Recommendation System using Deep Learning

Motivation

Business Problem

Amazon customers in the market of digital music have up to 460,000 products to choose from. This many options often lead customers to confusion on which product will best fit their needs.

Objective

Develop a recommendation engine which facilitates this selection criteria

Data Exploration

Number of Users: 840,000

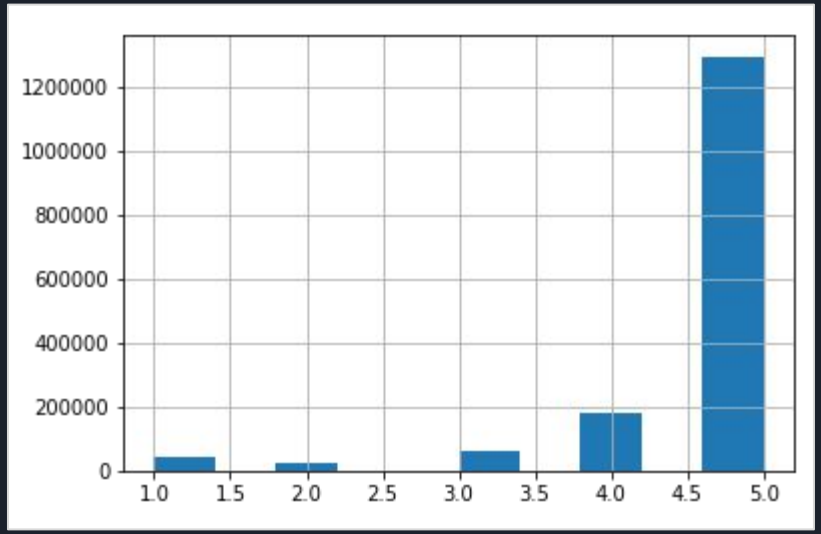
Number of Items: 460,000

Number of Ratings: 1.6 Million

Columns: Rating, ProductID, Title, Brand,

AlsoBought, ReviewerID,

ReviewerName, ReviewText



Methodology

Content Filtering

This Approach relies on the similarity of the items being recommended.

The basic idea is that if you like an item, then you will also like a “similar” item.

Collaborative Filtering

This Approach relies on the similarity of the users it recommending to.

The basic idea is that if someone who is similar to you likes an item, then you will also like that same item.

Typical Challenges for Collaborative Filtering Recommender

Popularity bias

Refers to system recommends the movies with the most interactions without any personalization

Scalability issue

Refers to lack of the ability to scale to much larger sets of data when more and more users and movies added into our database

Item cold-start problem

Refers to when movies added to the catalogue have either none or very little interactions while recommender rely on the movie's interactions to make recommendations

Data Sparsity problem

What ML algorithms can be trained and reliable to make inference when you have an extremely sparse matrix with 99% of values missing

Methodology

Step 1 Embeddings (matrix factorization)
- User Embeddings
- Item Embeddings
- Text Embeddings

Step 2 Merging of Embeddings
- Dot product
- Concatenation

Step 3 Neural Network
- Implicit Feedback Model
- Explicit Feedback Model
- Multi-Layer Perceptron
- Word2Vec Multi-Layer Perceptron

Methodology

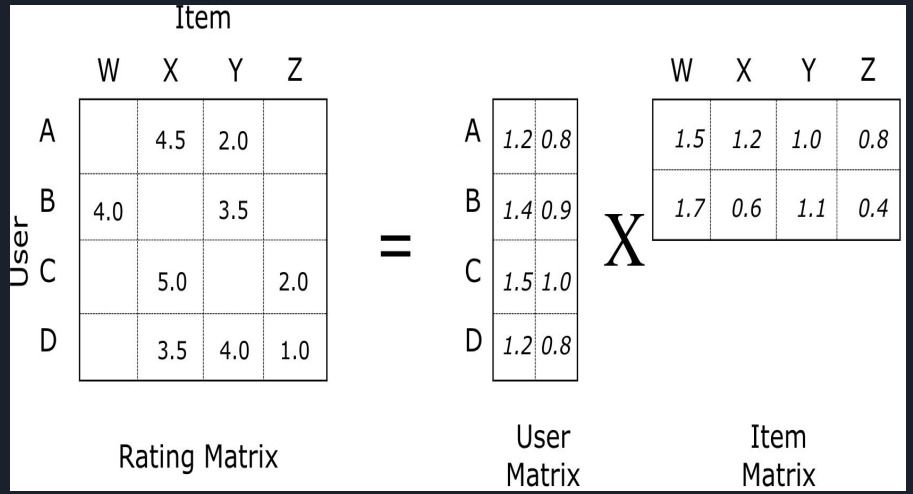
Neural Networks

- Implicit Feedback Model
  - Hinge, BPR, Pointwise
- Explicit Feedback Model
  - Generalized Matrix Factorization Model
    - ➢ Regression(MSE)
    - ➢ Logistic Regression
  - Word2Vec Multi-Layer Perceptron
  - Multi-Layer Perceptron

Matrix Factorization

A family of Mathematical operations to factorize a matrix into a product of matrices.

In Collaborative Filtering, Matrix factorization algorithms work by decomposing the user-item interaction matrix into the product of two lower dimensionality rectangular matrices.



Matrix Factorization

In the sparse user-item interaction matrix, the predicted rating user u will give item i is computed as:

$$\tilde{r}_{ui} = \sum_{f=0}^{nfactors} H_{u,f} W_{f,i}$$

Rating of item i given by user u can be expressed as a dot product of the user latent vector and the item latent vector.

The idea behind matrix factorization is to use latent factors to represent user preferences in a much lower dimension space.

How does Matrix Factorization solve our problems?

Model learns to factorize rating matrix into user and music representations, which allows model to predict better personalized music ratings for users.

With matrix factorization, less-known music tracks can have rich latent representations as much as popular ones, which improves recommender's ability to recommend less-known tracks.

Models

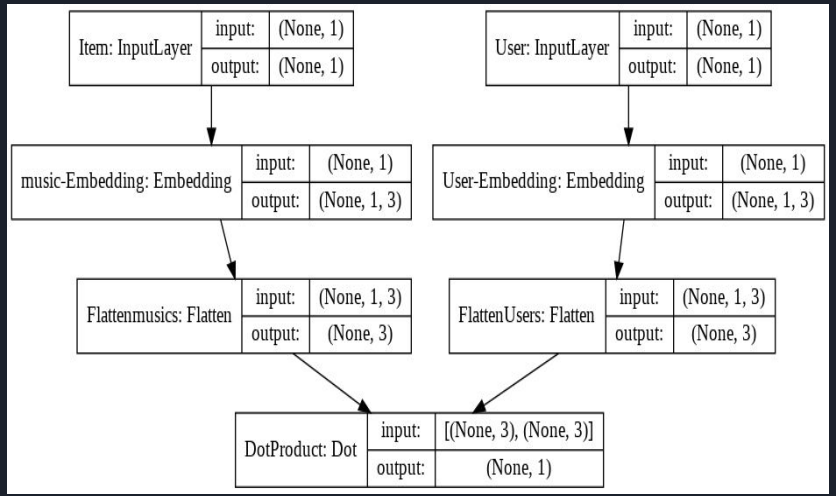
Generalized Matrix Factorization Model(1)

Key Parameters

Deep Learning Framework = Keras
# of iterations = 20
Loss function = MSE
Optimizer = Adam

Results

Train RMSE = 4.352
Test RMSE = 4.643



Generalized Matrix Factorization Model Architecture

Bilinear Factorization Representation

- dot product of the item and user latent vectors.
- 4 embedding layers
  - (num_users x latent_dim) embedding layer to represent users
  - (num_items x latent_dim) embedding layer to represent items
  - (num_users x 1) embedding layer to represent user biases
  - (num_items x 1) embedding layer to represent item biases

Optimizer

Stochastic Gradient Descent

Train/Test Split

Random 80:20 Split

Generalized Matrix Factorization Model (2)

Key Parameters

Deep Learning Framework = PyTorch

# of iterations = 20

Loss function = Regression

Results

Train RMSE = 2.507

Test RMSE = 2.490

Generalized Matrix Factorization Model (3)

Key Parameters

Deep Learning Framework = PyTorch

# of iterations = 20

Loss function = Logistic

Results

Train RMSE = 3.756

Test RMSE = 3.907

Non-Negative Matrix Factorization Model

Key Elements

Non-negative embedding constraint

Key Parameters

# of iterations = 80+
Loss function = MSE

Results

Train RMSE = 0.1809

Test RMSE = 4.951

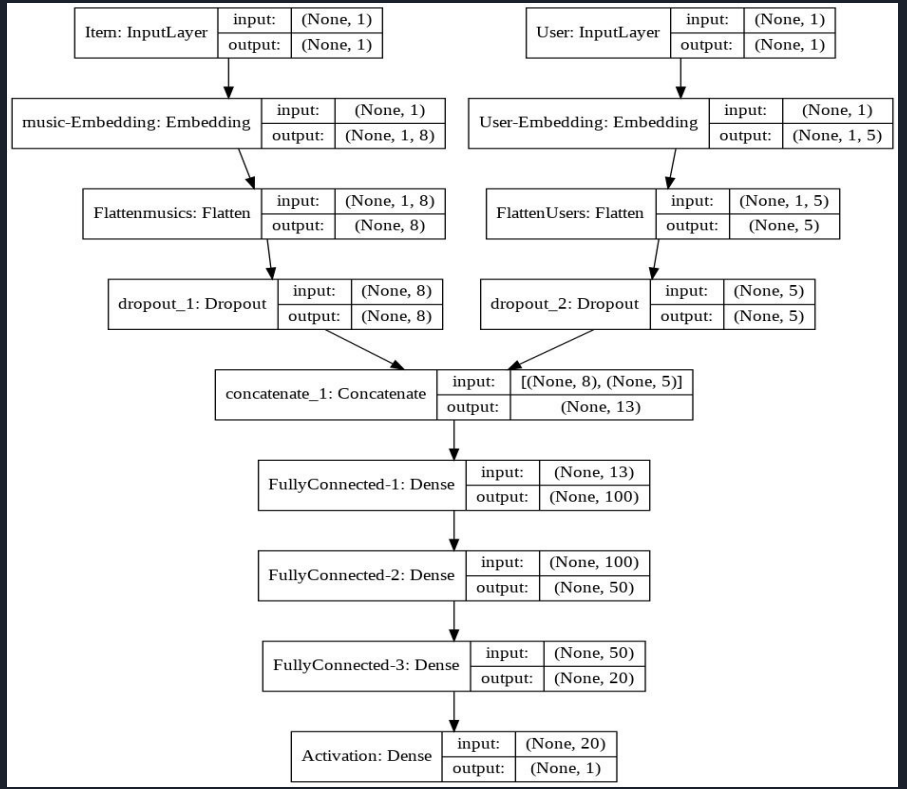
Multilayer Perceptron Model

Key Parameters

Number of Latent Factors = 8 and 5
Loss Function = Mean Squared Error
Optimizer = Adam
Epochs = 10

Results

Train RMSE = 0.7743
Test RMSE = 0.8174



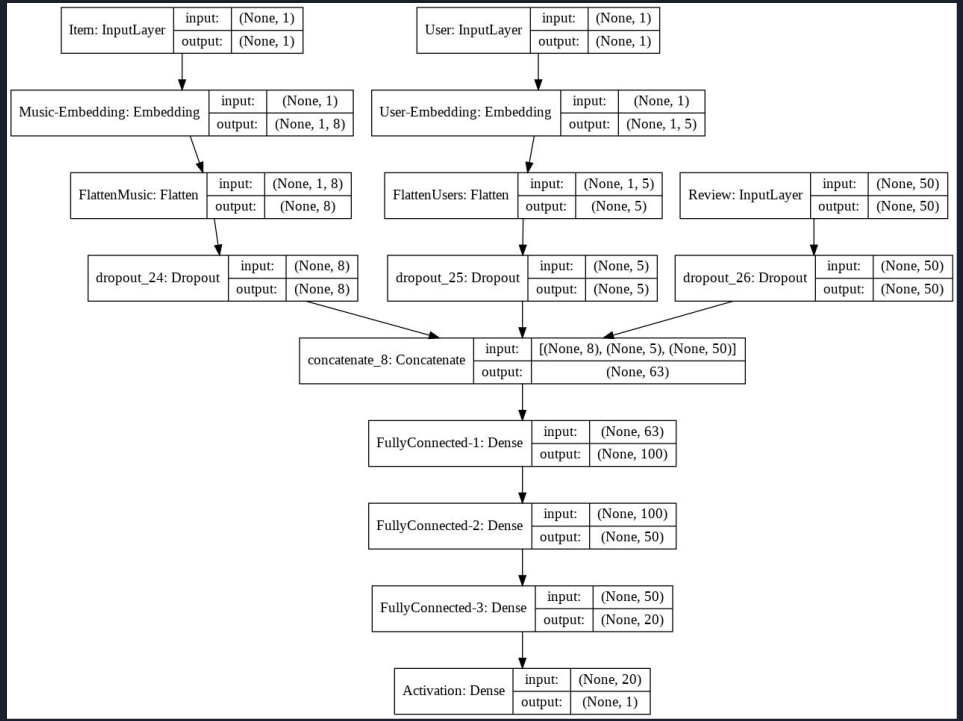
Word2Vec Multilayer Perceptron Model

Key Parameters:

Manual Word2Vec Embeddings
Number of Latent Factors = 8, 5 and 50
Loss Function = Mean Squared Error
Optimizer = Adam
Epochs = 10

Results

Train RMSE = 0.5269
Test RMSE = 0.7296



Conclusion

The best model to facilitate an Amazon's customers selection criterion for digital music is the Word2Vec Multilayer Perceptron model.

This model is able to predict what the user will rate the recommended item within +/- 0.7.

Model	Key Parameters	Test RMSE
GMF(1)	Dot Product(Keras)	4.643
GMF(2)	Regression Loss	2.490
GMF(3)	Logistic Loss	3.907
NNMF	Non-neg Embedding	4.951
MP	Concatentation	0.817
Word2Vec MP	Word2Vec	0.730

Thank you

Q & A Session