Data Matching: Global Film Titles

Solving mismatched titles

Summary

1. Introduction
2. Unique Identifiers
3. Possible Solutions
4. Using IMDb
5. Using Google
6. Using Other APIs
7. Improvements
8. Conclusions

OpusData comscore usheru INTERNATIONAL SHOWTIMES API AccuWeather ni gruvi

Introduction

When retrieving data from several sources, it is important to guarantee its relevance and consistency.

In the case of movies retrieved from the webpages of cinemas, the title is often the only identifier of the movie. However, the same film can be given multiple versions of its title, as a result of translations, formatting issues, naming conventions, etc.

The challenge is to match movie titles, in order to understand when the same film is being referenced, so it can be tracked correctly.

Unique Identifiers

In order to match different titles to the same movie, it is useful to use unique identifiers.

For books, there is the International Standard Book Number (ISBN).

For movies, there is the International Standard Audiovisual Number (ISAN). However, this number system is not widely used. As such, we initially decided to use the IMDb number. The movie website IMDb has cataloged virtually every movie ever made, and uses a numeric value to identify each one. There's no guarantee IMDb will keep things that way, but it's probably the best solution that currently exists.

OpusData comscore usheru INTERNATIONAL SHOWTIMES API AccuWeather ni gruvi

Possible Solutions

We figured there would be a need for a movie database with search functionality in order to match the film titles, as a methodical approach of trying to normalize the titles would never be enough to match them.

As such, once again we initially decided to use the IMDb searching functionality to find the mismatched titles' closest correspondence. Afterwords, we experimented using Google and other APIs.

Using IMDb

The Internet Movie Database is a publicly available online movie, television programs, video games and streaming content database. Originally a fan-operated website, the database is now owned and operated by IMDb.com, Inc., a subsidiary of Amazon.

IMDb does have its own api, however it's paid and needs to be licensed. As such, we started by trying using web scraping (with the help of the requests and BeautifulSoup Python libraries) to get the desired search results and were able to get the IMDb number of a title's closest match.

The results were satisfactory, as we were able to successfully match about 86% of the movie titles in the sample set.

OpusData comscore usheru INTERNATIONAL SHOWTIMES API AccuWeather Ni gruvi

Using Google

When using Google for searching movie titles in imdb.com, we started getting better results, but were quickly met with a HTTP error 429 (Too Many Requests). After realizing we could get a free trial of its Custom Search API, we started using it to search the movies in IMDb. It worked great and we were able to match over 90% of the film titles.

Using Other APIs

We then experimented with other APIs from TMDB (The Movie Database) and OMDb (Open Movie Database). These gave us fewer matches, but were still useful for comparing results between APIs and also for getting more information on a movie.

Improvements

When not getting results, we started making some automatic changes to the titles, like removing some special characters, deleting some words, and overall normalizing the title, always keeping in mind not to turn our program biased to our dataset. We also allowed the possible disambiguation for movies with the same name released in different years.

This fallback normalization, combined with the use of the several matching sources with given priorities, gave us a matching result for 100% of the cases, although with a few differences between APIs, in which the result can be chosen either by the most popular or by following a certain criteria depending on the mismatched titles source (e.g. if we know the movies are recent or are currently in theatres).

All the requests to the 4 APIs were taking about 4 minutes for the 51 titles. However, that time improved by about 90% to about 25 seconds, once we implemented multithreading.

Conclusions

Although it is a tough job to match every possible title variant, we were able to get good matching results by using several APIs and normalization techniques. If the mismatched titles had any other information on the films, we believe the results could have been even more consistent.

We would also like to be able to try this technique in a much larger dataset to check the results.

Overall, it was a fun experience that allowed us to learn about web scraping and the use of APIs, as well as data manipulation, matching and consistency.

RealTeamComputing

- José Pedro Ferreira
- Lucas Calvet Santos
- Sérgio da Gama