MWD PROJECT

Phase 1 – Group 2

Abstract

Tags have been associated with content for quite some time. The world wide web made a use of meta tags to identify the content of the page resulting in fast search algorithms for search engines like Google. When tags are not present, a technique called Term Frequency – Inverse Document Frequency (TF-IDF) has been used associate tags to assist in document querying and text retrieval. It is also used in the case when document matching is to be done. In the project phase, we have used the concept of TF-IDF weights to map actors, users and genres on a tag vector space. It helps in identifying which tags are most representative of actors, users and genres of movies in the IMDB dataset. Once we have vectors that have tags as their dimensions and tag TF-IDF weights as their values, we can use all kinds of vector space distances to understand how users, actors and genres are related, whether they are related or if they are completely unrelated. Queries like what two actors have done similar kind of movies, what two genres are similar, or what two users have watched similar kinds of movies can be answered in an efficient manner.

Keywords: Text, document, IMDB data, movies, retrieval, TF-IDF, probabilistic, feedback

Members:

Syed Hashmi
Mihika Shah
Vaishnavi Raj
Ishan Dikshit
Satheesh Kurunthiah
Raj Sadaye

Table of Contents

Introduction 2
Terminology 2
TF-IDF 2
Vector Space 2
Probabilistic Feedback Mechanism 2
Goal description 3
Assumptions 3
Task 1 3
Task 2 3
Task 3 3
Task 4 3
Description 4
Task1 4
Task 2 4
Task 3 4
Task 4 4
Interface Specification and Execution Instructions 5
Task 1 5
Task 2 5
Task 3 5
Task 4 6
Task X 6
Conclusion 6
Bibliography 7

Introduction

This phase deals with converting actors, users and genres into n-dimensional tag space. A task of differentiating genre using TF-IDF weights associated with their tags, and two probabilistic feedback mechanisms. These mechanisms are used to generate tag weights to represent the importance of tags instead of TF-IDF weights.

Terminology

This section will deal with description of technical terminologies used throughout the report.

TF-IDF

Term frequency – Inverse Document Frequency is a combination of two terms. Term Frequency and Inverse Document Frequency. Term Frequency is a measure of how many times a term has appeared in a document with respect to all the terms that have appeared. Term frequency signifies the importance of a tag in this document. (SALTON G, 1988)

$$tf = \frac{n}{T}$$

where n = number of times a term has appeared in the document,
and T = total number of terms in the document

Inverse document frequency is used for getting the inverse of importance of the tag in the corpus. The more number of documents the tag would have appeared in, the less its IDF will be.

$$idf = \log\left(\frac{N}{m}\right)$$

where N = number of total documents in the corpus
and m = number of documents where this term has appeared

As we can see, IDF does not signify anything of importance itself so it is never used alone.

$$tfidf = tf \times i\text{df}$$

It is used with TF to reduce the weight associated to a document if it is used in many other documents too. It creates a more realistic weight when using term frequency for measuring document importance.

Vector Space

A space consisting of vectors, together with the associative and commutative operation of addition of vectors, and the associative and distributive operation of multiplication of vectors by scalars. (Google, n.d.)

Probabilistic Feedback Mechanism

The formula for PFM is taken from the assignment problem description document.

The model below is used for task 4 part 2

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

where:

* $r_{1,j}$ is the number of movies in genre, g_1, containing the tag t_j
* $m_{1,j}$ is the number of movies in genre, g_1 or g_2, containing the tag t_j
* $R = \|movies(g_1)\|$, and
* $M = \|movies(g_1) \cup movies(g_2)\|$.

The model below is used for task 4 part 3

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

where:

* $r_{1,j}$ is the number of movies in genre, g_2, not containing the tag t_j
* $m_{1,j}$ is the number of movies in genres, g_1 or g_2, not containing the tag t_j
* $R = \|movies(g_2)\|$, and
* $M = \|movies(g_1) \cup movies(g_2)\|$.

Goal description

The tasks performed in this phase will help in representing genres, actors and users in an n-dimensional vector space where tags are dimensions and the values given to each tag in this space for any actor, genre and user is defined using TF-IDF. TF-IDF weights and probabilistic feedback mechanism weights are also used to differentiate between genres.

Assumptions

Task 1

Actor is a document, and tag is a term. All the actors will make the corpus.

Task 2

Genre is a document, and tag is a term. All the genres will make the corpus.

Task 3

User is a document, and tag is a term. All the users who have rated a movie, or tagged it are assumed to have watch the movie hence are all part of the corpus.

Task 4

Genre is a document, and tag is a term. The corpus is only the total number of genres associated with the movies that has the given two genres.

Description

Task1

When calculating TFIDF for tags for actors, n is the number of times tag has appeared for all the movies done by the actor and T is the number of all the tags that have appeared for all the movies done by the actor. N is the total number of actors in the corpus and m is the number of actors in the movies that have the tag. (Refer to the formula of TFIDF in terminology)

Task 2

When calculating TFIDF for tags for genres, n is the number of times tag has appeared for all the movies that have the given genre and T is the number of all the tags that have appeared for all the movies done by the actor. N is the total number of genres in the corpus and m is the number of genres for the movies that have the tag. (Refer to the formula of TFIDF in terminology)

Task 3

When calculating TFIDF for tags for users, n is the number of times tag has appeared for all the movies that have the given user's watched movies and T is the number of all the tags that have appeared for all the movies watched by the user. N is the total number of users in the corpus and m is the number of users who have watched the movies that have the tag. (Refer to the formula of TFIDF in terminology)

Task 4

This task has three parts.
First part uses the difference in TF-IDF to calculate weights for the difference vector of the two genres. First both the genres are mapped onto the tag space using TF-IDF. Once two vectors are available, Euclidean distance is used to signify the distance between the two vectors.

On its own, one distance does not tell much about the differentiation between two vectors but if a genre is compared to two different genres, getting two distances, they can help understand which genre is closer to the given genre with respect to the other genre.

Second part and third part uses probabilistic feedback mechanism. Whenever a divide by zero appears, the four parts of the left side of the formula are all added with 0.5. This will ensure that the divide by zero will never occur.

I have calculated the final difference vector as well as the distance of the final vector as well. I have done it for all possible genre1, genre2 combinations where genre1 and genre2 are all the genres in the dataset and stored in output/pdiff1.txt, output/pdiff2.txt and output/tfidfdiff.txt files. It can be observed that the diff value for genres Action and Adventure is much smaller (less different from each other) than the diff value for genres Action and Children. It means that

on the mapping provided by the tag space, Action and Adventure genres are very less different than Action and Children genres are different.

Interface Specification and Execution Instructions

The software is developed in python 2.7.xx(any distribution of 2.7 can be used to execute the program). External python libraries are required for the software to run. A readme file is provided that explains exactly how to install them. Once installed, below is a description of how to run each task.

Sanity check before running tasks:
Open command line interface, Powershell or Command Prompt, in Windows, Terminal in Linux and run the command:

```
python --version
```

It should output “Python 2.7.xx+” where xxx is a number
If it does not return Python 2.7.xx+ then please install python 2.7 and add it to system variables, restart terminal/cmd and try again.

Task 1

To run task 1, go to the codes directory, open a command line interface and enter command

```
python print_actor_vector.py actorid model
```

where actorid can be replaced by any integer from the list of actorids in the dataset
and model can only take two values “tf” and “tfidf”

Task 2

To run task 2, go to the codes directory, open a command line interface and enter command

```
python print_genre_vector.py genre model
```

where genre can be replaced by any of the 19 genres from the list of genres in the dataset
and model can only take two values “tf” and “tfidf”

Task 3

To run task 3, go to the codes directory, open a command line interface and enter command

```
python print_user_vector.py userid model
```

where userid can be replaced by any of the userids given the tags and ratings data in the dataset and model can only take two values “tf” and “tfidf”

Task 4

To run task 4, go to the codes directory, open a command line interface and enter command

python differentiate_genre.py genre1 genre2 model

where genre1 and genre2 can be replaced by any of the genres given the movies data in the dataset and model can only take values “tfidf”, “pdiff1” and “pdiff2”

Task X

To run all the tasks for all the possible options, I have created an option. This option will allow you to generate data for all the tasks and all possible input options and then store them in the output file. You can change the dataset and generate the output for all possible combinations and analyze them if you wish to. The command to enter on command line is:

python generate_all_output.py

Conclusion

This phase was a difficult phase to implement not knowing in what direction the project will head in the future. We have done it to the best of our capabilities and to the best of our understanding of vector spaces and document representation in Hilbert spaces. Part 4 was particularly confusing where we had to choose our own methods of differentiation between vectors coming out from the two genres given. I am confident that with the tasks accomplished in this phase of the project can help answer difficult queries like which two genres are similar and which two genres are not. The output directory contains a rather interesting set of Euclidean distances between genres mapped onto tag space showing how Crime and Thriller have a much smaller distance when compared to the distance between Crime and Children genres.

Bibliography

Google. (n.d.). *https://www.google.com/search?q=define+vector+space&oq=define+vector+space&aqs=chrome.0.0l6.3168j0j7&sourceid=chrome&ie=UTF-8*. Retrieved from Google.

SALTON G, B. C. (1988). Term-weighting approaches in automatic text retrieval . *Information Processing and Management*, 513 - 523.