Movie Recommendations based on Embeddings

Promothesh Chatterjee*

Creating a Movie Recommendation System Using Genre Embeddings: Business Context and Rationale

In the modern entertainment industry, streaming platforms like Netflix, Hulu, and Amazon Prime rely heavily on personalized recommendations to enhance user experience, drive engagement, and retain subscribers. Understanding user preferences and providing tailored content is crucial for staying competitive in this market. This is where sophisticated recommendation systems come into play.

Business Rationale of Movie Recommendation Systems

1. Enhancing User Experience: Providing personalized recommendations improves user satisfaction and engagement. Users are more likely to find content that aligns with their tastes, leading to longer watch times and increased loyalty.

2. Increasing Retention Rates: A well-functioning recommendation system can reduce churn rates by keeping users engaged with relevant content. Happy users are less likely to cancel their subscriptions.

3. Driving Content Discovery: Recommendation systems help users discover new content they might not have found on their own. This is particularly important for keeping the platform's content fresh and exciting.

4. Competitive Advantage: In the crowded streaming market, offering superior recommendations can differentiate a platform from its competitors. It becomes a key selling point in attracting and retaining users.

The Idea Behind Genre Embeddings As we have seen, word embeddings are a type of word representation that allows words with similar meaning to have a similar representation. They are used in various NLP applications to capture semantic relationships between words. Common techniques to generate embeddings include Word2Vec, GloVe, and fastText.In this context, genre embeddings represent movie genres in a high-dimensional space. By converting genres into embeddings, we can quantify and compare the semantic similarity between different genres. This method leverages the underlying concept of word embeddings to understand the relationships and similarities between different movie genres.

Technical Implementation: Detailed Explanation

Step 1: Data Preparation We start with a dataset of movies, each associated with a list of genres. This data is essential as it forms the basis for our embeddings and similarity calculations.

Step 2: Genre Embeddings We use pre-trained word vectors to convert genres into embeddings. The embedding process involves: - Splitting the genre string into individual genres. - Converting these genres into lowercase to ensure consistency. - Filtering out genres that don't have pre-trained embeddings. - Calculating the mean vector for each movie's genres to get a single embedding per movie.

Step 3: Cosine Similarity Cosine similarity is used to measure the similarity between two embeddings. It quantifies how similar two vectors (in this case, movie embeddings) are, based on the angle between them.

Step 4: Recommendation Function The core function of the recommendation system: - Takes a movie title as input. - Finds the corresponding embedding. - Calculates cosine similarities with all other movie embeddings. - Excludes the input movie from the results. - Returns the top N most similar movies along with their genres.

Visualization and Analysis

Though visualization isn't the primary focus, it can be useful: - **PCA (Principal Component Analysis)**: Reduces the dimensionality of genre embeddings for visualization. - **Plotting**: Helps in understanding the distribution and clusters of genres, providing insights into genre similarities.

Implementing Movie Recommendation System Using Word Embeddings in R

1. Setting Up the Environment Libraries and Data: To begin, we need to load the necessary libraries and prepare the data. We'll use `wordVectors` for pre-trained word embeddings, `text2vec` for cosine similarity, and `dplyr` and `ggplot2` for data manipulation and visualization.

```
# Load necessary libraries
library(wordVectors)
library(dplyr)
library(text2vec)
library(ggplot2)

# Load your pre-trained word vectors
model = read.vectors("Data/IMDB_vectors.bin")
```

Movie Data: Next, we'll create a dataset of real-life movies and their genres. Of course, here we are using a very limited list of movies with genres, the list can be expanded and the same code will work.

```
# Example movie data with real-life movie names and genres
movie_data <- data.frame(
  title = c("Inception", "The Dark Knight", "Forrest Gump",
            "Pulp Fiction", "The Matrix", "Titanic", "The Avengers",
            "Get Out", "La La Land", "Interstellar", "Gladiator",
            "Schindler's List", "The Godfather", "The Shawshank Redemption",
            "The Lion King", "Jurassic Park", "The Silence of the Lambs",
            "Star Wars", "Back to the Future", "The Terminator", "Goodfellas",
            "Braveheart", "The Sixth Sense", "Avatar", "Fight Club", "The Departed",
            "The Prestige", "The Social Network", "Mad Max: Fury Road",
            "Guardians of the Galaxy"),
```

```
  genres = c("sci-fi, thriller",
             "action, thriller",
             "drama, romance",
             "crime, drama",
             "sci-fi, action",
             "drama, romance",
             "action, sci-fi",
             "horror, thriller",
             "romance, musical",
             "sci-fi, drama",
             "action, drama",
             "drama, history",
             "crime, drama",
             "drama, crime",
             "animation, adventure",
             "sci-fi, adventure",
             "crime, thriller",
             "sci-fi, adventure",
             "sci-fi, comedy",
             "sci-fi, action",
             "crime, drama",
             "action, drama",
             "horror, mystery",
             "sci-fi, adventure",
             "drama, thriller",
             "crime, thriller",
             "drama, mystery",
             "drama, biography",
             "action, adventure",
             "action, sci-fi")
)
```

2. Extracting Genre Embeddings To utilize word embeddings effectively, we need to convert the genres of each movie into embeddings.

Function to Get Genre Embeddings: This function splits the genre string, converts it to lowercase, and calculates the mean embedding vector for the genres that exist in our pre-trained model.

```
# Function to get embeddings for a list of genres
get_genre_embedding <- function(genres, model) {
  genre_list <- unlist(strsplit(genres, ",\\s*"))
  genre_list <- tolower(genre_list)
  genre_list <- genre_list[genre_list %in% rownames(model)]
```

```
  if (length(genre_list) == 0) return(NULL)
  colMeans(model[[genre_list, average = FALSE]])
}

# Calculate embeddings for each movie based on genres
movie_data$embedding <- lapply(movie_data$genres, get_genre_embedding, model)

# Filter out movies with no valid embeddings
movie_data <- movie_data[!sapply(movie_data$embedding, is.null), ]
embeddings <- do.call(rbind, movie_data$embedding)
```

3. Creating the Recommendation Function Cosine Similarity Function: We'll use the `text2vec` package to calculate cosine similarity between movie embeddings.

```
# Function to calculate cosine similarity using text2vec
calculate_cosine_similarity <- function(a, b) {
  return(text2vec::sim2(a, b, method = "cosine", norm = "l2"))
}
```

Recommendation Function: This function takes a movie title, finds its embedding, calculates cosine similarities with other movies, and returns the top N most similar movies along with their genres.

```
# Recommendation function
recommend_movies <- function(movie_title, movie_data, top_n = 3) {
  # Find the embedding of the input movie
  movie_index <- which(movie_data$title == movie_title)
  if (length(movie_index) == 0) {
    stop("Movie not found in the dataset.")
  }
  movie_embedding <- matrix(movie_data$embedding[[movie_index]], nrow = 1)

  # Calculate cosine similarities
  similarities <- calculate_cosine_similarity(embeddings, movie_embedding)[, 1]

  # Exclude the input movie from recommendations
  similarities[movie_index] <- -Inf

  # Get the indices of the top_n most similar movies
  top_indices <- order(similarities, decreasing = TRUE)[1:top_n]

  # Return the titles and genres of the recommended movies
  recommendations <- movie_data[top_indices, c("title", "genres")]
```

```
  return(recommendations)
}
```

Example Usage: Here we demonstrate how to use the recommendation function to find movies similar to "Inception".

```
# Example: Recommend movies similar to "Inception"
recommendations <- recommend_movies("Inception", movie_data)
print(recommendations)
```

```
##              title           genres
## 25      Fight Club  drama, thriller
## 8          Get Out horror, thriller
## 2  The Dark Knight action, thriller
```

Understanding the Code

1. Data Preparation: - We define a dataset of movies with their titles and genres. - `get_genre_embedding` converts genre lists into embeddings by averaging the embeddings of individual genres.

2. Cosine Similarity: - Cosine similarity measures the cosine of the angle between two vectors, providing a similarity score between -1 and 1. - `calculate_cosine_similarity` uses `text2vec` to compute these scores, which is essential for comparing movie embeddings.

3. Recommendation Logic: - The `recommend_movies` function finds similar movies based on their genre embeddings. - The function: - Finds the embedding for the input movie. - Computes cosine similarities with all other movies. - Excludes the input movie from the similarity comparison. - Returns the top N most similar movies along with their genres.

Visualization and Analysis

Although the primary goal is to recommend movies, visualizing genre embeddings and their clusters can offer additional insights. One can perform PCA and plot the embeddings to understand how genres are grouped in the embedding space.

Applications in Business

1. Personalized Recommendations: Use genre embeddings to tailor recommendations to individual users based on their viewing history. For example, if a user frequently watches sci-fi and thriller movies, recommend movies with similar genre embeddings.

2. Marketing and Targeting: Segment users based on their preferred genres and create targeted marketing campaigns. For example, users who watch a lot of romance and drama could be targeted with promotions for new romantic dramas.

3. Content Acquisition and Production: Analyze genre trends and user preferences to inform content acquisition and production decisions. If embeddings reveal a high demand for a specific genre blend (e.g., sci-fi and action), the platform can prioritize acquiring or producing more content in that space.

4. Improving User Retention: By consistently providing relevant and engaging content, platforms can improve user retention. An effective recommendation system reduces the likelihood of users leaving due to content fatigue or dissatisfaction.

Conclusion

Genre embeddings and cosine similarity offer a sophisticated approach to building a recommendation system that significantly enhances user experience, drives engagement, and provides a competitive edge in the streaming industry. Understanding and implementing these techniques is essential for business analytics professionals aiming to leverage data-driven insights to optimize content delivery and user satisfaction. This project demonstrates the practical application of advanced NLP techniques in a business context, showcasing the value of personalized recommendations in the digital entertainment landscape.