Final Project Report. Information Retrieval System (IRS)

Authors:
José Miguel Zayas Pérez C-312
Adrian Hernandez Santos C-311

Computer Science, MATCOM, UH.

Project Link: github

Table of Contents

1 Introduction

Information Retrieval Systems (IRS) propose a model and a similarity function between the query and the pre-processed data of the system. The system then returns the information that matches the query in an ordered manner.

This project aims to find a model and/or metrics that filter the data and only return information that genuinely responds to the query. To achieve this, it is necessary to implement a system that meets the requirements described above.

The preceding paragraph outlines the topic chosen for this project. To carry it out, a simple search engine was implemented to find video games within a dataset.

2 Data Acquisition

The dataset used in this project was obtained through web scraping. The Scrapy library in Python was used to extract the following data for each game on the gg.deals website:

- Title
- Genres
- Tags
- Description
- Image

Approximately 25,000 games were collected in total. The code used for scraping can be found in src/scraper.

3 Methodology

To construct the search engine, three similarity strategies were implemented to compare a query with a game. Each strategy utilizes different game features:

- Vectorial Model: Similar to document search, each game is treated as a document, and the game's description is the document's content. The TF-IDF technique is employed to convert each game's description into a feature vector. The user's query is also converted into a feature vector using the same technique. Cosine similarity is used to compare the query with each game.
- Tag-Based Similarity: This approach calculates similarity between the query and a game based on the tags associated with the game. It is measured as the number of query tags also present in the game divided by the total number of tags in the game.
- Genre-Based Similarity: Similar to the previous method, this approach calculates similarity using the genres associated with each game. It is measured as the number of query genres also present in the game divided by the total number of genres in the game.

To obtain the final similarity score between the query and a game, the similarity values from each of the aforementioned strategies are normalized and averaged. This results in a similarity score between 0 and 1.

4 Implementation

The Python programming language was used to implement the described system. Additionally, the following libraries were used:

- Scrapy: For web scraping.
- Gensim: For the vectorial model, cosine similarity, and TF-IDF.
- Spacy: For text pre-processing.
- SQLAlchemy: For database connection.
- FastAPI: For creating the API.

4.1 Tokenization

The previously mentioned Spacy library was used to tokenize the data for each game, removing stop words and lemmatizing the remaining words.

During the tokenization process for the user query, words were classified as tags or genres for subsequent similarity calculation.

5 Results and Transparency

During the similarity calculation between the query and games, the similarity values for each strategy were independently saved.

This allows users to see the similarity values for each strategy and understand why a particular game was recommended. Games are then presented to the user in descending order based on the total similarity score.

A simple web page was created using the FastAPI library to display these results.

6 Potential Improvements

- The system currently lacks semantic checking for user queries, which can lead to unwanted results. Furthermore, the results could be improved by expanding the query, for example, by using synonyms.
- A Boolean model based on tags and genres could be implemented to reduce the search space. While an initial attempt was made to implement this model, processing the user query and converting it into a Boolean expression proved challenging. This could potentially be achieved only through an API call to a large language model (LLM), leading to the decision to omit this implementation.
- A recommendation model based on the user's search history could be implemented.