Machine Learning Engineer Nanodegree

Capstone Proposal

Ramya Sahithi Adari
December 9th, 2018

Title: Collaborative filtering Movie Recommendation system using embeddings and Neural Network.

Domain Background:

Movie recommendation engine filters the data using different algorithms and recommends the most relevant movies to users. It first captures the past behavior of a customer and based on that, recommends movies which the users might be likely to watch.

We use MovieLens 20M dataset which are widely used in education, research, and industry. They are downloaded hundreds of thousands of times each year, reflecting their use in popular press programming books, traditional and online courses, and software. These datasets are a product of member activity in the MovieLens movie recommendation system, an active research platform that has hosted many experiments since its launch in 1997.

A movie recommendation system, based on collaborative filtering approach makes use of the information provided by users, analyzes them and then recommends the movie that is best suited to the user at that time.

Collaborative Filtering Recommendation Systems are of Two Types:

1. User based Collaborative Filtering Recommendation Systems:
    - This algorithm first finds the similarity score between users based on the ratings they have previously given to different movies. Based on this similarity score, it then picks out the most similar users and recommends movies.
2. Item based Collaborative Filtering Recommendation Systems:
    - In this algorithm, we compute the similarity between each pair of items.
    - In our case we will find the similarity between each movie pair and based on that, we will recommend similar movies which are liked by the users in the past. This algorithm works like user-based collaborative filtering with just a little change – instead of taking the weighted sum of ratings of "user-neighbors", we take the weighted sum of ratings of "item-neighbors"

Related Work and research work:
https://www.researchgate.net/publication/288041090_The_MovieLens_Datasets

http://ijesc.org/upload/a23663ade860d69d5589b1a298301546.Movie%20Recommender%20System%20Movies4u.pdf

Problem Statement:

Building a **collaborative filtering movie recommendation system using Word embeddings, Matrix factorization methods by defining and implementing a Neural network with Keras**.

What Word embeddings?

> A word embedding is a class of approaches for representing words and documents using a dense vector representation.

Why word embeddings?

It is an improvement over more the traditional bag-of-word model encoding schemes where large sparse vectors were used to represent each word or to score each word within a vector to represent an entire vocabulary. These representations were sparse because the vocabularies were vast, and a given word or document would be represented by a large vector comprised mostly of zero values.

- Instead, in an embedding, words are represented by dense vectors where a vector represents the projection of the word into a continuous vector space.
- The position of a word within the vector space is learned from text and is based on the words that surround the word when it is used.
- The position of a word in the learned vector space is referred to as its embedding.

Datasets and Inputs:

The dataset is obtained from grouplens.org: **https://grouplens.org/datasets/movielens/20m/**
MovieLens itself is a research site run by GroupLens Research group at the University of Minnesota. The first automated recommender system was developed there in 1993.

The datasets describe ratings and free-text tagging activities from MovieLens, a movie recommendation service. It contains 20000263 ratings and 465564 tag applications across 27278 movies. These data were created by 138493 users between January 09, 1995 and March 31, 2015. This dataset was generated on October 17, 2016.

Users were selected at random for inclusion. All selected users had rated at least 20 movies.

Content:

No demographic information is included. Each user is represented by an id, and no other information is provided.

The data are contained in six files.

1. **tag.csv** that contains tags applied to movies by users:
   - userId
   - movieId
   - tag
   - timestamp
2. **rating.csv** that contains ratings of movies by users:
   - userId
   - movieId
   - rating
   - timestamp

3. **movie.csv** that contains movie information:
- movieId
- title
- genres

4. **link.csv** that contains identifiers that can be used to link to other sources:
- movieId
- imdbId
- tmbdId

5. **genome_scores.csv** that contains movie-tag relevance data:
- movieId
- tagId
- relevance

6. **genome_tags.csv** that contains tag descriptions:
- tagId
- tag

Acknowledgements: To acknowledge use of the dataset in publications, the following paper:
F. Maxwell Harper and Joseph A. Konstan. 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

We make use of Ratings.csv and movie.csv files to build a recommendation system. We want to build a model that takes a user, u_i and a movie, m_j, and outputs a number from 0.5-5, representing how many stars we think this user would give that movie.

Solution Statement:

The solution to the problem statement defined above is to build a Neural network model of recommendation system using Collaborative Filtering methods with word embeddings, process of finding solution as follows:

- User based collaborative filtering:

The prediction of an item for a user *u* is calculated by computing the weighted sum of the user ratings given by other users to an item ***'i'***.
The prediction *Pu,i* is given by:

$$P_{u,i} = \frac{\sum_v (r_{v,i} * s_{u,v})}{\sum_v s_{u,v}}$$

Here,
Pu,i is the prediction of an item
Rv,i is the rating given by a user *v* to a movie *i*
Su,v is the similarity between users
Now, we have the ratings for users in profile vector and based on that we have to predict the ratings for other users. Following steps are followed to do so:

1. For predictions we need the similarity between the user u and v. We can make use of Pearson correlation.
2. First, we find the items rated by both the users and based on the ratings, correlation between the users is calculated.
3. The predictions can be calculated using the similarity values. This algorithm, first calculates the similarity between each user and then based on each similarity calculates the predictions. **Users having higher correlation will tend to be similar.**
4. Based on these prediction values, recommendations are made.

- Movie based collaborative filtering:

  The prediction is given by:

$$P_{u,i} = \frac{\sum_N (s_{i,N} * R_{u,N})}{\sum_N (|s_{i,N}|)}$$

  Now we will find the similarity between items.

$$sim(i,j) = cos(\vec{i}, \vec{j}) = \frac{\vec{i} \cdot \vec{j}}{||\vec{i}||_2 * ||\vec{j}||_2}$$

  Now, as we have the similarity between each movie and the ratings, predictions are made and based on those predictions, similar movies are recommended.

- We make use of word embeddings dense vector representations instead of using traditional encodings for sparse categorical values in dataset

Benchmark Model:

During my research under this project and its dataset, I found the following two methods which could be considered as benchmark models for the problem statement:

1. **MovieLens 20M using Factorization Machine**.
   a. They got Mean Absolute Error: **0.60**, and
   b. Root Mean Squared Error: **0.80**.

   The research results can be found : https://support.treasuredata.com/hc/en-us/articles/360001260847-MovieLens-20m-Rating-Prediction-using-Factorization-Machine

2. **MovieLens 20M using Autoencoders.**
   a. They got Root Mean Squared Error(RMSE): **0.81**.

   The research results can be found : https://arxiv.org/pdf/1606.07659.pdf

Evaluation Metrics:

The Common evaluation metrics for Recommendation systems are:

a) **Mean Absolute Error(MAE):** Traditionally, mean absolute error (MAE) has been used to evaluate the performance of Collaborative Filtering algorithms. MAE works well for measuring how accurately the algorithm predicts the rating of a randomly selected item.
b) **Root Mean Squared Error(MSE):** Root Mean Squared Error (RMSE) is perhaps the most popular metric used in evaluating accuracy of predicted ratings. The system generates predicted ratings Rˆui for a test set T of user-item pairs (u,i) for which the actual ratings Rui are known. Typically, Rui are known because they are hidden in an offline experiment, or because they were obtained through a user study or online experiment. It varies from 0.0 to 1.0 with lower values being signaling less error (therefore "better"). So, here the evaluation of recommending a movie rating score is performed using RMSE. Root Mean Squared Error is not application-specific as such, and so tends to be included in most academic literature when evaluating the predictive accuracy of Recommender Systems.

Project Design:

- **Data Preprocessing:**
  - Extracting data, preprocess data by checking null values, outliers. Merging ratings and movies files to build a Collaborative recommendation system.
  - Discussing the problems occurs using old categorical data encoding methods.

- **Building a Neural Network Model :**
  - Building a Neural Network by adding **"Embeddings Layer"** to the architecture.
  - Making use of **"Matrix Factorization"** method to extract word embeddings.

- **Data Visualization:**
  - Performing exploratory data analysis to visualize important features of data to extract, and to find the degree of correlations and make use of them in building recommendation system using matplotlib.
  - Visualizing embeddings with t-SNE.

- **Evaluating model:**
  - Evaluating the performance of the model by comparing the evaluation metric scores of recommendations found with the benchmark model.
  - Fine tuning the parameters of the model to improve its efficiency.