# Understanding Consumer Preferences Through Latent Spaces

# Entendendo as preferências do consumidor através dos espaços latentes

DOI:10.38152/bjtv4n1-004

Recebimento dos originais:08/12/2020
Aceitação para publicação: 22/01/2021

**André Uratsuka Manoel**
Bacharel, Adm.de Empresas, EAESP/FGV
Instituição de atuação atual: Insite Tecnologia
Endereço completo: Av.República do Líbano, no.331, São Paulo/SP, CEP 04501-000
E-mail: andre@corp.insite.com.br

**Gustavo Corrêa Mirapalheta**
Doutor, Adm.de Empresas, EAESP/FGV
Instituição de atuação atual: EAESP/FGV
Endereço completo: Rua Itapeva 474, 9º andar, Bela Vista, São Paulo/SP, CEP 01332-000
E-mail: gustavo.mirapalheta@fgv.br

**João Luiz Chela**
Pós Doutor, Computação, Unifesp, S.J. dos Campos/SP
Instituição de atuação atual: EAESP/FGV
Endereço completo: Rua Itapeva 474, 9º andar, Bela Vista, São Paulo/SP, CEP 01332-000
E-mail: joao.chela@fgv.br

## ABSTRACT

The Latent Spaces technique has applications in areas such as natural language processing, image recognition and multi-language translation. It permits embedding vectors use. These are vectors in a k-dimensional vector space representing increasingly advanced study objects learning models, forming entirely new basis areas. Those vectors can capture semantic study object features and once trained can be reused in other models, decreasing training time and increasing knowledge transfer. This happens in Google Word2Vec and Facebook FastText pre-trained word vectors set. This work explores Latent Spaces techniques to understand preferences through recommendation mechanism implementation on top of MovieLens dataset from University of Minnesota. From it were extracted a sequence of triples *(userId, movieId, rating)*, representing ratings given by users to particular films. Two k-dimensional Latent Vector Spaces representing film characteristics and the corresponding user preferences were created using *Google Tensorflow*, Machine Learning techniques like SGD and Matrix Factorization. The performance benchmark was the mean-square error on the dimensionality k of the Latent Spaces. The capacity of the Latent Spaces was than evaluated to abstract non-trivial information about films and compared with small cosine distance technique. The system was able to find subjective related similarities that would be tough to code in a straightforward manner. Finally, it was 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 be used in business decision making.

**Keywords**: Latent Spaces, TensorFlow, Machine Learning, Recommendation Systems, User Preferences

## RESUMO

A técnica de Espaços Latentes tem aplicações em áreas como processamento de linguagem natural, reconhecimento de imagem e tradução em vários idiomas. Ela permite a incorporação de vetores de uso. Estes são vetores em um espaço vetorial k-dimensional representando modelos de aprendizagem de objetos de estudo cada vez mais avançados, formando áreas de base inteiramente novas. Esses vetores podem capturar características de objetos de estudo semânticos e uma vez treinados podem ser reutilizados em outros modelos, diminuindo o tempo de treinamento e aumentando a transferência de conhecimento. Isto acontece no Google Word2Vec e no conjunto de vetores de palavras pré-treinados FastText do Facebook. Este trabalho explora as técnicas de Espaços Latentes para entender as preferências através da implementação de mecanismos de recomendação sobre o conjunto de dados MovieLens da Universidade de Minnesota. Dele foi extraída uma seqüência de triplos (userId, filmId, classificação), representando classificações dadas pelos usuários a determinados filmes. Espaços vetoriais latentes bidimensionais representando as características do filme e as correspondentes preferências do usuário foram criados usando o Google Tensorflow, técnicas de aprendizado de máquina como SGD e Matrix Factorization. A referência de desempenho foi o erro de quadrado médio sobre a dimensionalidade k dos Espaços Latentes. A capacidade dos Espaços Latentes foi então avaliada para abstrair informações não triviais sobre filmes e comparada com a técnica de pequena distância co-seno. O sistema foi capaz de encontrar semelhanças subjetivas relacionadas que seriam difíceis de codificar de uma maneira direta. Finalmente, foi explorada uma alternativa para gerar vetores de usuário através de redes neurais. As técnicas aqui expostas apóiam o caso de que técnicas de aprendizado de máquina como Espaços Latentes podem ser usadas na tomada de decisões comerciais.

**Palavras-chave:** Espaços Latentes, TensorFlow, Aprendizagem de Máquina, Sistemas de Recomendação, Preferências do Usuário

## 1 INTRODUCTION

### 1.1 PROBLEM FORMULATION AND GOALS

A growing number of marketing tasks require understanding users and their preferences so as to predict better segmentations, and create products that better fulfill their tastes. This paper presents one particular technique: the use of latent spaces to describe user tastes and target object features as vectors. The vectors that correspond to each user and each object can be discovered through several different techniques and encode a plethora of information and relationships that are not directly evident from the data.One common technique involves the use of factor analysis to discover latent variables for each user, which are then used with clustering techniques to produce segments. We propose that expanding the latent variable approach we can encode not only users but also other objects as factors in very powerful models that extend beyond the prediction of tastes to understand fundamental characteristics of users and objects. That mindset is now common in Machine Learning but has not yet found widespread adoption in other areas. The huge assortment of tools that has been developed to support the Machine Learning boom means that such new techniques are now available for old applications. We will introduce the use of the Machine Learning technique of Latent Space Embeddings – which has found use in areas such as machine translation, image recognition and generation and text generation – to the relatively known problem of user preference modeling. We will apply the widely used technique of Matrix Factorization to demonstrate the Machine Learning Point of View. We will then show that the technique does expose the underlying structure of preferences by finding user and object vectors from the MovieLens databaseof film ratings and qualitatively analyze the generated vectors making use of external semantic information.

## 2 REVIEW OF THE LITERATURE

### 2.1 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 (Bartholomew et al.,p. 12).

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 previously possible.

PCA was introduced by Pierson (1901) and then rediscovered and named by Hotellingin 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 as the use ofnew representations of data through orthogonal transformations such that each new basis vector explains variation inincreasing proportion the lowest its ordinal number, and such that each vector is uncorrelated with all the others. While 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:

$$\boldsymbol{X} = \ \boldsymbol{\Lambda f} + \boldsymbol{e} \quad \text{(2.3.1) (Jolliffe, p. 151),}$$

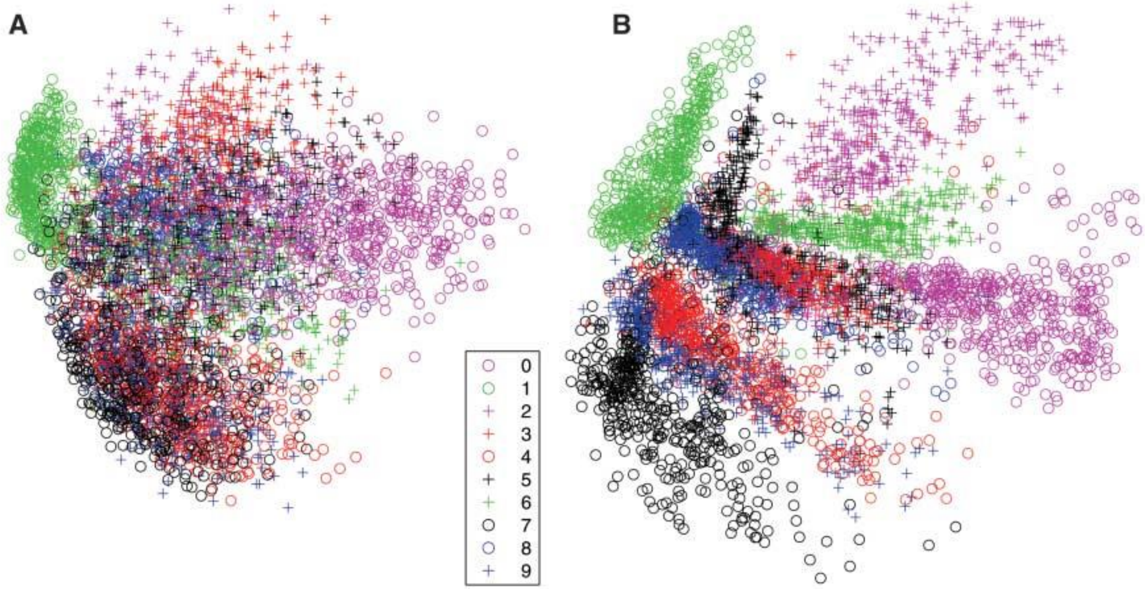
Solving a factor analysis problem, i.e., finding two matrices $\Lambda$ and $f$ that satisfy this formula is equivalent to the problem of matrix factorization.

## 2.2 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 specific 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 techniquethat 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 form the MNIST dataset, the network was capable of finding vectors that placed representations of same digits together, without any previous information on the meaning of each image. OnFigure 1, Hinton and Salakhutdinovcompare the placement of images of handwritten digits on a vector space obtained by using the autoencoder to those obtained witha standard PCA approach.

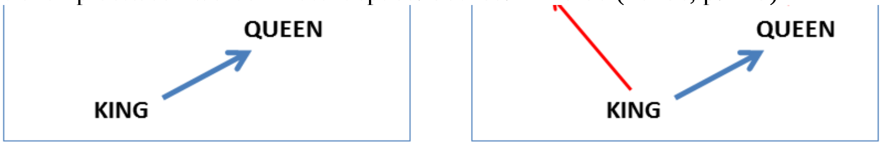
Figure 1 - Comparison of results from classification of digit images with PCA(A) and with Autoencoders(B). Source: Hinton &Salakhutdinov (2006, p. 506)



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.

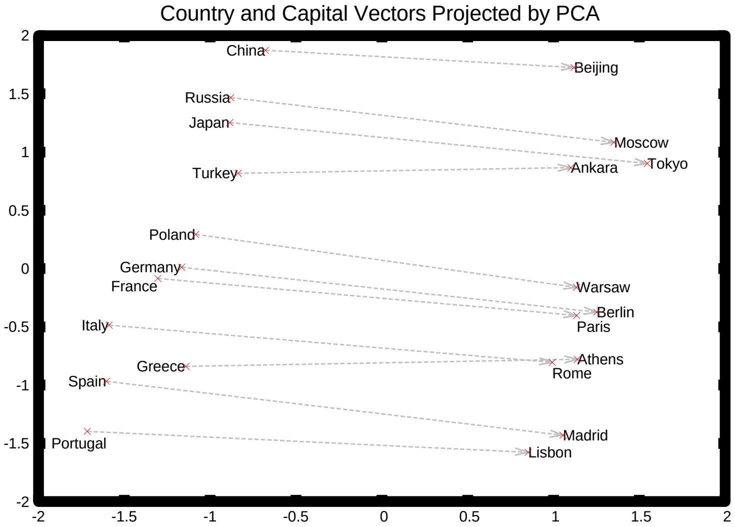
Probably the most successful use of Latent Space Embeddings comes from the work of Mikolov (2013a, p. 749), where a vector representation of words was created with a recurrent neural network model whose goal was to, given a word, predict words that would surround it in a large corpus of text. It was found that the vector representations exhibited a structure through which some 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”, thus capturing the concept of gender. 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, as demonstrated on Figure 2.

Figure 2 - 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)



Mikolov et al.  (2013c) went further and used a more efficient model to train a larger corpus, demonstrating that the embeddings were capable of capturingmore complex semantic relationships, like the relationship between countries and capitals demonstrated on Figure 3.

Figure 3 - 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, especially 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 fundamental 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, but also hinted at the possibility of capturing meaning above languages.

### 2.3 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 then combined. One of the most important techniques, described in Koren;Bell; Volinsky(2009) and followed in the work was the Matrix Factorization technique. This technique is equivalent to finding latent factors in a Factor Analysis model. 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(Maxwell et al., 2015), 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 isto create a model that would predict the rating that a given user would likely give to a film they have not yet seen while using a Latent Space Embedding to attribute a vector to each user and to each movie whichcan represent preferences and features, thus capturing semantic information about the films, like genre and mood and about particular 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 ratings 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 chosen because that technique is easy to understand with standard statistical knowledge, quick to train, and can be performed 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 report on section 0.

The Matrix Factorization technique works bychoosing 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:

$$\boldsymbol{R'} = \boldsymbol{UM},\text{or}\boldsymbol{R'}_{\boldsymbol{ij}} = \boldsymbol{u_i} \cdot \boldsymbol{m_j}$$

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 iand 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:

$$(1)\boldsymbol{R'} = \boldsymbol{UM} + \boldsymbol{B^u} + \boldsymbol{B^m} \qquad (2)$$

### 3.2 TRAINING PROCESS

While factorizing a matrix is easy with standard methods like Eigenvectors decomposition, SVD or LU factorization. To apply a standard SVD to the data, it would be necessary to impute missing data with zeros or averages, which would add errors to our estimates. Instead, we use Gradient Descent (GD) to minimize a loss function. 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$\dot{U}$of the user vectors for this batch as rows, $\dot{M}$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:

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

We then proceed to minimize the Square Error of the model:

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

After each step, we calculate the gradient, i.e, the vector containing all the partial derivativesof 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:

$$\boldsymbol{\dot{\theta}_\iota \leftarrow \dot{\theta}_\iota - \alpha \frac{\partial loss}{\partial \dot{\theta}_\iota}}$$

where$\dot{\theta}_\iota$ corresponds to each parameter in our model, and $\alpha$ 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 have completed an epoch, and the results were evaluated and then repeated or stopped.

### 3.3 AVOIDING OVERFITTING

The first technique we used to avoid overfittingwas the use of regularization, in which we augment our loss function, i.e., the function we intend to minimize with a regularization parameter:

$$\boldsymbol{Loss \ = \ SqError \ + \lambda reg}$$

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

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

where k is the number of dimensions in our model, and b is the number of ratings in each batch.

A second technique that we used was early stopping: we divided data into a training set and a validation set.We trained data using only the training set, then, after each epoch, we used the learned parameters to calculate the loss on the validation set. When the validation-set loss stopped decreasing, we interruptedtraining, 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,our gradient descent process may find itself in local minima. To avoid this problem, we used a modified version of the technique called Mini-batch Stochastic Gradient Descent (SGD). Our implementation combined a standard SGD with rule-based changes in the learning rate $\alpha$ 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 $\alpha$ and $b$.

### 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.

For our model, we deviated from the matrix factorization model to the neural network model by applying the following activation function to the result of the matrix multiplication:

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

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.

## 3.6 EMBEDDING SPACE DIMENSIONALITY

The most important parameter in our 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 trainedthe 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 TENSORFLOWIMPLEMENTATION OF MODEL

The model was implemented using the Tensorflow framework in Python. Tensorflow is a ML 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 increasedour processing speed by 100 to 1000 percent. The model, is reproduced on the **Erro! Fonte de referência não encontrada.** and is represented visually in **Erro! Fonte de referência não encontrada.**, which was generated by the Tensorboard tool from the actual model used.

## 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:

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

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 0.

## 4 DATA ANALYSIS

### 4.1 DESCRIPTION OF THE DATA

For this work we used the MovieLensLatestDataset, downloaded inOctober 2016. This dataset is the result of the GroupLens project of the University of Minnesota. That group runs an online application thatrecommends films to users and collects ratings given by users to those films. The dataset contained22,884,377 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 also contains tags attributed by users to films, and their classification. Given our goals, we usedjust the ratings,contained in the file **ratings.csv.**

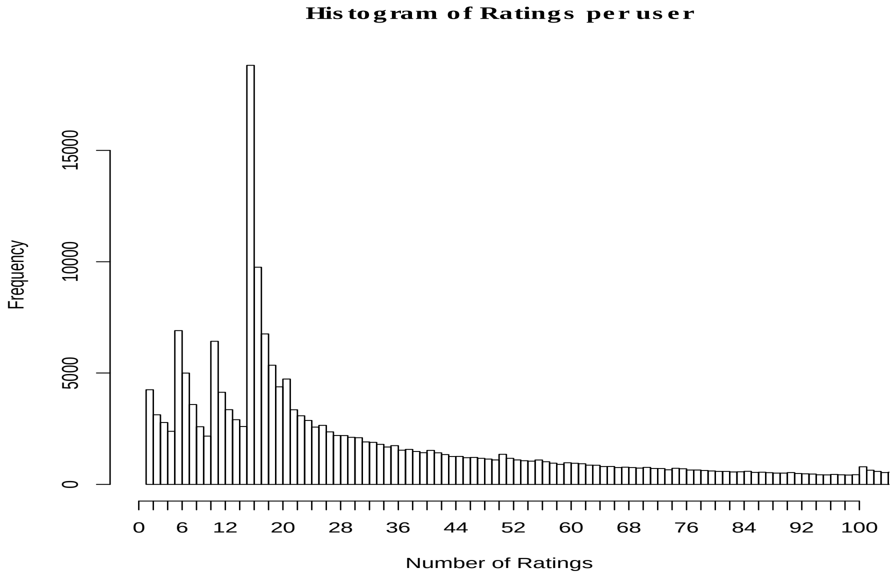
For reference we later used the names of the films, contained in the file **movies.csv** to analyze the behavior of the model.

### 4.1.1 Ratings

The ratings table was the only data used during the Matrix Factorization process to train our model. It consists on the rating from 0.5 to 5.0 given by users, identified only by numbers, to movies, also identified by numbers. 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. The ratings had a mean of 3.526 and standard deviation of 1.061.

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. The distribution of ratings also presented a long tail, but modes on 1, 5, 10 and specially 15 samples, as demonstrated on Figure 4. We attribute that strange distribution to the logistics of obtaining data from users.

Figure 4 - Histogram of Ratings per User



## 4.2 DATA PREPARATION

Our first step while working with data was to clean up the data set, removing movies with no ratings and renumberingmovieIds 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 1.

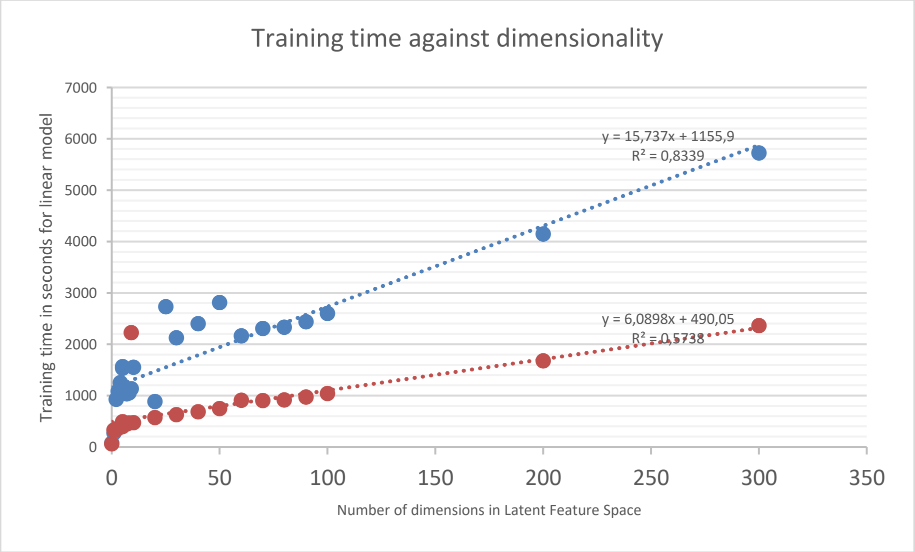
Table 1 - Distribution of data among the three sets used for training, validation and testing

| **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% |

## 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 (which is a mere calculation of bias and thus sample average), 4 minutes for one dimension, and 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 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 onFigure 5. 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 5 - 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 time complexity as the number of dimensions increased and more than doublingof speed when calculated with GPUs.We detected that the performance decreased quickly as batch sizes were reduced, which is why we decided to adopt an adaptive model where the learning rate and batch size were changeable. 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, which 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, to 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. That variable has mean -0.073 and standard deviation 0.921; although at first sign it appeared to follow aGaussian distribution, it fails the normality tests with a D'Agostino and Pearson p-value of 0.000.

Increasing the number of dimensions, we find a multi-variate distribution that follows the same general shape as the 1-D version. For 300 dimensions, we find that the average user vector has a mean on each direction of $0.05774 \pm 0.0032$. It also fails tests of normality.

## 4.5 MOVIE EMBEDDINGS

The analysis of the movie embeddings is easier than the analysis of users embeddings since films are not anonymous so that we could use external knowledge about the films understand their relationship.For each characteristic x that might interfere with one's taste for films, a well-trained model with enough dimensions would codify x as a direction in that latent vector space.Such behavior is more noticeable in high-dimensional spaces.

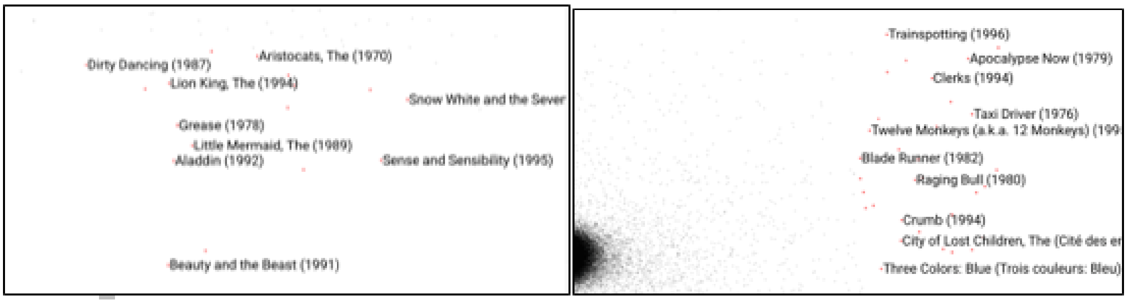
In the 300-dimensional space, and using PCA to find the three most important directions in this space, the films formed the shape of a pyramid or a comet, with the head representing films with a low number of ratings (most often less than 100) while famous films are on the sparser part of the space. 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.

Using just the second and third PCA directions, and turning them into a 2-D map, we selected some films with similar second and third components. We show on Figure 6(first panel), a series of films on the bottom of the map (small second component, negative third componentthat possess a lighter mood than films on the second panel (high second component, medium to high third component).

Another way to demonstrate the encoding of user preferences in the vector space is the comparison of films through their vector's cosine distance. Our first experiment was to find the films that were most similar to "**The Sound of Music**, " a classic musical from 1965, shown. 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). All of those films are family films more than fifty years old, which may reflect an existing split in the movie market between generations.

The ability of the technique to discover similarities between films based on nothing more than the ratings given by users films becomes more pronounced as the number of dimensions increase. With 3 dimensions, the first few films with high similarity wouldn't be considered similar by an unbiased observer. With 10 dimensions, the films seem to start to cater to a similar public. Moving to 300 dimensions, the model becomes capable of perceiving the kind of similarities that humans would perceive. Some examples are available on APPENDIX C – FILM SIMILARITY RESULTS.

Figure 6 - Selection of films in the 2-D map using the same projection. On each panel the “head” of the cometis 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.

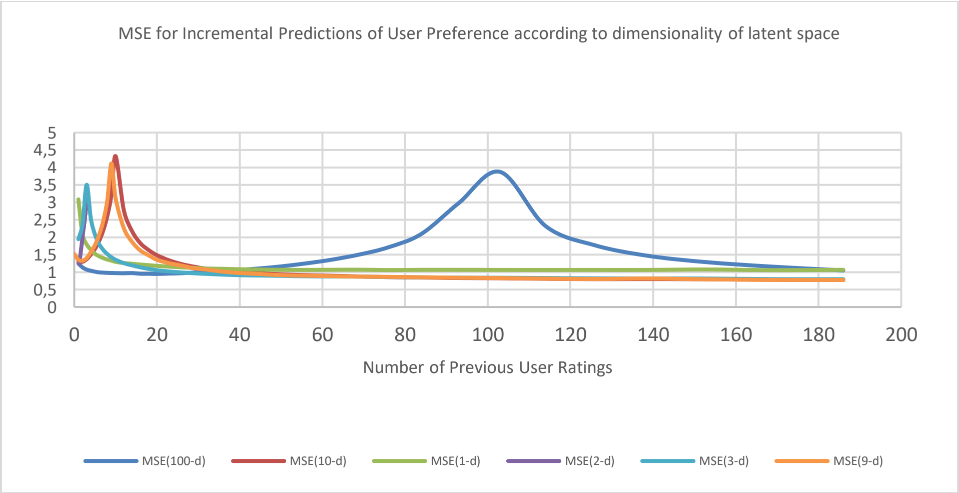


## 4.6 INCREMENTAL PREDICTION

Our next step was to check if the matrix factorization model could be used to predict ratings for previously unseen users. Wehave trained both user and movie vectors at the same time. When a previously unknown user is presented with ratings related to known movies, we should be able to estimate the user vector and thus predict ratings that the user would give to other film. Similarly, given a new film, a few ratings by users with known vectors would allow us to estimate the score that other users would give it. We used theRoger-Penrose pseudo-inverse to provide those predictions. We found that our results were useful but, without controlling for the quality of vectors used for prediction, the mean squared error is sensitive to the number of samples used, with a peak when the number of samples is equal to the dimensionality of the vector space as can be seen on Figure 7.

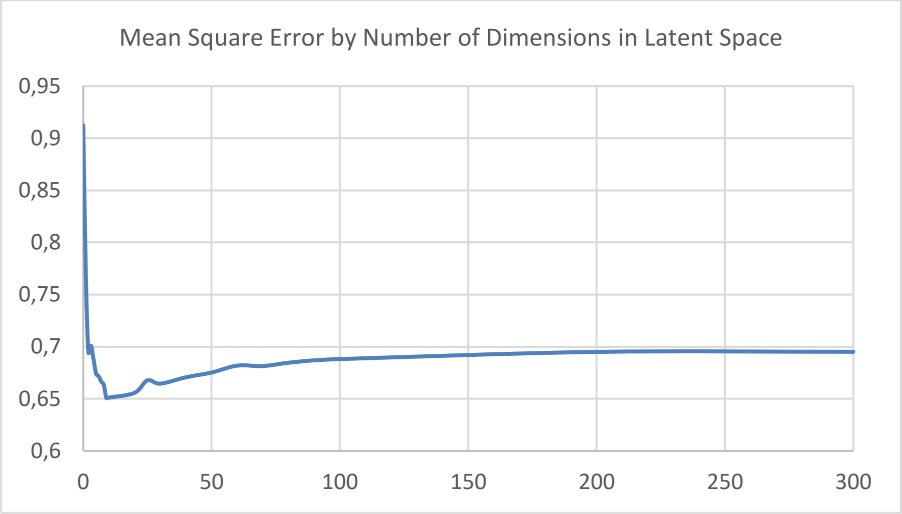
## 4.7 EFFECTS OF THE DIMENSIONALITY OF THE LATENT SPACE

Figure 7 - MSE of predictions about user film preferences after user vector is estimated from varying number of scores for different dimensionalities.



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. 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, with 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 overfitting.

Figure 8 - Mean Square Error on Test Set by Number of Dimensions in Latent Space



## 4.8 HANDLING SAMPLE BIAS

User vectors allow us to estimate the scores that users would give films if they had seen them. The ratings that we collectare usually biased as different groups of users may have different probabilities of rating a film. 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 user vectors of the population.

## 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 layerneural network with the Keras library, whose structure isavailableon

APPENDIX B – TENSORFLOW MODELS and trained it with the same data as the Matrix Factorization model. This model usedReLU activation between layers and, instead of regularization, used Gaussian Noise to reduce the possibility of overfitting. Despite being very different and not having been through any meaningful optimization effort, the results we obtained were similar to the matrix factorization method, achieving just slightly higher MSE, and also sharing the same properties,of allowing us to identify similar films by a simple cosine similarity measure.

## 5 CONCLUSION

### 5.1 THE RESEARCH PROBLEM AND ITS ANSWER

We developed a Matrix Factorization model to predict movie ratingsfrom the MovieLensdatabase. We successfully implemented the model using the Tensorflow framework, and found that using a form of Adaptive Stochastic Gradient Descent with Early Stopping and Regularization, our model could produce an MSE of 0.65 for 10 dimensions, compared to our baseline of mean squared error (MSE) of 0.91. 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.

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, this model likewise explains users and films through latent movie features and the corresponding latent user preferences thus giving us a meaningful representation for both users and movies.That representation was tested by verifying the capacity of the model to identify similarities between films through the cosine distance metric. By identifying the films with low cosine distance betweenthem, we qualitatively verifiedthat the model extracteda deeper level of meaning than would be expected without information about the content of the films.

We then analyzed the capacity of the model to incrementallypredict ratings by applying a simple pseudo-inverse calculation to predict user predictions given just partial data.

Finally,we implemented a 3 hidden layer neural networkand found that it has a qualitatively similar performance, quickly demonstratingthat Matrix Factorization is just one of several possible techniques to extract Latent Semantic Spaces.

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

## 5.2 CONTRIBUTIONS

The fact that through latent vectors trained in problems like the User Recommendation problem implemented here, wecan learn deep semantic data. Since movie vectors encode a great number of features, 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 to movie vectors to study their results;or relate user vectors to other preferences, thus inferring the taste in clothes or food based on the taste in films.We believe that by demonstrating that the vectors learned with our technique encode such latent information, we help open the way for the use of the whole toolset of Machine Learning to whole new areas.

## 5.3 LIMITATIONS

This research has some limitations. 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 also be non-separable. We also didn’t take into consideration time-specific circumstances. Our technique provided us with only point estimates of film ratings and didn’t track confidence on results. Also, betterpre-processing would be required to deal with the overfitting problems described in 0.

## 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 like attention and pooling.We could also work on a Bayesian model to keep track of confidence estimates.

Since one of our goals was to produce embedding vectors, it would be interesting to use the vectors we created for models that would connect that information to more that, predict classifications, understand tags and genres, understand movie reviews and video, the importance of directors, styles, moods, etc. It would be an interesting exercise to automatically generate descriptions of films based on film vectors.

Outside of the specific problem of user modeling, it would also be interesting totry this same model in other realms, like the modeling of retail purchases, thus estimating price-elasticity for never-before seen products,or to the analysis of market and financial data to better understand assets and their pricing.

## REFERENCES

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

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 fromhttp://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.

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

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.

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.

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.

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

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

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 – MSE ACCORDING TO DIMENSIONALITY OF FEATURE SPACE

Table 2 - Prediction error according to dimensions offeature space and number of samples.

| Samples | Dimensionality of Feature Space | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 6 | 8 | 9 | 10 | 20 | 40 | 100 |
| 1 | 3,086 | 2,253 | 1,944 | 1,489 | 1,406 | 1,340 | 1,357 | 1,309 | 1,314 | 1,327 | 1,248 |
| 2 | 2,071 | 3,266 | 2,361 | 1,764 | 1,478 | 1,349 | 1,330 | 1,298 | 1,223 | 1,195 | 1,137 |
| 3 | 1,796 | 2,255 | 3,504 | 2,303 | 1,702 | 1,464 | 1,404 | 1,371 | 1,207 | 1,140 | 1,073 |
| 4 | 1,646 | 1,766 | 2,542 | 3,486 | 2,088 | 1,654 | 1,574 | 1,502 | 1,233 | 1,136 | 1,041 |
| 5 | 1,528 | 1,542 | 2,081 | 2,529 | 2,714 | 1,912 | 1,774 | 1,662 | 1,271 | 1,125 | 1,014 |
| 6 | 1,455 | 1,380 | 1,809 | 2,077 | 3,859 | 2,302 | 2,060 | 1,888 | 1,312 | 1,132 | 0,994 |
| 7 | 1,404 | 1,289 | 1,638 | 1,812 | 2,842 | 2,935 | 2,461 | 2,191 | 1,374 | 1,151 | 0,986 |
| 8 | 1,361 | 1,232 | 1,519 | 1,647 | 2,359 | 4,065 | 3,062 | 2,610 | 1,448 | 1,153 | 0,979 |
| 9 | 1,331 | 1,195 | 1,426 | 1,530 | 2,065 | 3,037 | 4,117 | 3,231 | 1,523 | 1,177 | 0,975 |
| 10 | 1,295 | 1,160 | 1,355 | 1,435 | 1,852 | 2,501 | 3,118 | 4,324 | 1,616 | 1,202 | 0,972 |
| 12 | 1,261 | 1,111 | 1,257 | 1,309 | 1,595 | 1,937 | 2,256 | 2,770 | 1,852 | 1,252 | 0,968 |
| 14 | 1,240 | 1,077 | 1,194 | 1,236 | 1,457 | 1,647 | 1,865 | 2,170 | 2,179 | 1,321 | 0,973 |
| 16 | 1,218 | 1,028 | 1,138 | 1,157 | 1,327 | 1,453 | 1,618 | 1,828 | 2,600 | 1,351 | 0,954 |
| 18 | 1,199 | 0,994 | 1,093 | 1,113 | 1,249 | 1,330 | 1,480 | 1,633 | 3,301 | 1,428 | 0,953 |
| 20 | 1,179 | 0,961 | 1,054 | 1,064 | 1,187 | 1,241 | 1,354 | 1,484 | 4,813 | 1,485 | 0,947 |
| 23 | 1,156 | 0,946 | 1,018 | 1,026 | 1,123 | 1,164 | 1,255 | 1,336 | 3,003 | 1,661 | 0,959 |
| 26 | 1,138 | 0,929 | 0,990 | 0,996 | 1,075 | 1,101 | 1,185 | 1,229 | 2,294 | 1,837 | 0,969 |
| 29 | 1,122 | 0,921 | 0,967 | 0,972 | 1,043 | 1,045 | 1,110 | 1,155 | 1,932 | 2,066 | 0,989 |
| 32 | 1,103 | 0,907 | 0,949 | 0,949 | 1,015 | 1,010 | 1,073 | 1,097 | 1,709 | 2,388 | 1,003 |
| 36 | 1,095 | 0,893 | 0,930 | 0,932 | 0,988 | 0,977 | 1,014 | 1,045 | 1,499 | 3,158 | 1,031 |
| 40 | 1,083 | 0,887 | 0,912 | 0,910 | 0,963 | 0,948 | 0,973 | 1,000 | 1,362 | 5,210 | 1,064 |
| 45 | 1,073 | 0,876 | 0,904 | 0,899 | 0,945 | 0,926 | 0,946 | 0,968 | 1,242 | 3,035 | 1,108 |
| 50 | 1,067 | 0,881 | 0,893 | 0,885 | 0,924 | 0,906 | 0,913 | 0,937 | 1,154 | 2,328 | 1,164 |
| 56 | 1,064 | 0,883 | 0,881 | 0,875 | 0,909 | 0,886 | 0,899 | 0,912 | 1,080 | 1,902 | 1,250 |
| 62 | 1,068 | 0,891 | 0,876 | 0,867 | 0,896 | 0,874 | 0,884 | 0,896 | 1,032 | 1,668 | 1,351 |
| 69 | 1,069 | 0,886 | 0,870 | 0,860 | 0,885 | 0,861 | 0,872 | 0,877 | 0,980 | 1,463 | 1,505 |
| 76 | 1,060 | 0,873 | 0,858 | 0,848 | 0,869 | 0,846 | 0,864 | 0,857 | 0,943 | 1,335 | 1,713 |
| 84 | 1,065 | 0,861 | 0,851 | 0,846 | 0,866 | 0,833 | 0,852 | 0,845 | 0,911 | 1,211 | 2,099 |
| 93 | 1,066 | 0,871 | 0,843 | 0,836 | 0,853 | 0,824 | 0,840 | 0,831 | 0,878 | 1,121 | 2,981 |
| 103 | 1,064 | 0,875 | 0,839 | 0,833 | 0,852 | 0,820 | 0,830 | 0,824 | 0,860 | 1,067 | 3,873 |
| 114 | 1,063 | 0,857 | 0,831 | 0,827 | 0,840 | 0,812 | 0,815 | 0,810 | 0,839 | 0,999 | 2,320 |
| 126 | 1,063 | 0,848 | 0,821 | 0,818 | 0,826 | 0,798 | 0,803 | 0,799 | 0,814 | 0,945 | 1,767 |
| 139 | 1,063 | 0,868 | 0,820 | 0,813 | 0,827 | 0,797 | 0,818 | 0,801 | 0,804 | 0,933 | 1,464 |
| 153 | 1,077 | 0,849 | 0,817 | 0,813 | 0,828 | 0,794 | 0,795 | 0,796 | 0,796 | 0,917 | 1,285 |
| 169 | 1,057 | 0,850 | 0,799 | 0,796 | 0,807 | 0,781 | 0,778 | 0,780 | 0,775 | 0,866 | 1,156 |
| 186 | 1,066 | 0,855 | 0,797 | 0,793 | 0,807 | 0,775 | 0,776 | 0,773 | 0,765 | 0,844 | 1,050 |
| 205 | 1,047 | 0,861 | 0,787 | 0,785 | 0,794 | 0,765 | 0,790 | 0,765 | 0,750 | 0,829 | 0,982 |
| 226 | 1,066 | 0,832 | 0,786 | 0,786 | 0,790 | 0,763 | 0,776 | 0,761 | 0,748 | 0,812 | 0,939 |
| 249 | 1,064 | 0,808 | 0,778 | 0,777 | 0,783 | 0,756 | 0,758 | 0,753 | 0,735 | 0,793 | 0,889 |
| 274 | 1,063 | 0,820 | 0,773 | 0,775 | 0,778 | 0,752 | 0,768 | 0,750 | 0,734 | 0,793 | 0,859 |
| 302 | 1,057 | 0,795 | 0,760 | 0,760 | 0,769 | 0,738 | 0,740 | 0,738 | 0,722 | 0,758 | 0,824 |
| 333 | 1,063 | 0,795 | 0,763 | 0,758 | 0,762 | 0,738 | 0,744 | 0,735 | 0,719 | 0,766 | 0,807 |
| 367 | 1,061 | 0,801 | 0,749 | 0,750 | 0,751 | 0,735 | 0,735 | 0,732 | 0,710 | 0,745 | 0,790 |

# APPENDIX B – TENSORFLOW MODELS

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.

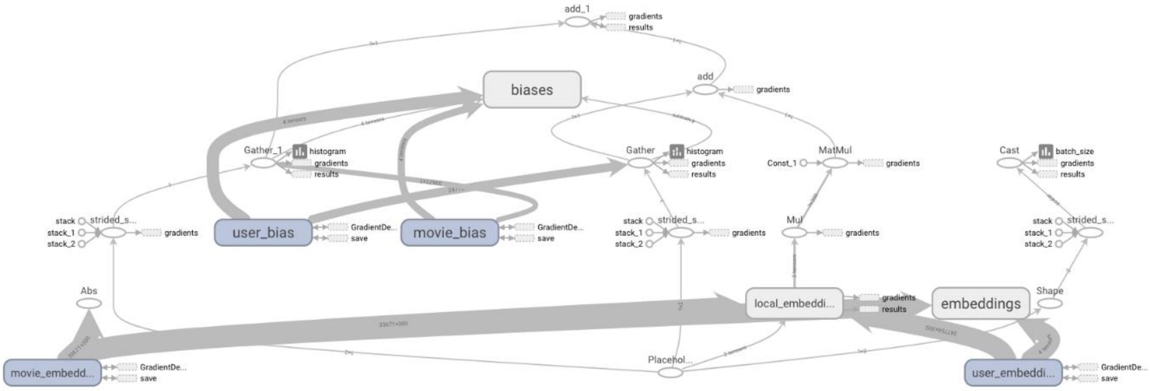
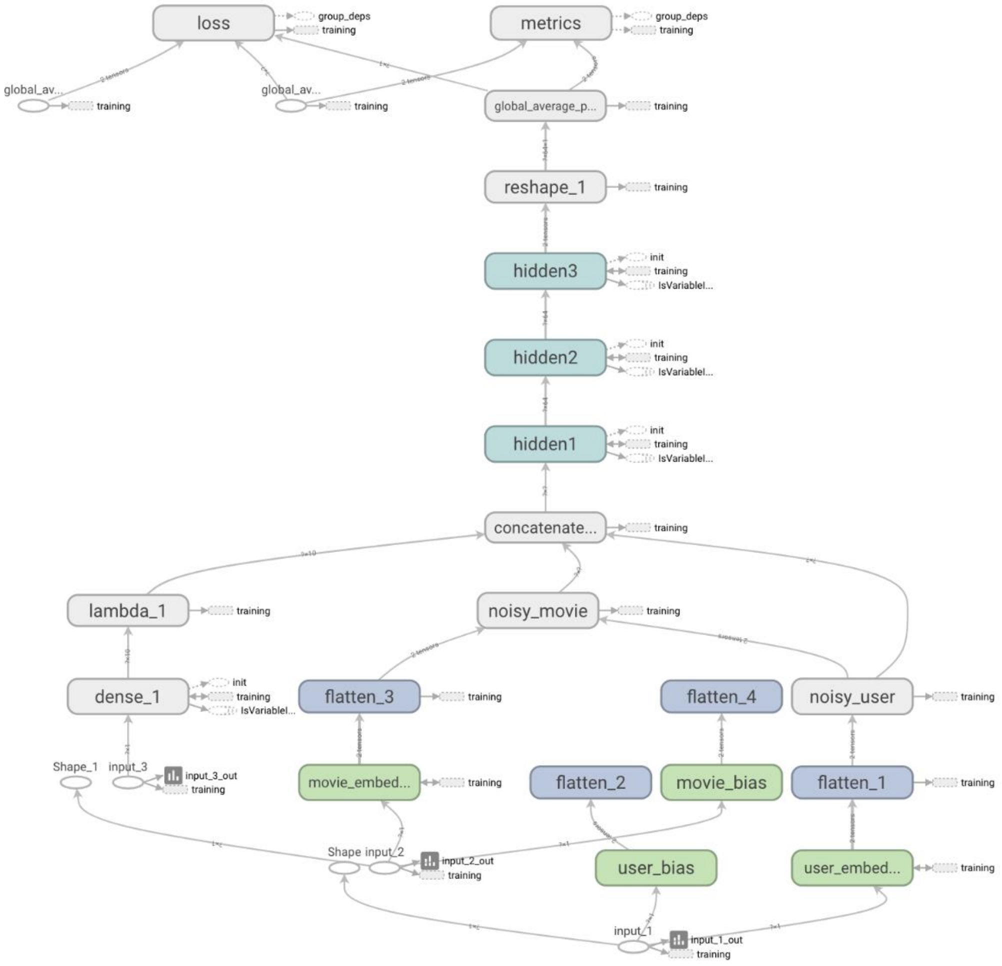


Figure 10 - Alternative Model for obtaining Film and User Embeddings using a standard 4-layer neural network with relu activations and regularization through a Gaussian Noise Layer with standard deviation 0.01.



## APPENDIX C – FILM SIMILARITY RESULTS

Table3–Example of most similar films according to cosine distance for 300-dimension vector spaces.

| **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 | AmoresPerros | 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: Empathy / Emotion / Coming of Age** |

Table 4 - Films most similar to Bergman's The Seventh Seal

| **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 |

Figure 11 - 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.

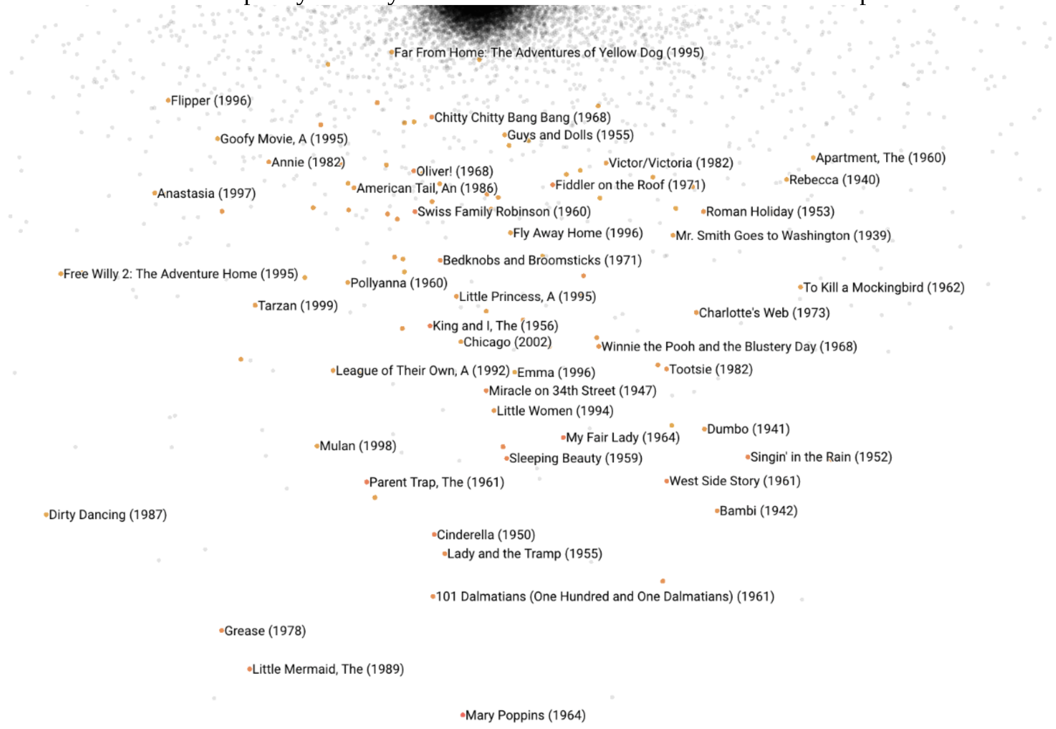


Table 5 - Films most similar to Star Wars Episode IV, by latent space dimensionality

| 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 |