli_pariser_beware_online_filter_bubbles

Pariser, E. (2011b). The filter bubble : what the Internet is hiding from you. New York : Penguin Press, 2011.

Pearson, Karl. (1901). On lines and planes of closest fit to systems of points in space, Philosophical Magazine, Series 6, vol. 2, no. 11, pp. 559-572. Retrieved November 25, 2017 from http://www.stats.org.uk/pca/Pearson1901.pdf.

Piote , Chabbert. (2009). The Pragmatic Theory Solution to the Netflix Grand Prize. (2009, August). Retrieved November 27, 2017 from https://www.netflixprize.com/assets/GrandPrize2009_BPC_PragmaticTheory.pdf

PwC. Global Top 100 Companies by Market Capitalization. (2017, March 31). Retrieved from https://www.pwc.com/gx/en/audit-services/assets/pdf/global-top-100-companies-2017-final.pdf. Retrieved November 21st, 2017.

Salton, G. et al (1975 November). A Vector Space Model For Semantic Indexing. Communications of the ACM. Volume 8, Number 11. Retrieved November 25, 2017 from https://pdfs.semanticscholar.org/4008/d78a584102086f2641bcb0dab51aff0d353b.pdf

Scrapehero.com. (2017, October). How Many Products Does Amazon Sell? Retrieved November 23, 2017 from https://www.scrapehero.com/how-many-products-does-amazon-sell-october-2017/

Schwartz, B., Ward, A., Monterosso, J., Lyubomirsky, S., White, K., & Lehman, D. R. (2002). Maximizing versus satisficing: happiness is a matter of choice. Journal Of Personality And Social Psychology, (5), 1178.

Simon, H. A. (1955). A BEHAVIORAL MODEL OF RATIONAL CHOICE. Quarterly Journal Of Economics, 69(1), 99-118.

Todd, P., & Benbasat, I. (1992). The Use of Information in Decision Making: An Experimental Investigation of the Impact of Computer-Based Decision Aids. MIS Quarterly, 16(3), 373-393.

Töscher, A., Jahrer M. (2009, September) The BigChaos Solution to the Netflix Grand Prize. Retrieved November 27, 2017 from https://www.netflixprize.com/assets/GrandPrize2009_BPC_BigChaos.pdf

Win, X. Xin L. Yihong G. (2003). Document Clustering Based on Non-Negative Matrix Factorization. SIGIR '03. Proceedings of the 26th annual international ACM SIGIR conference on Research and development in information retrieval. Pages 267-273

APPENDIX A – PYTHON CODE FOR MF MODEL

```
with graph.as_default():
   tf_train_data = tf.placeholder(tf.int32, shape=(None, 2))
   tf_train_labels = tf.placeholder(tf.float32, shape=(None, 1))
   tf_lr = tf.placeholder(tf.float32)
   tf_batch_size = tf.cast(tf.shape(tf_train_data)[0], tf.float32)

   tf_count = tf.get_variable("count", dtype=tf.int32, initializer=tf.constant(count))
   if (NUM_FEATURES > 0):
      ones = tf.constant(1., shape=(NUM_FEATURES,1))
      with tf.name_scope('embeddings'):
                  user_embeddings  =  tf.get_variable("user_embeddings",  [NUM_USERS,  NUM_FEATURES],
initializer=tf.random_normal_initializer(0,1*math.sqrt(1/NUM_FEATURES)))
               movie_embeddings  =  tf.get_variable("movie_embeddings",  [NUM_MOVIES,  NUM_FEATURES],
initializer=tf.random_normal_initializer(0,1*math.sqrt(1/NUM_FEATURES)))
         variable_summaries(user_embeddings, 'user_embeddings')
         variable_summaries(movie_embeddings, 'movie_embeddings')

      with tf.name_scope('local_embeddings'):
         tf_user_embeddings = tf.gather(user_embeddings, tf_train_data[:,0])
         tf_movie_embeddings = tf.gather(movie_embeddings, tf_train_data[:,1])
         variable_summaries(tf_user_embeddings, 'tf_user_embeddings')
         variable_summaries(tf_movie_embeddings, 'tf_movie_embeddings')
   else:
      user_embeddings = tf.get_variable("user_embeddings", initializer = tf.constant(0.0))
      movie_embeddings = tf.get_variable("movie_embeddings", initializer = tf.constant(0.0))
   if normalize:
     unorm = tf.sqrt(tf.reduce_sum(tf.square(tf_user_embeddings), 1, keep_dims=True))
     tf_user_embeddings = tf_user_embeddings / unorm
         # There's o need to regularize the biases
   movie_embeddings = tf.abs(movie_embeddings)
   #bias = tf.get_variable("bias", dtype=tf.float32, initializer=tf.constant(3.5))
                     user_bias       =       tf.get_variable("user_bias",       [NUM_USERS,       1],
initializer=tf.random_normal_initializer(0.0))
                  movie_bias      =      tf.get_variable("movie_bias",      [NUM_MOVIES,      1],
initializer=tf.random_normal_initializer(3.1))
   tf_user_bias = tf.gather(user_bias, tf_train_data[:,0])
   tf_movie_bias = tf.gather(movie_bias, tf_train_data[:,1])
   with tf.name_scope("biases"):
      variable_summaries(user_bias, 'user_bias')
      variable_summaries(movie_bias, 'movie_bias')
      variable_summaries(tf_user_bias, 'tf_user_bias')
      variable_summaries(tf_movie_bias, 'tf_movie_bias')

   if (NUM_FEATURES > 0):
      if (use_bias):
         train_prediction = apply_activation(
            tf.matmul(tf.multiply(tf_user_embeddings, tf_movie_embeddings), ones) +
            tf_user_bias + tf_movie_bias)
      else:
         train_prediction = apply_activation(
            tf.matmul(tf.multiply(tf_user_embeddings, tf_movie_embeddings), ones))
```

```
   else:
      tf_movie_bias

   if (clip_output):
      train_prediction = tf.minimum(5.0, tf.maximum(0.5, apply_activation(train_prediction)))
      loga("feature: clipping output")

   if  use_bias:
           loga("feature: using biases")
   else:
           loga("feature: NOT using biases")
   with tf.name_scope('results'):
      error = tf.subtract(train_prediction, tf_train_labels)
      sse = tf.reduce_sum(tf.square(error))
      tf.summary.scalar("sse", sse)
      if (NUM_FEATURES > 0):
         if (use_square):
            loga("feature: using L2 on movie embedding regularization")
            regularization =
               tf.reduce_sum(tf.square(tf_user_embeddings))/NUM_FEATURES/tf_batch_size +
               tf.reduce_sum(tf.square(tf_movie_embeddings))/NUM_FEATURES/tf_batch_size
         else:
            loga("feature: using L1 on movie embedding regularization")
            regularization = tf.reduce_sum(tf.square(tf_user_embeddings))/NUM_FEATURES/tf_batch_size
+ tf.reduce_sum(tf.abs(tf_movie_embeddings))/NUM_FEATURES/tf_batch_size
      else:
         regularization = tf.reduce_sum(tf.square(tf_movie_bias)) +
            tf.reduce_sum(tf.square(tf_user_bias))
tf.reduce_sum(tf.square(tf_user_bias)) * batch_size / NUM_USERS
      loss = sse + lbda * regularization
      tf.summary.scalar("loss", loss)
      mse = sse / tf_batch_size
      tf.summary.scalar("batch_size", tf_batch_size)
      tf.summary.scalar("mse", mse)
   optimizer = tf.train.GradientDescentOptimizer(tf_lr).minimize(loss)
   histogram = tf.histogram_fixed_width(error, [-4.5, 4.5], nbins=10)
   tf.summary.histogram('error', error)
   merged = tf.summary.merge_all()
   saver = tf.train.Saver()
```

APPENDIX B – LISTS OF FILMS WITH HIGH DEGREE OF SIMILARITY (300 DIMENSION LATENT SPACE)

Elite Squad	School of Rock	Beautiful Mind	Good bye, Lenin!
City of God	Toy Story 3	Scent of a Woman	Rabbit-Proof Fence
The Last King of Scotland	Kung Fu Panda	Catch me if you can	The Motorcycle Diaries
Some Folks call it a Sling Blade	Elf	The Green Mile	Amores Perros
Exit through the Gift Shop	The Sandlot	Good Will Hunting	In America
Buena Vista Social Club	The Simpsons Movie	Cinderella Man	My Left Foot
Matchstick Men	Wedding Crashers	Rain Man	The Last King of Scotland
Elite Squad: The Enemy Within	Holes	The Terminal	Monster
The Lives of Others	City Slickers	Finding Neverland	The Lives of Others

Juno	Trainspotting	Being John Malkovich	Eat Pray Love
Little Miss Sunshine	Delicatessen	Adaptation	Uptown Girls
The Last King of Scotland	Amores Perros	Burn After Reading	Sweet Home Alabama
Garden State	City of God	There Will Be Blood	Practical Magic
The Constant Gardener	Reservoir Dogs	Election	Because I said so
500 Days of Summer	Lock, Stock and Two Smoking Barrels	Ghost World	Kate & Leopold
Stranger than Fiction	The Basketball Diaries	Moon	Shall We Dance
Super Size Me	All about my Mother	The Man Who Wasn't there	Three Man and a Little Lady
Hotel Rwanda			

Saving Private Ryan	**Rambo: First Blood**	**Akira (1988)**	**Spirited Away (Chihiro)**	**Modern Times (1936)**
Schindler's List	Predator	Ghost in the Shell (1995)	My Neighbor Totoro	City Lights (1931)
Band of Brothers	Rocky II	For a Few Dollars More	Howl's Moving Castle	The Great Dictator (1940)
Gladiator	Dirty Harry	Princess Mononoke	Laputa: Castle in the Sky	Rashomon (1950)
Braveheart	Rocky III	A Fistful of Dollars	Nausicaä of the Valley of the Wind	The Apartment (1960)
Black Hawk Down	Lethal Weapon	The Professional	Princess Mononoke	On the Waterfront (1954)
Good Will Hunting	Beverly Hills Cop	Amores Perros	Grave of the Fireflies	The Maltese Falcon (1931)
The Patriot	48 Hrs.	Grave of the Firefly	Kiki's Delivery Service	Yojimbo (1961)
The Green Mile	Die Hard 2	Laputa: Castle in the Sky	Triplets of Belleville	Paths of Glory (1957)
Voices from the List	Die Hard	Moon	The Professional	The Gold Rush (1925)
Road to Perdition		Birdman	Porco Rosso	Metropolis (1927)
Cast Away				Bicycle Thieves (1948)
Platoon				Strangers on a Train(1951)
				Rebecca (1940)

Titanic (1997)	**Ghost**	**The good the bad and the ugly**	**Dr. No**
Titanic (1953)	Pretty Woman	For a Few Dollars More	From Russia With Love
Free Willy (1993)	Sleepless in Seattle	A Fistful of dollars	Goldfinger
Ghost (1990)	Mrs Doubtfire	Once upon a time in the west	Thunderball
The Bodyguard (1992)	The Bodyguard	Dirty Harry	Live and Let Die
The Little Mermaid (1989)	Look Who's Talking	Paths of Glory	Diamonds are Forever
Cast Away (2000)	Sister Act	Spartacus	The Spy who loved me
Dante's Peak (1997)	Dirty Dancing	Cape Fear	The Man with the Golden Gun
Pretty Woman (1990)	Three Men and a Baby	Scent of a Woman(1992)	You only live Twice
Homeward Bound: The Incredible Journey (1993)	Disclosure	Awakenings (1990)	For Your Eyes Only
Free Willy 2 (1995)		A Beautiful Mind (2001)	Dirty Harry – Lonely hero
Field of Dreams (1989)		Good Morning Vietnam (1987)	The Guns of Navarone – Small team on a mission
Lion King (1994)		Rain Main (1988)	On Her Majesty's Secret Service
		Murder in the First (1995)	Moonraker
		Just Cause (1995)	The Dirty Dozen – Small team on a mission
		Good Will Hunting (1997)	Enter the Dragon – Lonely hero
		Mystic River (2003)	License to Kill
			Spartacus – Lonely hero
			A Fistful of Dollars – Lonely hero