Capstone Project Proposal

Movie Similarity

David Pinto

2020-10-29

Contents

I. Definition

Project Overview

The main goal of this project is to develop a **Movie Similarity Recommender** over the full version of the famous MovieLens dataset. The developed solution aims to recommend similar items given a particular movie, so that users that already watched the later are likely to enjoy the recommended movies. It involves **Recommender Systems**[1] theory and **Similarity Matching**[2] techniques.

To properly achieve this we combine two datasets, one that provides movie ratings to be used as interaction data, and another that provides movie metadata to be used as content features.

Movie Ratings

The full MovieLens dataset, which contains `27M ratings` given to `58K movies` by `280K users`.

Movie Metadata

Data retrieved from the TMDB Open API and available as a Kaggle Dataset. It contains information on 45k movies featured in the full MovieLens dataset. Features include title, description, genre and cast&crew information.

Problem Statement

Recommending similar items is a very common task in many online selling platforms[3]. It provides recommendations like: **related to items you've viewed**, **recommended for you based on your previous purchase**, **customers who bought this item already bought**, so on. The main challenges involved are:

(a) The cold start problem

How to recommend recent added products with few or no associated interaction data?

One way to cope with this consists in take advantage of product metadata (content features): `name`, `description`, `category`, `price level`, so on. Then we can link a product with others by content similarity. We call this as **Content Based** approach.

(b) The popularity bias problem

According to the **Long Tail phenomenon**[4], selling niche products has many advantages. Users are likely to pay more for niche products because they cannot find it everywhere. However, best seller products have more interaction data available to train a recommender system so that the fitted model may be subjected to popularity bias. In other words, the model will be more likely to recommend popular items.

Again, using content features to find similar items reduces popularity bias. But interaction data is very important to understand user behavior and cannot be discarded. So, in this project we apply a **Collaborative Filtering** factorization that takes into account item and user popularity.

Movie Embeddings and Similarity Matching

First of all, we use both interaction and content data to extract item embedding[5] features, which are low-dimensional dense vectors that captures some of the semantics of the item by placing semantically similar items close together in the embedding space. It allows efficient nearest neighbors search, so that we can find similar movies fast in application time.

- **Content Embedding:** embedding features extracted from text fields like title and description, and from categorical fields like genre and cast&crew.
- **Collaborative Filtering Embedding:** embedding features extracted from user-movie interactions given by the ratings.

All the details about the feature extraction process will be discussed later.

After computing the embedding features, similar movies will be recommended using **Approximate Nearest Neighbors** (ANN) search with cosine similarity. The cosine similarity is the most common affinity metric applied for similarity search in the embedding space.

Metrics

Users Intersection as a Similarity Measure

Let's consider every rating (from 1 to 5) above 3.5 as an explicit feedback that the user liked the movie. Then we can measure the similarity between a pair of movies (x, y) as:

$$S(x, y) = \frac{N_{xy}}{\sqrt{N_x N_y}}$$

where N_x is the number of users that liked movie x, N_y is the number of users that liked movie y, and N_{xy} is the number of users that liked both x and y.

So, $S(x, y)$ close to 1 is a strong evidence that users who liked x will like y too. We call this as **User Correlation**.

Despite providing good recommendations, this similarity is very sparse, resulting in a discrete recommendation. So, if we try to recommend the top 30 similar to each movie we will get less than 30 for many movies. In other words, we will have many movies with few or none similar items to be recommended.

The embedding features solve this problem. Since we are computing distances between pairs of movies in a dense and low-dimensional space, the embedding features provide a continuous recommendation. We are able to find similar items to any movie, with high evidence (low distance) or low evidence (high distance).

Evaluating the Embedding Features

How close the similarity match with embedding features can get to the similarity match given by the user's likes (user correlation)?

(a) Ranking correlation

In a attempt to answer this question, we can randomly select n test movies and get their top N similar using the $S(x, y)$ measure. Then, for each test movie, we get the distances $D(x, y)$ from the top N similar using the embedding features. Finally, for each test movie, we apply the Spearman's Rank-Order Correlation Coefficient (ρ) between S and $1/D$, which is equal to the Pearson's correlation between the rank values of them.

For all test movies, we get an average rank correlation:

$$\frac{1}{n} \sum_{i=1}^{n} \rho(S_i, 1/D_i)$$

(b) Precision and Recall at K

Given a threshold t_S for $S(x, y)$, we will assume that $S(x, y) \geq t_S$ indicates a strong/relevant similarity. So, given the K nearest neighbors from embedding features of a test movie:

$$\text{precision@k} = \frac{\#\text{relevants@k}}{K}$$

$$\text{recall@k} = \frac{\#\text{relevants@k}}{\#\text{relevants}}$$

For all test movies, we get an average Precision@K and an average Recall@K.

II. Analysis

Data Preparation

First of all, take a look at `data/raw` to get instructions on how to download the datasets.

Data Preparation was a quite hard extra work in this project. As a previous step, we had to apply some manipulations in order to prepare data for analysis:

1. We set MovieLens ID as the primary key in both datasets, MovieLens and TMDB.
2. We transformed Cast&Crew, movie Genre and movie Keywords data from a JSON format to a Tidy Data format.
3. We concatenated all movie Keywords in a single text field.
4. We concatenated all movie Tags assigned by the users in a single text field.

Then, we applied some filters in an attempt to discard weak relations in the data:

1. In the Cast data we selected actors that appeared in at least 10 movies.
2. In the Crew data we considered just `director`, `producer` and `writer`, and selected those that appeared in at least 10 movies.
3. From the movie metadata we selected just the text fields: `title`, `overview` and `tagline`.
4. From the ratings data (interaction data) we removed users which voted less than 3 tines and movies with less than 30 votes. Items and users with poor interaction information can induce Collaborative Filtering to learn wrong correlations in the data.

All code is well organized and documented in the `01-data-preparation.ipynb` notebook.

Data Exploration

From the Data Preparation step we finished up with 7 datasets in tidy format:

- `data/movie_info.csv`:

id	title	overview	tagline
18	Four Rooms	It's Ted the Bellhop's first n (. . .)	Twelve outrageous guests. Four (. . .)
479	Judgment Night	While racing to a boxing match (. . .)	Don't move. Don't whisper. Don (. . .)
260	Star Wars	Princess Leia is captured and (. . .)	A long time ago in a galaxy fa (. . .)

id	title	overview	tagline
6377	Finding Nemo	Nemo, an adventurous young clo (. . .)	There are 3.7 trillion fish in (. . .)
356	Forrest Gump	A man with a low IQ has accomp (. . .)	The world will never be the sa (. . .)

- `data/movie_keywords.csv`:
- `data/movie_tags.csv`:
- `data/movie_genre.csv`:
- `data/movie_actor.csv`:
- `data/movie_producer`:
- `data/movie_ratings`

In this section, you will be expected to analyze the data you are using for the problem. This data can either be in the form of a dataset (or datasets), input data (or input files), or even an environment. The type of data should be thoroughly described and, if possible, have basic statistics and information presented (such as discussion of input features or defining characteristics about the input or environment). Any abnormalities or interesting qualities about the data that may need to be addressed have been identified (such as features that need to be transformed or the possibility of outliers). Questions to ask yourself when writing this section: *- If a dataset is present for this problem, have you thoroughly discussed certain features about the dataset? Has a data sample been provided to the reader? - If a dataset is present for this problem, are statistics about the dataset calculated and reported? Have any relevant results from this calculation been discussed? - If a dataset is* ***not*** *present for this problem, has discussion been made about the input space or input data for your problem? - Are there any abnormalities or characteristics about the input space or dataset that need to be addressed? (categorical variables, missing values, outliers, etc.)*

Exploratory Visualization

In this section, you will need to provide some form of visualization that summarizes or extracts a relevant characteristic or feature about the data. The visualization should adequately support the data being used. Discuss why this visualization was chosen and how it is relevant. Questions to ask yourself when writing this section: *- Have you visualized a relevant characteristic or feature about the dataset or input data? - Is the visualization thoroughly analyzed and discussed? - If a plot is provided, are the axes, title, and datum clearly defined?*

Algorithms and Techniques

In this section, you will need to discuss the algorithms and techniques you intend to use for solving the problem. You should justify the use of each one based on the characteristics of the problem and the problem domain. Questions to ask yourself when writing this section: *- Are the algorithms you will use, including any default variables/parameters in the project clearly defined? - Are the techniques to be used thoroughly discussed and justified? - Is it made clear how the input data or datasets will be handled by the algorithms and techniques chosen?*

Benchmark

In this section, you will need to provide a clearly defined benchmark result or threshold for comparing across performances obtained by your solution. The reasoning behind the benchmark (in the case where it is not an established result) should be discussed. Questions to ask yourself when writing this section: *- Has some result or value been provided that acts as a benchmark for measuring performance? - Is it clear how this result or value was obtained (whether by data or by hypothesis)?*

III. Methodology

(approx. 3-5 pages)

Data Preprocessing

In this section, all of your preprocessing steps will need to be clearly documented, if any were necessary. From the previous section, any of the abnormalities or characteristics that you identified about the dataset will be addressed and corrected here. Questions to ask yourself when writing this section: - *If the algorithms chosen require preprocessing steps like feature selection or feature transformations, have they been properly documented? - Based on the* ***Data Exploration*** *section, if there were abnormalities or characteristics that needed to be addressed, have they been properly corrected? - If no preprocessing is needed, has it been made clear why?*

Implementation

In this section, the process for which metrics, algorithms, and techniques that you implemented for the given data will need to be clearly documented. It should be abundantly clear how the implementation was carried out, and discussion should be made regarding any complications that occurred during this process. Questions to ask yourself when writing this section: - *Is it made clear how the algorithms and techniques were implemented with the given datasets or input data? - Were there any complications with the original metrics or techniques that required changing prior to acquiring a solution? - Was there any part of the coding process (e.g., writing complicated functions) that should be documented?*

Refinement

In this section, you will need to discuss the process of improvement you made upon the algorithms and techniques you used in your implementation. For example, adjusting parameters for certain models to acquire improved solutions would fall under the refinement category. Your initial and final solutions should be reported, as well as any significant intermediate results as necessary. Questions to ask yourself when writing this section: - *Has an initial solution been found and clearly reported? - Is the process of improvement clearly documented, such as what techniques were used? - Are intermediate and final solutions clearly reported as the process is improved?*

IV. Results

(approx. 2-3 pages)

Model Evaluation and Validation

In this section, the final model and any supporting qualities should be evaluated in detail. It should be clear how the final model was derived and why this model was chosen. In addition, some type of analysis should be used to validate the robustness of this model and its solution, such as manipulating the input data or environment to see how the model's solution is affected (this is called sensitivity analysis). Questions to ask yourself when writing this section: - *Is the final model reasonable and aligning with solution expectations? Are the final parameters of the model appropriate? - Has the final model been tested with various inputs to evaluate whether the model generalizes well to unseen data? - Is the model robust enough for the problem? Do small perturbations (changes) in training data or the input space greatly affect the results? - Can results found from the model be trusted?*

Justification

In this section, your model's final solution and its results should be compared to the benchmark you established earlier in the project using some type of statistical analysis. You should also justify whether these results and the solution are significant enough to have solved the problem posed in the project. Questions to ask yourself when writing this section: - *Are the final results found stronger than the benchmark result reported earlier? - Have you thoroughly analyzed and discussed the final solution? - Is the final solution significant enough to have solved the problem?*

V. Conclusion

(approx. 1-2 pages)

Free-Form Visualization

In this section, you will need to provide some form of visualization that emphasizes an important quality about the project. It is much more free-form, but should reasonably support a significant result or characteristic about the problem that you want to discuss. Questions to ask yourself when writing this section: - *Have you visualized a relevant or important quality about the problem, dataset, input data, or results? - Is the visualization thoroughly analyzed and discussed? - If a plot is provided, are the axes, title, and datum clearly defined?*

Reflection

In this section, you will summarize the entire end-to-end problem solution and discuss one or two particular aspects of the project you found interesting or difficult. You are expected to reflect on the project as a whole to show that you have a firm understanding of the entire process employed in your work. Questions to ask yourself when writing this section: - *Have you thoroughly summarized the entire process you used for this project? - Were there any interesting aspects of the project? - Were there any difficult aspects of the project? - Does the final model and solution fit your expectations for the problem, and should it be used in a general setting to solve these types of problems?*

Improvement

In this section, you will need to provide discussion as to how one aspect of the implementation you designed could be improved. As an example, consider ways your implementation can be made more general, and what would need to be modified. You do not need to make this improvement, but the potential solutions resulting from these changes are considered and compared/contrasted to your current solution. Questions to ask yourself when writing this section: - *Are there further improvements that could be made on the algorithms or techniques you used in this project? - Were there algorithms or techniques you researched that you did not know how to implement, but would consider using if you knew how? - If you used your final solution as the new benchmark, do you think an even better solution exists?*

References

[1] Google's Introduction to Recommendation Systems.

[2] Building a real-time embeddings similarity matching system.

[3] The Amazon Recommendations Secret to Selling More Online.

[4] Anderson, Chris. The long tail: Why the future of business is selling more for less. Hyperion, 2006.

[5] Overview: Extracting and serving feature embeddings for machine learning.