PHASE 1 F17

Group Members:

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

Abstract:

The project includes database of movies, actors, tags-movies, users and ratings. Based on that we need to propose a solution to relate components of dataset to each other based on various inputs like actorid, genreid, userid. This project introduces a TF-IDF computations on movie lens dataset based on various input criteria. It also proposes how computations for TF and TF-IDF differs in range of documents. All the user ratings and tags are timestamped and that helped us to distinguish between same tags. We also have experimented and analyzed the differentiating factor between range of documents based on given **TF-IDF-DIFF, P-DIFF1, P-DIFF2**. We also analyzed that combining weights with TF-IDF model based on some comparative conditions makes a subtle difference in output like in Task 4 section b, c we used probabilistic feedback mechanism instead of TF-IDF model and came to know that how documents weighing criteria makes difference in output for same range of input. We used **movieid** as a middleware between two entities to form corpus (like in Task I – **movieid** used as middleware between **actorid** and **tags**).

Keywords: TF, TF-IDF, probabilistic relevance feedback, Time-weighted TF

Introduction:

- **Terminology:**

Term Frequency (TF): TF is basically number of times a term occurs in a document. To normalize it we usually define TF as number of times a term occurs/ total number of documents.

Document Frequency(DF): DF is defined as number of documents in which term has occurred

Inverse Document Frequency (IDF): IDF is defined as log of reciprocal of Document Frequency

TF-IDF: TF-IDF is defined as product of term frequency and IDF. It is a measure of how important the term is in set of documents (Corpus)

Min-Max normalization - It is used to normalize set of datasets in 0 and 1 to give more perspective on how dataset has spanned.

- **Goal Description:**

Task1 –

Calculate TF and TF-IDF for a tag appeared in movies played by given actorid on weighing criteria such as higher timestamp should get more weight than lower timestamp and same with movie-actor rank

Task2 –

calculate TF and TF-IDF for a tag appeared in movies under given genre on weighing timestamp weighing criteria

Task3-

Calculate TF and TF-IDF for a tag appeared in movies that are watched by given userid. The weighing criteria is higher weight should give to tags with higher timestamp

Task4-

In this task, we basically need to compare how given genres are different from each other. For this reference will be tags under given genres

In subtask 1, we will compute TF-IDF for two given genres and will display how they are different under this model

In Subtask 2 and 3 we will compute weight of each tag in genre1 based on probabilistic feedback mechanism.

Proposed Solution and Implementation:

Task 1:

First, we need corpus of documents to compute TF-IDF for given tags. After getting input as actorid, we will get all the movies played by actor. For each movie, we will get all the tags related to it. So, **movies->tag** map will be the corpus. **Each movie-tag map** will act as a document in a corpus.

To compute TF, we need to consider every single tag in movie documents. TF will be per document for a given tag. To find out time-weighted TF, we first need to normalize the timestamp data in 0 and 1. For that, we have used min-max normalization. Normalization will contain all the timestamps in a corpus for tags. After calculating TF for each tag per document basis we will add timestamp to it based on current tag and its value.

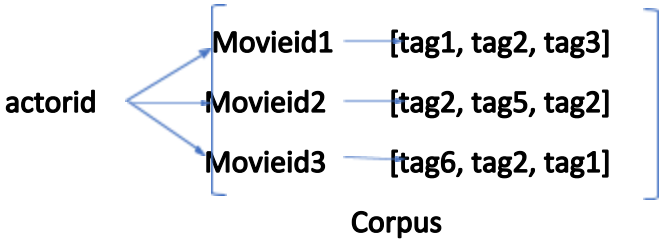
To compute IDF, we need to consider all the documents in corpus for a given tag. We will compute IDF as log (total number of tags/number of documents in which tag has appeared).

To compute TF-IDF, we will multiply TF(tag) with IDF(tag). As we computed TF on per document basis and IDF on corpus basis, it will retain the property of TF-IDF.

After computing TF-IDF we need to merge all tags based on **movie_actor_rank.** We also need to normalize movie_actor_rank set using min-max normalization.

Based on what model has passed as command line argument we will compute TF or TF-IDF

Structure of corpus –



Formula –

TF = (TF + time-weight based on timestamp)

TF-IDF weight = (TF + time-weight based on timestamp) * IDF + movie_actor_rank

Min-Max Normalization = x – min(set)/(max(set)-min(set))

Task 2:

Task 2 is very like Task1. In Task 2 we are getting input as genre. So, we will get all the movies under given genre. For each movie, we will get related tags to it. Every step is same as Task 1 except while combining timeweighted tag vector we will simply add them.

Formula –

TF-IDF weight = (TF + time-weight based on timestamp) * IDF

Task 3:

For task 3 we will get input as userid. Based on that we will find out movies that user has watched and associative tags to it. Again, we will follow same process to compute TF-IDF as followed in Task1 and 2.

Formula –

TF-IDF weight = (TF + time-weight based on timestamp) * IDF

Task 4:

For task 4 we have 3 subtasks. We will approach them differently. Basically, in task 4 we need to compare two genres based on tags they contain.

Subtask1 –

In sub task 1 there are two documents in a corpus. Document is **genre-tag map.** First, we will find out all the tags for genre1 by using movieid as a middleware. Similarly, we will find out all the tags for genre2. Now In our corpus we have two documents genre1-tags map and genre2-tags map. Now, we will calculate TF of each tag per document and IDF considering a complete corpus.

After TF and IDF calculations, compute TF-IDF for each tag. As a matter of fact, TF-IDF is used to differentiate genre1 and genre2 in terms on tags. But We need more suitable differentiating factor. We have used manhattan distance because this will remove redundancy by some margin if same tag is presents in both genre1 and genre2.

Subtask2 and Subtask3-

In subtask 2 we are using probabilistic feedback mechanism to assign weights to tags in same corpus as defined in subtask1. In subtask 1 we used TF-IDF, instead of that we will use probabilistic feedback formula. Now there are many edge conditions to it. Now after calculating this we need some measure to find out actual diff between two vectors.

For all divide by zero conditions, we will use approximation factor of 0.5 referred from research paper referenced in References.

System Requirements and Installation:

1. Operating System: Linux x86 architecture
2. Database – KDB/Q in memory database
3. Python version – 2.7
4. Packages – Pandas, Numpy, Logger

Installation steps:

I have created single bash executable that will automatically install all the required packages and database. Please run requirements.sh executable in sudo mode.

Command – sudo bash requirements.sh

This will download KDB/Q community version and install it. Also, it will install all packages of python with pip.

Execution Steps:

1. As you run requirements.sh it will load data into database
2. Running each Task
    a. Task 1
    Command **- ./print_actor_vector actorid model**
    Example - **./print_actor_vector 1234 TF**
    Model – TF or TF-IDF
    b. Task 2
    Command - **./ print_genre_vector genre model**
    Example - **./ print_genre_vector "Action" TF-IDF**
    Model - TF or TF-IDF
    c. Task 3
    Command - **./print_user_vector userid model**
    Example - **./print_user_vector 3 TF-IDF**
    Model – TF or TF-IDF
    d. Task 4
    Command- **./differentiate_genre genre1 genre2 model**
    Example- **./differentiate_genre "Action" "Action|Comedy" P-DIFF1**
    Model – TF-IDF-DIFF or P-DIFF1 or P-DIFF2

Conclusion:

TF-IDF is a technique used to categorize documents. TF-IDF basically classifies corpus of documents into sub categories inside the document itself. Also, TF-IDF can be used as differentiating factor for comparing number of documents. TF is just normal measure of how frequent term occurs in document. But to compare set of documents we need more exclusive and better method. TF-IDF exactly calculates terms importance in set of documents. Time weighted TF is a good way to modify original TF concept to give more detailed and summarized results. As, Time weighted consider input preferences based on some comparative parameters. Document relevance factor is one interesting area of exploration. Approximation factor in probabilistic relevance feedback can overcome edge conditions. we have determined the relevancy of objects in the feature space. 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.

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