Content-Based Movie Recommendation System Using Enhanced TF-IDF and Word Embeddings

Fanuel Dana and Wilbert Fundira

May 7, 2025

1 Introduction

Streaming platforms like Netflix and Amazon Prime have transformed the entertainment industry, with recommendation systems playing a massive role in their success. These platforms employ complex algorithms to analyze viewing histories and content metadata, suggesting titles that align with users' preferences and tastes. In this project, we were inspired by the platforms to develop our own content-based movie recommendation system that focuses on the semantic analysis of movie plot descriptions. Our approach, which we discuss in this write-up, involved processing the natural language descriptions of movie plots to identify thematic and conceptual similarities between films using TF-IDF and cosine similarity. We then enhanced the standard TF-IDF approach by incorporating semantic relationships between words using pre-trained GloVe embeddings, allowing the system to identify movies with thematically similar content even when they use different terminology. The system also employs edit distance inspired by Norvig's spell corrector, for robust movie title search functionality and provides recommendations by highlighting common important terms between movies using ANSI escape color-codes in the terminal.

2 Data Preprocessing

We obtained our dataset of movie plot descriptions from Kaggle [1]. The dataset is a subset of The Movie Database (TMDB). The raw data included 10 856 movies, each with fields such as `id`, `original_title`, `overview`, `release_year`, and metadata like `genres`, `cast`, and `director`. After loading the CSV file into a pandas DataFrame, we filtered out any entries missing an `overview`.

We then performed the following cleaning steps (all applied in `setup.py`):

1. Converted all text to lowercase.
2. Removed special characters and numeric digits using regular expressions.
3. Tokenized the text with NLTK's `word_tokenize` function.

4. Removed English stopwords (using NLTK's built-in list) and discarded tokens shorter than 3 characters.

We built a sparse count matrix of shape $10,856 \times 52,317$ in COO format. Rows represent individual movies, columns represent unique tokens, and values are computed using TF-IDF weighting. We then converted this matrix to CSR format for efficient arithmetic operations. Using the Python `pickle` library, we created serialized files of the various dictionaries and data structures to facilitate efficient loading during system runtime. These pickled objects include:

- `movie_dict`: Maps movie IDs to matrix indices
- `words_dict`: Maps unique tokens to their column indices in the TF-IDF matrix
- `id_to_title_dict`: Maps movie IDs to their titles for easy lookup
- `tfidf_matrix`: The sparse matrix containing TF-IDF values
- `df`: The preprocessed pandas DataFrame containing movie information
- `sim_csr`: A word similarity matrix based on GloVe embeddings
- `enhanced_tfidf`: The TF-IDF matrix enhanced with semantic relationships

3 Methodology

Our movie recommendation system employs a content-based filtering approach that uses the semantic richness of movie plot descriptions. This section details the technical implementation of our system, focusing on the search functionality, recommendation algorithm, and semantic enhancement techniques.

3.1 User Interface and Search

We implemented a command-line interface (CLI) that prompts users to enter a movie title query. Given the challenge of exact title matching, we incorporated a robust search mechanism using the Levenshtein distance algorithm to handle spelling variations and partial matches:

1. When a user enters a query, we compute the edit distance between the query and all movie titles in our dataset.
2. Titles with an edit distance below a threshold (set to 6) are considered as potential matches.
3. We return up to 10 closest matches, ranked by their edit distance from the query.

4. If no matches are found, we inform the user and prompt for a new query.

5. If multiple matches are found, we present the options and let the user select the intended movie.

6. If only one match is found, we automatically proceed with that selection.

This approach ensures that users can easily find their desired movie even with imperfect recall of the exact title or typing errors.

3.2 Recommendation Algorithm

To recommend movies similar to a selected movie m, we:

1. Retrieved the enhanced TF-IDF vector for m.

2. Computed cosine similarities between m and all other movies.

3. Excluded m itself and sorted the remaining movies by similarity.

4. Returned the top 5 most similar films along with their common high-scoring terms.

The system highlights the common "important" terms between the input movie and each recommendation using ANSI color codes via the colorama library [2]. These important terms are identified by comparing the highest TF-IDF scores across both movies and finding their intersection.

3.3 Word Embedding Enhancement

Initially, we computed standard TF-IDF weights on our cleaned corpus, but we observed that recommendations were limited to movies sharing the exact same vocabulary—semantic variants (e.g., *"automobile"* vs. *"car"*) were not captured. For example, a search for "Shrek" would only retrieve movies that explicitly contained the word "Shrek" in their descriptions, missing thematically similar animated fantasy films that used different terminology.

To overcome this limitation, we enhanced our TF-IDF representation via propagation over a word-similarity graph built from pre-trained GloVe embeddings [3]. We:

1. Loaded pre-trained GloVe word vectors (50 dimensions) for all words in our vocabulary.

2. Constructed a word similarity matrix by computing cosine similarity between all word vectors.

3. Applied a threshold of 0.8 to the similarity matrix to retain only meaningful semantic relationships.

4. Propagated TF-IDF values across this similarity graph with a weight parameter $\alpha = 0.1$:

$$\text{TF-IDF}_{\text{enhanced}} = (1-\alpha) \cdot \text{TF-IDF}_{\text{original}} + \alpha \cdot (\text{TF-IDF}_{\text{original}} \cdot \text{SimMatrix}) \tag{1}$$

[4]

This approach allowed the system to identify movies with thematically similar content even when they used different terminology, significantly improving the quality and diversity of recommendations. The enhanced TF-IDF matrix was stored in CSR format for efficient arithmetic operations and memory usage.

4 Results

We assessed the program in interactive terminal sessions and focused on three questions:

1. Can a user locate the right film even when the title is misspelled or only partially remembered?
2. Do the recommended movies feel narratively related to the query?
3. Does the interface give a clear textual rationale for each recommendation?

4.1 Robustness of Title Search

Recall that every user query is first passed through a **Levenshtein distance** function between the query string and every title in the dataset. A match is kept whenever its edit distance is ≤ 6, and the candidates are ranked by ascending distance.

- **Misspelled input handled gracefully.** Entering the misspelled string `shtek` demonstrated the effectiveness and efficiency of the algorithm. Although no exact title matched the input, the algorithm found *Shrek* within one edit operation and returned 381 candidate titles whose edit distance was ≤ 6. In our testing, any input within the six-character threshold produced at least one sensible hit, confirming that the edit-distance filter achieves the desired balance between fault tolerance and precision.
- **High-recall short queries.** The two-token query `iron man` produced 42 candidates. Titles such as *Birdman* or *Blankman* appear because their edit distance to *iron man* is small once tokenization and lower-casing are applied, giving users several plausible alternatives while still ranking the exact Marvel titles at the top.

4.2 Similarity Scores and Ranking Behaviour

Tables 1 and 2 list the five highest-scoring recommendations for the two test queries. Several patterns emerge.

Table 1: Top-5 recommendations for *Shrek Forever After*.

Rank	Recommended title	Cosine sim.
1	*Shrek 2*	0.362
2	*Shrek the Third*	0.316
3	*Transamerica*	0.250
4	*Elsa & Fred*	0.245
5	*The Disappearance of Eleanor Rigby: Her*	0.242

Table 2: Top-5 recommendations for *Iron Man*.

Rank	Recommended title	Cosine sim.
1	*The Clearing*	0.191
2	*Iron Man 2*	0.175
3	*Clown*	0.166
4	*The Disappearance of Alice Creed*	0.146
5	*3 Nights in the Desert*	0.128

Franchise awareness. For *Shrek Forever After* the two direct sequels head the list, each with a similarity score above 0.31. Their overviews share the character names (*Shrek*, *Fiona*) and setting (*Far Far Away*), all colour-highlighted in the CLI. This shows that the model captures word overlap and ranks near-duplicates appropriately.

Semantic recall beyond surface words. More interesting are the non-franchise hits:

- *Transamerica* ranks third despite not containing the token "shrek". Both plots revolve around a protagonist journeying *far* from home to repair family ties—the shared term *far*, strengthened via embedding propagation, appears in red.
- *Elsa & Fred* involves late-life romance; its presence is owed to themes of *marriage* and *reclaiming* one's life, surfaced through semantically related vocabulary (*reclaim*).

Lower absolute similarities for heterogeneous plots. Cosine scores for *Iron Man* are roughly half those of the *Shrek* case (0.19 vs. 0.36 at rank 1). We expected this: some of the tech jargon like ("arc•reactor", "Iron•Man") is rare across other franchises. Even so:

- *The Clearing* tops the list because the kidnapping story (*captive*, *held*) mirrors Iron Man's cave-captivity origin story.
- *Clown* appears solely through the concept pair (*suit*, *evil*), illustrating the embedding-Propagation signal: "suit" receives weight from semantically related garments (*armour*, *costume*).

5 Conclusion and Future Work

In summary, the combination of Levenshtein-based search with enhanced TF-IDF + GloVe propagation yields recommendations that are both fault-tolerant and semantically meaningful, though still susceptible to common keyword-frequency biases. Below are some limitations and possible future works for us.

5.1 Limitations

Current limitations:

- Some plot overviews in our data set may be poorly or inadequately written. Theymay alos be too short to really extract any meaningful semantics from them to aid in the recommendation.
- No modeling of multi-word expressions or context-specific meanings.
- A single global similarity threshold may not suit all word pairs.
- Lack of user-personalization or collaborative filtering components.

5.2 Future Work

Potential directions:

- Using more metadata from the dataset, like cast, genre etc.
- Incorporating user history for personalized weighting.

References

[1] Retrieved from https://www.kaggle.com/datasets/muhammetgamal5/tmdbmoviescsv/data, accessed 2020.

[2] Hartley, J. (2021). Colorama: Cross-platform colored terminal text. Retrieved from https://pypi.org/project/colorama/, accessed 2025.

[3] Pennington, J., Socher, R., Manning, C. D. (2014). GloVe: Global Vectors for Word Representation. Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), 1532-1543.

[4] Croft, W. B., Xu, J. (2000). Improving the effectiveness of information retrieval with local context analysis and other query expansion techniques. ACM SIGIR Forum, 34(2), 42–45.