**Identification of Representative Movie Actors: A Comparative Evaluation of Vector Space Model Capabilities**

Patrik Lytschman (50241262), Lukas Welker (50241179), Tony Meissner (50241167)

Chung Ang University

Natural Language Processing and Information Retrieval (53849)

Jason J. Jung

June 10, 2024

Contents

[1. Data Collection 1](#_Toc168998848)

[2. Classification model 1](#_Toc168998849)

[3. Implementation of IR system 1](#_Toc168998850)

[3.1 Mapping of actor the data 1](#_Toc168998851)

[3.2 Classification of scripts 1](#_Toc168998852)

[3.3 Vector space model 1](#_Toc168998853)

[4.1 FastAPI 2](#_Toc168998854)

[4.2 User Interface 3](#_Toc168998855)

[4.3 Deployment 3](#_Toc168998856)

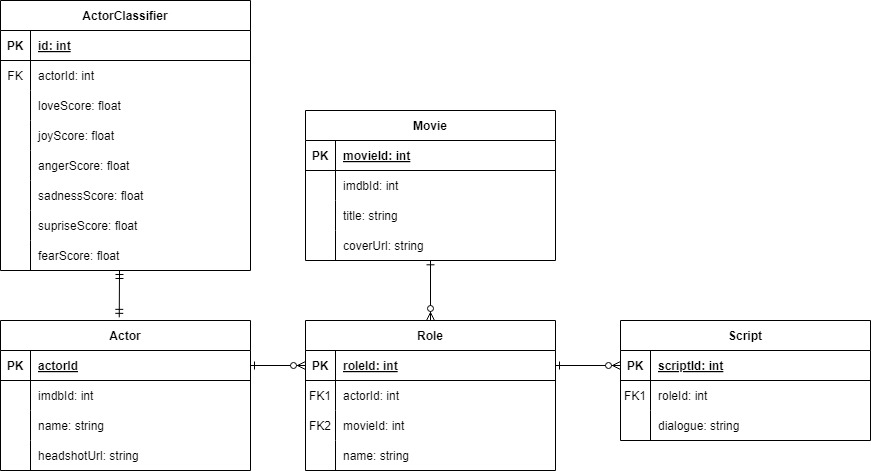
[5. Evaluation 3](#_Toc168998857)

# 1. Data Collection

# 2. Classification model

# 3. Implementation of IR system

## 3.1 Mapping of actor the data



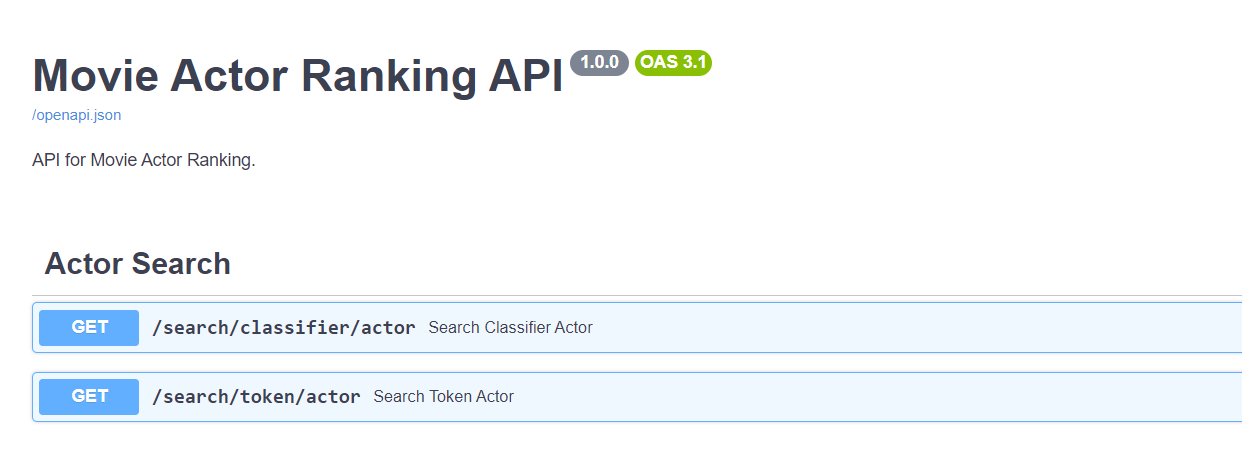
## 3.2 Classification of scripts

## 3.3 Vector space model

In the context of Information Retrieval (IR), the Vector Space Model (VSM) represents each actor or query as a vector in Euclidean space, where each dimension corresponds to a unique classification categories. For the implementation in this project, we used the script as the basis, thus representing actors and queries as vectors of classification values.  
In the context of actor or query vectors denoted as , where represents the real-valued classification value of the corresponding classification category, it's crucial to note that these values are constrained to fall within the interval I = [0,1]. This constraint is significant as it ensures that each classification category's value is normalized, meaning it ranges from complete absence (0) to absolute presence (1).

Now we have all prerequisites to carry out a ranking using the VSM and return the most representative movie actors based on a given query. Both Queries and actors are vectors in the |K|-dimensional Euclidean space, where the classification categories represent the axes of the space. We calculate the Cosine Similarity between each actor vector and query vector using the following formula: cosine\_similarity (q,d) = |​, where is the classification value of category in the query and di is the average classification value of category of the actor. And we rank the documents according to their Cosine similarity.  
For a random chosen query, we can observe cosine similarities between [0,99, 0,05].

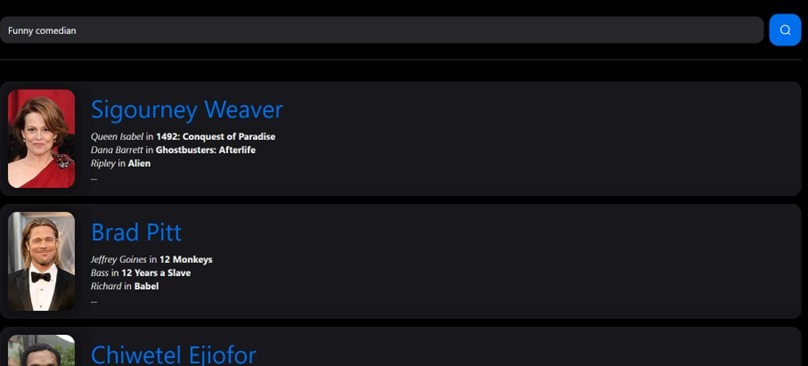
## 4.1 FastAPI

We implemented a FastAPI endpoint for both our *search\_vector\_space* function. This endpoint creates the opportunity for external applications or user interfaces to interact with our implemented IR system programmatically via HTTP requests. On the following picture you can see the implemented API endpoint. If a user is interacting with our IR system by entering an query, this specific query gets processed by our IR system. In the same manner it is possible to interact with our system by using an UI.  


Picture 1: API - Endpoint visualized using FastAPI

## 4.2 User Interface

Complementing the API access, we built a web interface with [Next.js](https://nextjs.org/) and [NextUI](https://nextui.org/) to offer a more interactive search experience. Users can directly interact with the IR system through their web browser.

On the following picture you can see the search process with the VSM, using an custom query.  


Picture 2: Implementation of the UI

## 4.3 Deployment

The entire application, including search functionalities and the web interface, is deployed on Microsoft Azure App Service to make it available to everyone.

Access the web interface here: https://app-movieactorranking-prod.azurewebsites.net/

Access the web API here: https://api-movieactorranking-prod.azurewebsites.net/

# 5. Evaluation