Project F-17 Phase 1

Group Members:

- Pungampalayam Shanmugasundaram, Sachin Sundar
- Gondane Shubham
- Thorat Abhishek
- Rampally Shiva Tejaswi
- Upmaka Raviteja
- Bejjipuram Krishna Chaitanya

Abstract

The project proposes the use of datasets MovieLens and IMDB consisting of movies, actors, tags, genres and ratings. This project focuses on using vector space models to use in information retrieval based on how these datasets relate to each other. All the user ratings and tags are timestamped and use this recent information to better map the objects to the vector tag space. The phase 1 of the project is divided into four tasks, first three of which uses the Term frequency - Inverse document frequency(Tf-idf) weights to evaluate how important a term or word is to a document in a corpus depending on the given inputs. Hence models are time weighted Tf and Tf-idf. The 4^{th} task uses the concept of probabilistic relevance feedback to differentiate between two vectors generated from the given input. We used movieid as the middle layer to map between given inputs of each task and the tags. The end result of each task is a vector consisting of tags and weights in decreasing order of weights.

Keywords: TF, TF-IDF, probabilistic relevance feedback, rank-reciprocal

1. Introduction

1.1. Terminology

- Corpus: Collection of all documents.

- Document: Collection of terms.

- Term Frequency(TF): Term frequency measures how a frequently a term occurs in a document.
  TF(t) = (Number of times term t appears in a document) / (Total number of terms in the document).

- Inverse Document Frequency(IDF): The inverse document frequency is a measure of how much information a term provides.
  IDF(t) = log (Total number of documents / Number of documents with term t in it).

- Term frequency–Inverse document frequency (TF-IDF): This weight is a statistical measure used to evaluate how important a word is to a document in a collection or corpus. The importance increases proportionally to the number of times a word appears in the document but is offset by the frequency of the word in the corpus.

- Min-Max normalization: It is a linear transformation that transforms real data values to a range between minimum and maximum values.

1.2. Goal description:

The project phase is divided into four tasks as follows

Task 1:
We need to consider for a given actor all the movies played by him and to store a weighted tag vector. The models used here are time-weighted TF and TF-IDF so that newer tags created should get more weight compared to older tags. Another factor to consider is that we need to consider the rank of an actor in a movie. The output of the program must be sorted in descending order of weights.

Task 2:
This task focuses on mapping the genre to tag vector space. The models used are time-weighted TF and TF-IDF.

Task 3:
This task focuses on mapping the users to tag vector space. The tags need will consist of all movies watched by the user. The models used are time-weighted TF and TF-IDF.

Task 4:
This task focuses on differentiating two genres from each other.
The task itself is divided into three subtasks

1. Given two genres g1 and g2 we need to find TF using set of movies for given genre and for calculating IDF we need to take union of all movies in g1 and g2.
2. Instead of using TF-IDF models we find weights for tags of genre1, here using the probabilistic relevance feedback model P-DIFF1. The mechanism used to calculate weights is given in the following formula.
3. This subtask focuses on using probabilistic relevance feedback model P-DIFF2. The inputs to the formula change here.

1.3. Assumptions

We make assumption that a user has watched a movie if he has rated it or tagged it.

2. Proposed solution/Implementation:

For each task, we need a time-weighted TF and TF-IDF so we need a way to calculate the weights to be used such that most recent timestamp will receive higher weight.
Method to calculate weights for timestamps:

1. Sort the timestamps in order of newest first to oldest last.
2. Give them a rank from 1 to n
3. Apply the following rank reciprocal **(Stillwell et al. 1981)** weighting formula for each timestamp rank

$$weight(r_i) = (1/r_i) / \sum_{i}^{n} 1/r_i$$

r_i is rank for tag t_i.
The weight is calculated by using the rank reciprocal which then normalized by dividing the rank reciprocal with the sum of reciprocals.

Task1:
We need to map the actorid to the tags considering the timestamps and actor rank in the movies. We need to calculate weights for actor ranks as well.

Method to calculate weights for actor ranks:
For each tag t, map the actor rank for the movie which has that tag t.
Again, applying the same weighting formula to calculate weights from ranks **(Stillwell et al. 1981)**.

$$weight(r_i) = (1/r_i) / \sum_{i}^{n} 1/r_i$$

r_i is rank for tag t_i.
The weight is calculated by using the rank reciprocal which then normalized by dividing the rank reciprocal with the sum of reciprocals.

So, each tag will have a corresponding actor rank weight. So, if same tag repeats in many movies it will have multiple weights then maximum of that weights will be taken into consideration while pairing with the tag.

To calculate TF, we need to find the tags related to the movies played by the actor. So, for each movie-tagid pair we calculate the TF for each tagid. The timestamp weights are then added to the calculated TF. If the tagid repeats in many movies the calculated (TF + timestamp) weights are summed up and only one value of (TF + timestamp) weights are maintained for each tagid. For model TF, we need to add actor rank weights to (TF + timestamp) weights. This will give the final weights for each tagid. These tagid-weight pairs are sorted according to the final weights in descending order. The tagid are mapped to the actual tags.

Model: TF Final weights = TF + timestamp + actor rank weights

To calculate TF-IDF we are calculating IDF based on the entire movies dataset of the actor. This will give IDF weights to each tagid. The IDF weights are multiplied to the (TF + timestamp) weights for each tagid. The resultant weights are then added with the actor rank weights. This tagid-weight pairs are sorted according to the final weights in descending order. The tagid are mapped to the actual tags.

Model: TF-IDF Final weights = ((TF + timestamp) * IDF) + actor rank weights

Task 2:
For the given genre as input we map the genres to all movies of that genre. The movies are mapped to the tags then, so we have a movie-tagid pair. The time stamps are weighted as specified in the beginning of this section.

For each movie-tagid pair we calculate the TF corresponding to a particular tagid. The timestamp weights are then added to the calculated TF. If the tagid repeats in many movies the calculated (TF + timestamp) weights are summed up and only one value of (TF + timestamp) weights are maintained for each tag. These final weights are sorted in decreasing order and then mapped to tags corresponding to each tagid.

Model: TF Final weights = TF + timestamp

To calculate TF-IDF we need to calculate IDF for each tagid. To calculate TF-IDF we are calculating IDF based on the entire movies dataset of the genre. This will give IDF weights to each tagid. The IDF weights are multiplied to the (TF + timestamp) weights for each tagid. This tagid- weight pairs are sorted according to the final weights in descending order. The tagid are mapped to the actual tags.

Model: TF-IDF Final weights = ((TF + timestamp) * IDF

Task 3:
The procedure remains same as in task 2. The only difference is that the dataset of movies is generated based on userid. To find the dataset we need to consider the fact that a user must have written a tag for a watched movie and didn't give a rating to the movie, or maybe the user may have watched the movie and gave rating but didn't write a tag.

Task 4:
The task is divided into three subtasks which will be handled as follows.

Task 4.1:
Here we are taking inputs genre1 and genre2, the aim is to show how genre1 differs from genre2. To calculate time-weighted TF we are using the dataset mapped from genre1 to the tagids. After calculating the time weights and TF we are adding them together for each tagid. To calculate the

IDF the dataset considers the union of the movies in genre1 and genre2. The resultant IDF is then multiplied with the TF calculated for the tagids of genre1 movies. The resultant TF-IDF weights are mapped to tags using tagid, which are then sorted according to the weights in decreasing order.
We are using Manhattan distance to compare between the (tag, weight) vector of the two given genres and printing the value of manhattan distance.

Task 4.2:
In this subtask, instead of using TF-IDF-DIFF to find how genre1 differs from genre2, we are using probabilistic relevance feedback model. The model used here is P-DIFF1 which calculates the weights for each tag for genre1 as follows:

$$w_{1,j} = \log \frac{r_{1,j}/(R-r_{1,j})}{(m_{1,j}-r_{1,j})/(M-m_{1,j}-R+r_{1,j})} \times \left| \frac{r_{1,j}}{R} - \frac{m_{1,j}-r_{1,j}}{M-R} \right|$$

where:

$r_{1,j}$ is the number of movies in genre, g1, containing the tag t_j

$m_{1,j}$ is the number of movies in genre, g1 or g2, containing the tag t_j

R = ∥ movies(g_1) ∥ , and

M = ∥ movies(g_1) ∪ movies(g_2) ∥ .

The above-mentioned formula poses problems for few cases when

$r_{1,j}$ *is equal to* $m_{1,j}$

R *is equal to* $r_{1,j}$

We need to add an approximation of 0.5 to the above formula (**Salton and Buckley [1990])** which then transforms to the following.

$$w_{1,j} = \log \frac{r_{1,j}+0.5/(R-r_{1,j}+0.5)}{(m_{1,j}-r_{1,j}+0.5)/(M-m_{1,j}-R+r_{1,j}+0.5)} \times \left| \frac{r_{1,j}}{R} - \frac{m_{1,j}-r_{1,j}}{M-R} \right|$$

Task 4.2:
Similar to task 4.1 with change in the parameters of the formula used to calculate the weights for tags in genre1

$$w_{1,j} = \log \frac{r_{1,j}/(R-r_{1,j})}{(m_{1,j}-r_{1,j})/(M-m_{1,j}-R+r_{1,j})} \times \left| \frac{r_{1,j}}{R} - \frac{m_{1,j}-r_{1,j}}{M-R} \right|$$

Where:

$r_{1,j}$ is the number of movies in genre, g1, not containing the tag t_j

$m_{1,j}$ is the number of movies in genre, g1 or g2, not containing the tag t_j

$R = \| \text{movies}(g_1) \|$, and

$M = \| \text{movies}(g_1) \cup \text{movies}(g_2) \|$.

4. System requirements/installation and execution instructions

4.1. System requirements/installation

1. Operating System: MacOS Sierra x86 architecture
2. Database: Using in-memory data structure. The python package pandas has data structure- dataframe to store the dataset.
3. Python version – 3.6
4. Packages – Pandas, Numpy

Installation steps:
Download Anaconda which is an open source distribution of the Python and R programming languages for large-scale data processing, predictive analytics, and scientific computing, that aims to simplify package management and deployment.
Download link: https://www.anaconda.com/download/

4.2. Execution instructions

To execute the programs, go to the terminal and navigate to the project folder and type the following.
Run task1:
./print_actor_vector.py actorid model
Example - ./print_actor_vector.py 1698048 TF

Run task2:
./print_genre_vector.py genreid model
Example – ./print_genre_vector.py Comedy TF

Run task3:
./print_ user_vector.py userid model
Example – ./print_ user_vector.py 146 TF

Run task4:
./differentiating_genre.py genre1 genre2 model
Example – ./differentiating_genre.py Comedy Drama P-DIFF1

5. Conclusion

Based on the TF-IDF weight calculation, we have determined the relevancy of objects in the feature space. Traditional TF-IDF considers the number of times the term appeared in the document and the its importance relative to others in the dataset. TF-IDF in this project has been modified to not only consider the occurrence of the terms but also other related weight which makes more relevance in calculating the final weight. To find the difference between object, distance or similarity measure is used. Manhattan distance is used to find the distance between the objects which will give how differentiating the objects are in the tag vector space. Lesser the distance, the more similar are the genres. To further differentiate the genres in better way, we are using probabilistic relevance feedback to calculate the weights for a genre

6. References

[1] SALTON G, BUCKLEY C. Term-weighting approaches in automatic text retrieval [J]. Information Processing and Management, 1988, PP513 - 523.

[2] Gerard Salton and Chris Buckley -Improving Retrieval Performance by Relevance Feedback. JOURNAL OF THE AMERICAN SOCIETY FOR INFORMATION SCIENCE- June 1990

[3] Stillwell, W.G., Seaver, D.A., Edwards, W., A comparison of weight approximation techniques in multiattribute utility decision making. Organizational Behavior and Human Performance, 28 (1981) 62-77.