SC4021 Information Retrieval

Course Assignment Report

Group 6

S/N	Team Member Name	Matriculation Number	Task
1	Sky Lim En Xing	U2223731A	Classification + Evaluation
2	Vijayanarayanan Sai Arunavan	U2120810F	Crawling + Indexing
3	Jaslyn Tan Hui Shan	U2121916G	Frontend + Backend
4	Xing Kun	U2023452E	Enhance Classification
5	Singhal Raghav	U2023945J	Dataset Labeling + Data Extraction

Table of Contents

1. Introduction

As part of the SC4021 Information Retrieval coursework, this project involves the development of an information retrieval system integrated with sentiment analysis capabilities. The objective is to first crawl a relevant text corpus based on a self-selected topic, construct a searchable index over the collected data, and subsequently analyze the sentiment embedded within the documents.

The topic and dataset domain are flexible, allowing exploration in areas such as social media marketing, political discourse, healthcare trends, financial forecasting, or personalized recommendation systems. This flexibility enables the system to be tailored according to the team's area of interest while fulfilling the core technical requirements of corpus processing, query handling, and sentiment evaluation.

1.1 Problem Statement

In the age of digital media, online platforms like IMDb have become central hubs for users to express their opinions about films. These movie reviews are rich in sentiment, ranging from overt praise to subtle critique. However, the sheer volume and subjectivity of user-generated content make it difficult for potential viewers, researchers, and content creators to systematically interpret public opinion.

The objective of this project is to perform sentiment analysis on IMDb movie reviews by classifying user opinions as positive or negative, while also capturing the emotional tone of each review (e.g., joy, anger, sadness). To further enhance interpretability and granularity, we also implemented:

- **Aspect-Based Sentiment Analysis (ABSA)** to identify sentiments tied to specific aspects of a movie (e.g., acting, plot, cinematography)
- **Sarcasm Detection** to mitigate misclassification caused by ironic or sarcastic reviews, which often reverse sentiment polarity

These enhancements enable a more precise and nuanced understanding of audience perception, improving applications such as targeted content recommendations and sentiment-driven marketing analytics.

Nonetheless, several challenges arise:

- **Subjectivity and Nuance:** Reviews often express mixed or layered sentiments and may include sarcasm that confuses traditional models.
- **Aspect-Level Complexity:** Unlike document-level sentiment, ABSA requires extracting relevant targets from within the review and associating them with sentiment phrases.
- **Language Variability:** Informal language, abbreviations, and inconsistent grammar reduce classification accuracy and complicate parsing.
- **Emotional Layering:** A single review may express multiple emotions simultaneously, requiring multi-class or multi-label handling.

By leveraging modern NLP techniques such as transformer-based models and task-specific fine-tuning, this project addresses these challenges through a robust pipeline that extracts sentiment polarity, emotional tone, specific aspect sentiments, and sarcastic cues from IMDb reviews.

2. Crawling of Text Data

Question 1.1: How you crawled the corpus (e.g., source, keywords, API, library) and stored it

2.1 Methodology Used for Data Crawling

The corpus was constructed by crawling IMDb to collect user reviews and structured movie metadata for **2024 science fiction films**. A hybrid crawling approach was employed combining both API calls and dynamic web scraping. We utilised the IMDbPY Python-wrapped API for IMDB, which was designed to aid in the retrieval and managing of the IMDb movie database about movies, people and companies.

As for our chosen library of scraping, we used Selenium, specifically the Chrome *WebDriver* extension in Selenium, for dynamic JavaScript-rendered pages. The process was divided into three main stages: **movie ID extraction**, **user review scraping**, and **metadata extraction**.

2.1.1 Movie Discovery and ID Extraction

The first phase involved identifying and collecting IMDb IDs corresponding to the relevant science fiction titles. Selenium was used to automate navigation to IMDb's Advanced Title Search page, where filters for genre and release year were applied to focus on the target dataset. To retrieve a comprehensive list of films, the crawler emulated repeated user interaction by automatically clicking the "Load More" button up to twenty-five times. This dynamic loading mechanism required the use of *WebDriverWait* to ensure that newly loaded content was fully rendered before subsequent interactions.

Once the full list of results was loaded, the page source was parsed using BeautifulSoup to locate all anchor (<*a*>) tags associated with movie titles. From these links, the IMDb IDs (in the form *ttXXXXXXX*) were extracted using regular expressions. Following extraction, each ID was passed into a custom *safe_get_movie_title()* function that leveraged the **IMDbPY** API to resolve each ID to its corresponding human-readable movie title. This function included a built-in retry mechanism to account for API timeouts or stochastic failures. Finally, all extracted IDs and resolved titles were stored persistently in a serialized dictionary using the Python pickle module, under the filename ***titles_ids.pkl***.

2.1.2 User Review Scraping

The second phase focused on retrieving user-submitted reviews for each film identified in the previous step. For each IMDb ID, the crawler navigated to the corresponding reviews page using Selenium. To ensure the inclusion of all available reviews, the script simulated user behavior by interacting with the “All” toggle and continuously clicking the “Load More” and “Show More” buttons until no further reviews were available. In addition, reviews marked as containing spoilers were programmatically expanded by executing JavaScript that triggered the click event on all spoiler-expansion buttons.

To maintain accuracy and completeness, the crawler used scroll emulation and *WebDriverWait* to ensure that the full review content was visible and the Document Object Model (DOM) had fully loaded before proceeding. The rendered HTML was then passed to BeautifulSoup for parsing. From each review, the script extracted the review title, body (with proper handling of <*br*> tags), the user’s star rating (if available), and the counts of helpful and unhelpful votes.

All reviews for a given movie were stored together in a .pkl file named using the pattern ***reviews/{imdb_id}_reviews.pkl***. This format ensured efficient and consistent storage, making the review data easily accessible for future indexing and analysis.

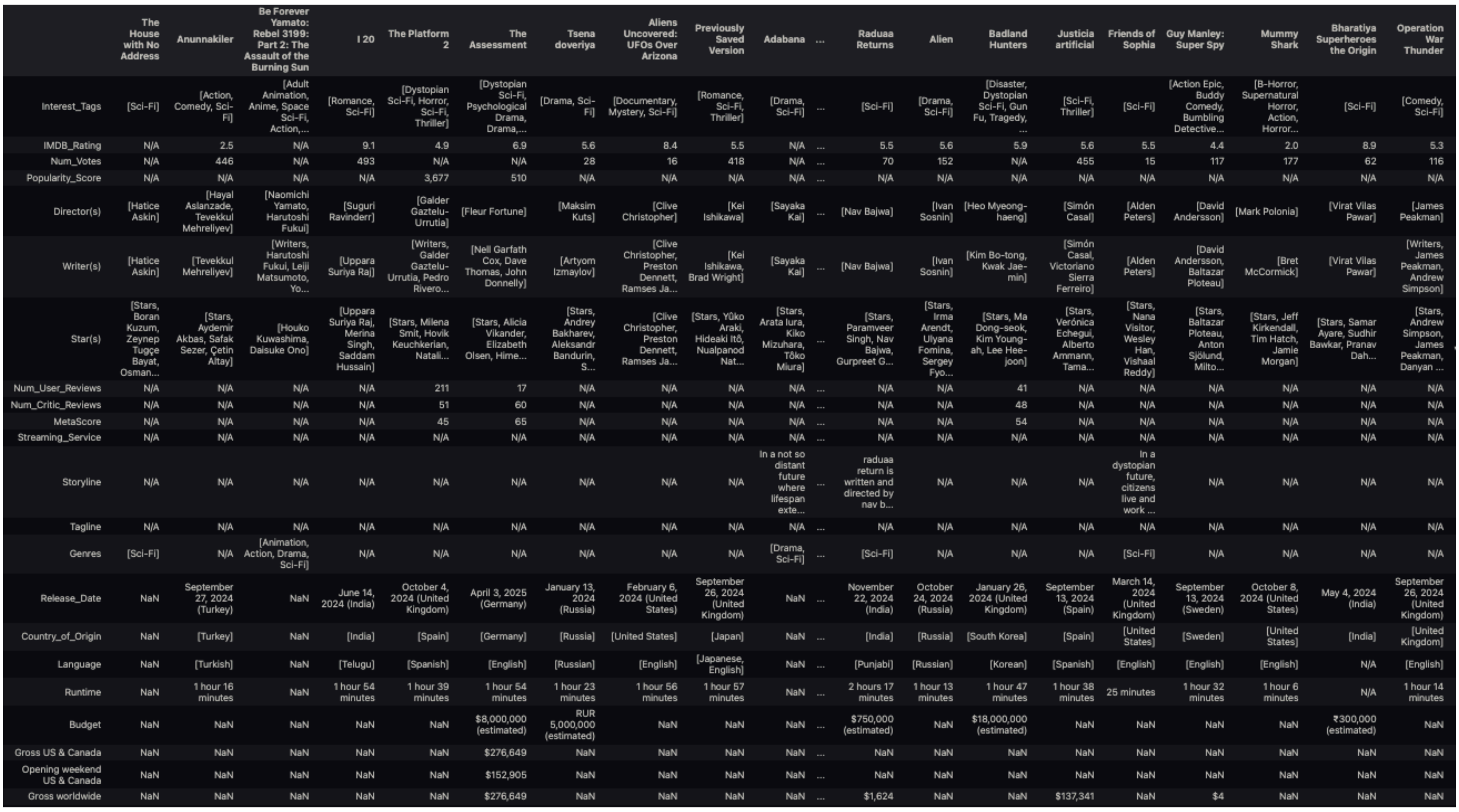
2.1.3 Structured Metadata Scraping

The final phase involved the extraction of structured metadata for each movie, designed to enrich the corpus with factual data and enable advanced filtering and faceted querying. For each IMDb ID, the crawler visited the main movie page and scraped a wide range of metadata fields, including the **IMDb rating** and **vote count**, **Metascore**, **director(s)**, **writer(s)**, **main cast**, and detailed box office information such as **budget**, **gross worldwide revenue**, **gross in the US and Canada**, and **opening weekend earnings**. Additional fields such as the **release date**, **runtime**, **primary language**, **country of origin**, **genre**, and **available streaming platforms** were also captured, along with descriptive elements like the movie’s **tagline** and **storyline summary**.

Many of these fields were hidden in collapsible or lazy-loaded sections of the IMDb interface. To address this, the crawler employed scroll emulation and DOM probing to detect and expand these elements dynamically. *WebDriverWait* was again used to ensure that all content was fully rendered before scraping began.

The extracted metadata was stored in a consistent and structured format using Python's pickle module. Each entry was saved in the directory Filtered_MetaData, with filenames in the form ***{imdb_id}_meta.pkl***, ensuring easy linkage with the corresponding review data.

An example of the crawled data, with both metadata and review data, is shown below in Figure 1.



	The House with No Address	Anunnakiler	Be Forever Yamato: Rebel 3199: Part 2: The Assault of the Burning Sun	I 20	The Platform 2	The Assessment	Tsena doveriya	Aliens Uncovered: UFOs Over Arizona	Previously Saved Version	Adabana
Interest_Tags	[Sci-Fi]	[Action, Comedy, Sci-Fi]	[Adult Animation, Anime, Space Sci-Fi, Action,...	[Romance, Sci-Fi]	[Dystopian Sci-Fi, Horror, Sci-Fi, Thriller]	[Dystopian Sci-Fi, Psychological Drama, Drama,...	[Drama, Sci-Fi]	[Documentary, Mystery, Sci-Fi]	[Romance, Sci-Fi, Thriller]	[Drama, Sci-Fi]
IMDB_Rating	N/A	2.5	N/A	9.1	4.9	6.9	5.6	8.4	5.5	N/A
Num_Votes	N/A	446	N/A	493	N/A	N/A	28	16	418	N/A
Popularity_Score	N/A	N/A	N/A	N/A	3,677	510	N/A	N/A	N/A	N/A
Director(s)	[Hatice Askin]	[Hayal Aslanzade, Tevekkul Mehreliyev]	[Naomichi Yamato, Harutoshi Fukui]	[Suguri Ravinderr]	[Galder Gaztelu-Urrutia]	[Fleur Fortune]	[Maksim Kuts]	[Clive Christopher]	[Kei Ishikawa]	[Sayaka Kai]
Writer(s)	[Hatice Askin]	[Tevekkul Mehreliyev]	[Writers, Harutoshi Fukui, Leiji Matsumoto, Yo...	[Uppara Suriya Raj]	[Writers, Galder Gaztelu-Urrutia, Pedro Rivero...	[Nell Garfath Cox, Dave Thomas, John Donnelly]	[Artyom Izmaylov]	[Clive Christopher, Preston Dennett, Ramses Ja...	[Kei Ishikawa, Brad Wright]	[Sayaka Kai]
Star(s)	[Stars, Boran Kuzum, Zeynep Tugçe Bayat, Osman...	[Stars, Aydemir Akbas, Safak Sezer, Çetin Altay]	[Houko Kuwashima, Daisuke Ono]	[Uppara Suriya Raj, Merina Singh, Saddam Hussain]	[Stars, Milena Smit, Hovik Keuchkerian, Natali...	[Stars, Alicia Vikander, Elizabeth Olsen, Hime...	[Stars, Andrey Bakharev, Aleksandr Bandurin, S...	[Clive Christopher, Preston Dennett, Ramses Ja...	[Stars, Yûko Araki, Hideaki Itô, Nualpanod Nat...	[Stars, Arata Iura, Kiko Mizuhara, Tôko Miura]
Num_User_Reviews	N/A	N/A	N/A	N/A	211	17	N/A	N/A	N/A	N/A
Num_Critic_Reviews	N/A	N/A	N/A	N/A	51	60	N/A	N/A	N/A	N/A
MetaScore	N/A	N/A	N/A	N/A	45	65	N/A	N/A	N/A	N/A
Streaming_Service	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Storyline	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	In a not so distant future where lifespan exte...
Tagline	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Genres	[Sci-Fi]	N/A	[Animation, Action, Drama, Sci-Fi]	N/A	N/A	N/A	N/A	N/A	N/A	[Drama, Sci-Fi]
Release_Date	NaN	September 27, 2024 (Turkey)	NaN	June 14, 2024 (India)	October 4, 2024 (United Kingdom)	April 3, 2025 (Germany)	January 13, 2024 (Russia)	February 6, 2024 (United States)	September 26, 2024 (United Kingdom)	NaN
Country_of_Origin	NaN	[Turkey]	NaN	[India]	[Spain]	[Germany]	[Russia]	[United States]	[Japan]	NaN
Language	NaN	[Turkish]	NaN	[Telugu]	[Spanish]	[English]	[Russian]	[English]	[Japanese, English]	NaN
Runtime	NaN	1 hour 16 minutes	NaN	1 hour 54 minutes	1 hour 39 minutes	1 hour 54 minutes	1 hour 23 minutes	1 hour 56 minutes	1 hour 57 minutes	NaN
Budget	NaN	NaN	NaN	NaN	NaN	$8,000,000 (estimated)	RUR 5,000,000 (estimated)	NaN	NaN	NaN
Gross US & Canada	NaN	NaN	NaN	NaN	NaN	$276,649	NaN	NaN	NaN	NaN
Opening weekend US & Canada	NaN	NaN	NaN	NaN	NaN	$152,905	NaN	NaN	NaN	NaN
Gross worldwide	NaN	NaN	NaN	NaN	NaN	$276,649	NaN	NaN	NaN	NaN

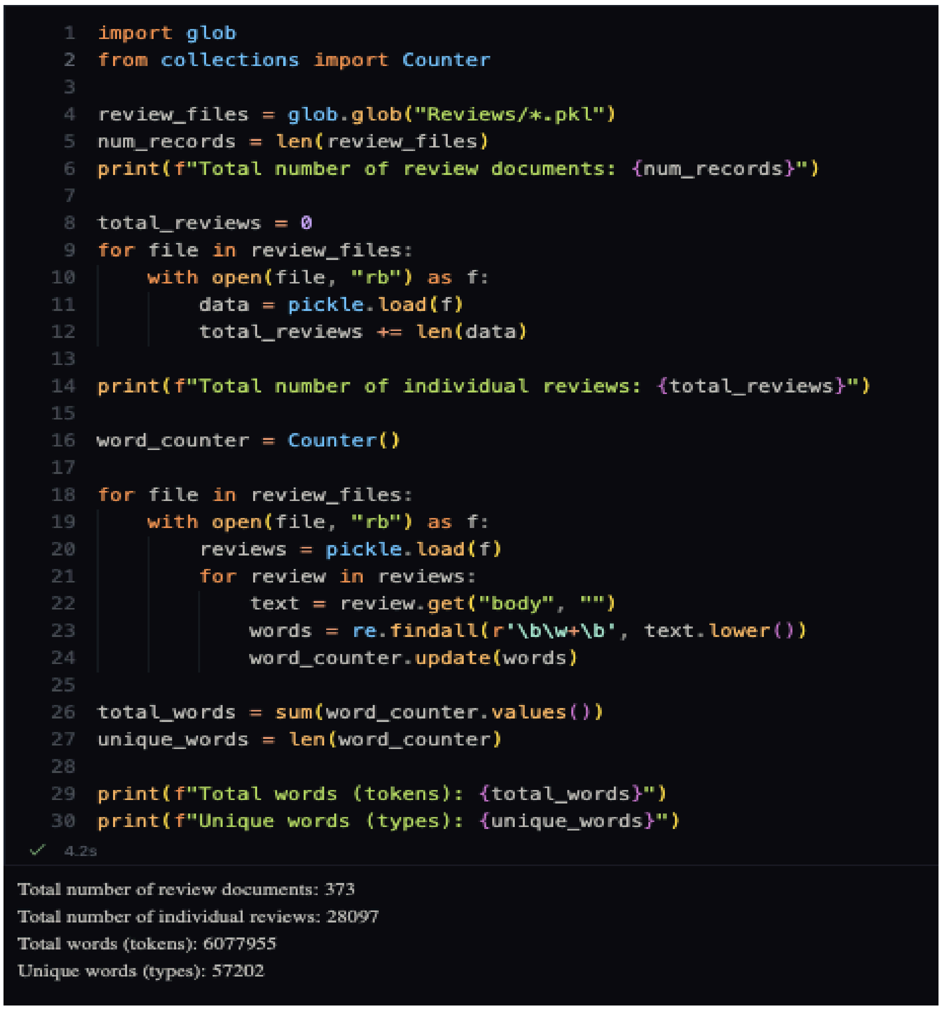
	...	Raduaa Returns	Alien	Badland Hunters	Justicia artificial	Friends of Sophia	Guy Manley: Super Spy	Mummy Shark	Bharatiya Superheroes the Origin	Operation War Thunder
Interest_Tags	...	[Sci-Fi]	[Drama, Sci-Fi]	[Disaster, Dystopian Sci-Fi, Gun Fu, Tragedy, ...	[Sci-Fi, Thriller]	[Sci-Fi]	[Action Epic, Buddy Comedy, Bumbling Detective...	[B-Horror, Supernatural Horror, Action, Horror...	[Sci-Fi]	[Comedy, Sci-Fi]
IMDB_Rating	...	5.5	5.6	5.9	5.6	5.5	4.4	2.0	8.9	5.3
Num_Votes	...	70	152	N/A	455	15	117	177	62	116
Popularity_Score	...	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Director(s)	...	[Nav Bajwa]	[Ivan Sosnin]	[Heo Myeong-haeng]	[Simón Casal]	[Alden Peters]	[David Andersson]	[Mark Polonia]	[Virat Vilas Pawar]	[James Peakman]
Writer(s)	...	[Nav Bajwa]	[Ivan Sosnin]	[Kim Bo-tong, Kwak Jae-min]	[Simón Casal, Victoriano Sierra Ferreiro]	[Alden Peters]	[David Andersson, Baltazar Ploteau]	[Bret McCormick]	[Virat Vilas Pawar]	[Writers, James Peakman, Andrew Simpson]
Star(s)	...	[Stars, Paramveer Singh, Nav Bajwa, Gurpreet G...	[Stars, Irma Arendt, Ulyana Fomina, Sergey Fyo...	[Stars, Ma Dong-seok, Kim Young-ah, Lee Hee-joon]	[Stars, Verónica Echegui, Alberto Ammann, Tama...	[Stars, Nana Visitor, Wesley Han, Vishaal Reddy]	[Stars, Baltazar Ploteau, Anton Sjölund, Milto...	[Stars, Jeff Kirkendall, Tim Hatch, Jamie Morgan]	[Stars, Samar Ayare, Sudhir Bawkar, Pranav Dah...	[Stars, Andrew Simpson, James Peakman, Danyan ...
Num_User_Reviews	...	N/A	N/A	41	N/A	N/A	N/A	N/A	N/A	N/A
Num_Critic_Reviews	...	N/A	N/A	48	N/A	N/A	N/A	N/A	N/A	N/A
MetaScore	...	N/A	N/A	54	N/A	N/A	N/A	N/A	N/A	N/A
Streaming_Service	...	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Storyline	...	raduaa return is written and directed by nav b...	N/A	N/A	N/A	In a dystopian future, citizens live and work ...	N/A	N/A	N/A	N/A
Tagline	...	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Genres	...	[Sci-Fi]	N/A	N/A	N/A	[Sci-Fi]	N/A	N/A	N/A	N/A
Release_Date	...	November 22, 2024 (India)	October 24, 2024 (Russia)	January 26, 2024 (United Kingdom)	September 13, 2024 (Spain)	March 14, 2024 (United Kingdom)	September 13, 2024 (Sweden)	October 8, 2024 (United States)	May 4, 2024 (India)	September 26, 2024 (United Kingdom)
Country_of_Origin	...	[India]	[Russia]	[South Korea]	[Spain]	[United States]	[Sweden]	[United States]	[India]	[United Kingdom]
Language	...	[Punjabi]	[Russian]	[Korean]	[Spanish]	[English]	[English]	[English]	N/A	[English]
Runtime	...	2 hours 17 minutes	1 hour 13 minutes	1 hour 47 minutes	1 hour 38 minutes	25 minutes	1 hour 32 minutes	1 hour 6 minutes	N/A	1 hour 14 minutes
Budget	...	$750,000 (estimated)	NaN	$18,000,000 (estimated)	NaN	NaN	NaN	NaN	₹300,000 (estimated)	NaN
Gross US & Canada	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Opening weekend US & Canada	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Gross worldwide	...	$1,624	NaN	NaN	$137,341	NaN	$4	NaN	NaN	NaN

Figure 1 : Example of crawled Review Data

2.3 Corpus Statistics

Question 1.3: The numbers of records, words, and types (i.e., unique words) in the corpus

The final crawled corpus comprises a total of **373 review documents**, each corresponding to a unique IMDb movie. Within these documents, a combined **28,097 individual user reviews** were collected. After preprocessing the textual data, the corpus was found to contain approximately **6,077,955 word tokens**, reflecting the total number of words across all reviews. The vocabulary of the corpus is also notably diverse, consisting of **57,202 unique word types**. These figures were computed by iterating through each review, extracting the text content, and applying a regular-expression-based tokenizer to count both the total words and the number of distinct tokens, shown in Figure 5 below. These statistics provide insight into the scale and richness of the dataset, supporting its suitability for a range of natural language processing and information retrieval applications.



```python
import glob
from collections import Counter

review_files = glob.glob("Reviews/*.pkl")
num_records = len(review_files)
print(f"Total number of review documents: {num_records}")

total_reviews = 0
for file in review_files:
    with open(file, "rb") as f:
        data = pickle.load(f)
        total_reviews += len(data)

print(f"Total number of individual reviews: {total_reviews}")

word_counter = Counter()

for file in review_files:
    with open(file, "rb") as f:
        reviews = pickle.load(f)
        for review in reviews:
            text = review.get("body", "")
            words = re.findall(r'\b\w+\b', text.lower())
            word_counter.update(words)

total_words = sum(word_counter.values())
unique_words = len(word_counter)

print(f"Total words (tokens): {total_words}")
print(f"Unique words (types): {unique_words}")
```

```
Total number of review documents: 373
Total number of individual reviews: 28097
Total words (tokens): 6077955
Unique words (types): 57202
```

Figure 5 : Code used to generate summary information

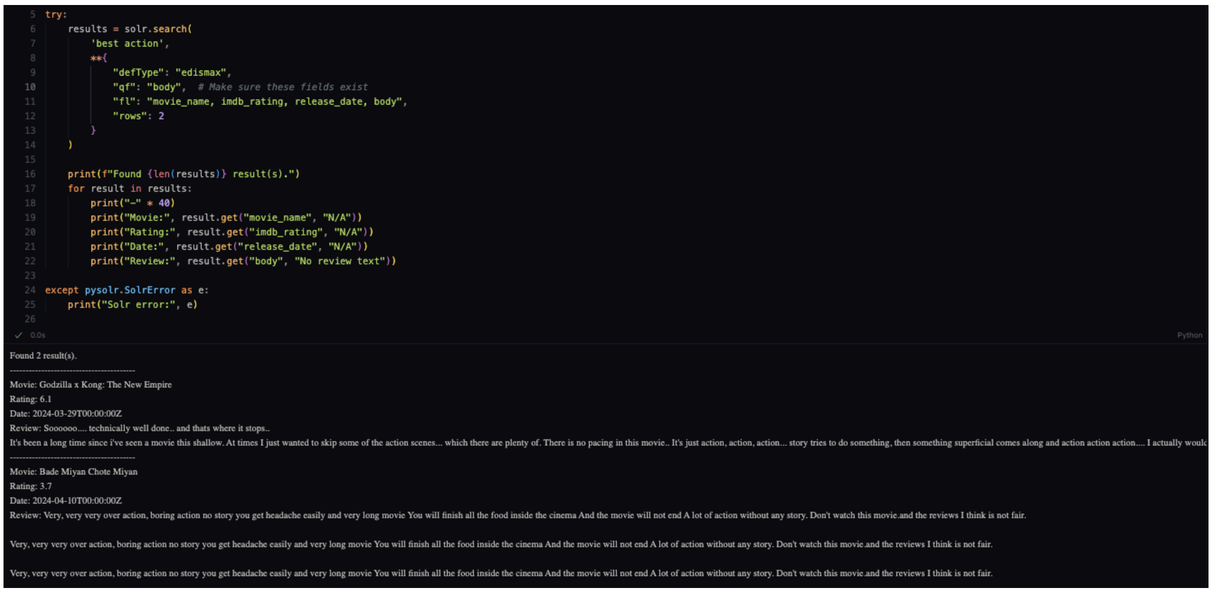
3. Indexing and Querying of Text Corpus

3.1 Information Retrieval from crawled corpus

Question 1.2: What kind of information users might like to retrieve from your crawled corpus (i.e., applications), with sample queries

The crawled IMDb corpus is designed to support both full-text retrieval and structured, metadata-driven querying, thereby enabling a wide array of practical applications. One prominent use case lies in sentiment analysis, where user-generated reviews can be mined to track evolving audience sentiment towards specific themes, characters, or genres in 2024 science fiction films. This can help uncover broader opinion trends over time or in response to specific releases.

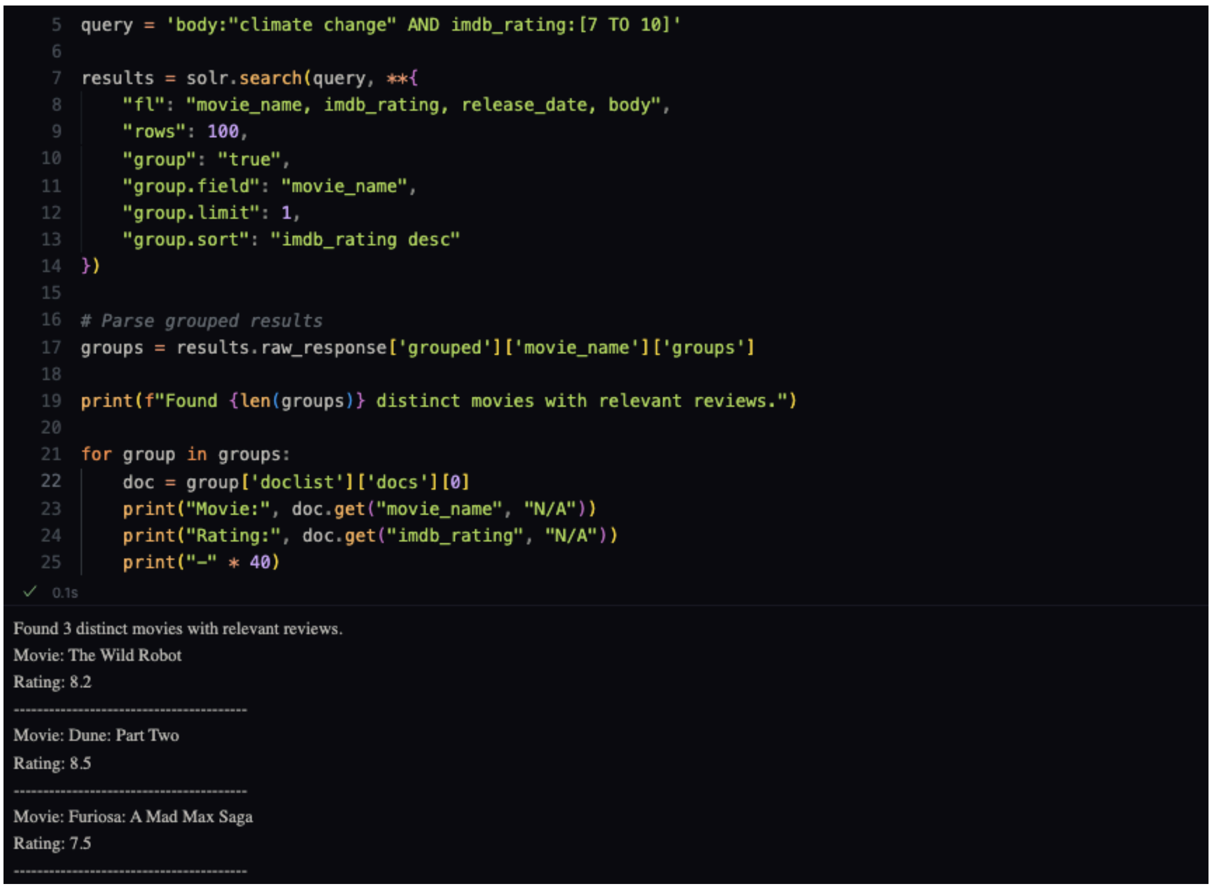
To illustrate these capabilities, several sample queries can be considered. A user searching for "best action" would retrieve highly-rated reviews that mention both the word "best" and references to "action," ranking results based on textual relevance and rating metrics. This example is shown in Figure 2 below. Another query like body:"climate change" AND imdb_rating:[7 TO 10] would identify well-received films whose reviews discuss environmental or climate-related themes. This is shown in Figure 3 below. Additionally, the autocomplete feature allows for intuitive search experiences; for instance, typing "dea" into the movie title search bar would dynamically suggest results such as *Dead Dead Demon's Dededede Destruction* and *Deadpool & Wolverine 2024*, based on prefix matching using an **EdgeNGram** tokenizer. This is shown in Figure 4 below.



```python
try:
    results = solr.search(
        'best action',
        **{
            "defType": "edismax",
            "qf": "body",  # Make sure these fields exist
            "fl": "movie_name, imdb_rating, release_date, body",
            "rows": 2
        }
    )

    print(f"Found {len(results)} result(s).")
    for result in results:
        print("-" * 40)
        print("Movie:", result.get("movie_name", "N/A"))
        print("Rating:", result.get("imdb_rating", "N/A"))
        print("Date:", result.get("release_date", "N/A"))
        print("Review:", result.get("body", "No review text"))

except pysolr.SolrError as e:
    print("Solr error:", e)
```

```
Found 2 result(s).
----------------------------------------
Movie: Godzilla x Kong: The New Empire
Rating: 6.1
Date: 2024-03-29T00:00:00Z
Review: Soooooo.... technically well done.. and thats where it stops..
It's been a long time since i've seen a movie this shallow. At times I just wanted to skip some of the action scenes... which there are plenty of. There is no pacing in this movie.. It's just action, action, action... story tries to do something, then something superficial comes along and action action action.... I actually woul
----------------------------------------
Movie: Bade Miyan Chote Miyan
Rating: 3.7
Date: 2024-04-10T00:00:00Z
Review: Very, very very over action, boring action no story you get headache easily and very long movie You will finish all the food inside the cinema And the movie will not end A lot of action without any story. Don't watch this movie.and the reviews I think is not fair.

Very, very very over action, boring action no story you get headache easily and very long movie You will finish all the food inside the cinema And the movie will not end A lot of action without any story. Don't watch this movie.and the reviews I think is not fair.

Very, very very over action, boring action no story you get headache easily and very long movie You will finish all the food inside the cinema And the movie will not end A lot of action without any story. Don't watch this movie.and the reviews I think is not fair.
```

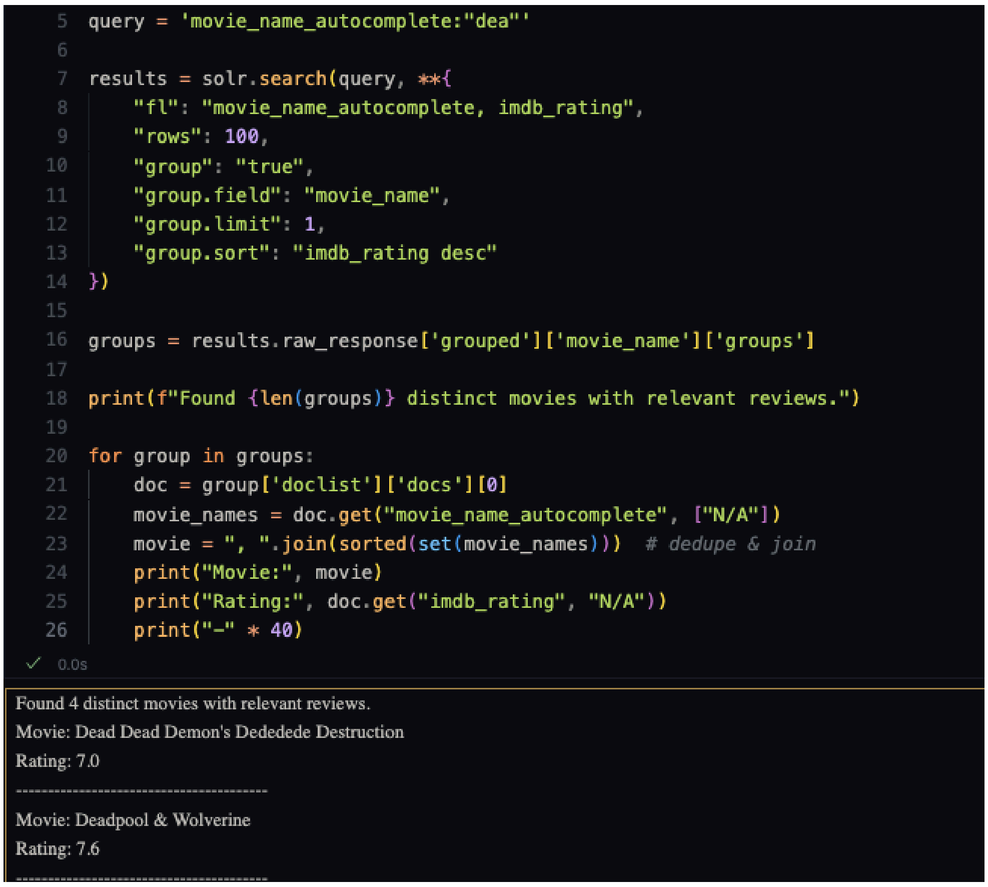
Figure 2 : Example of Query with “best action”



```python
query = 'body:"climate change" AND imdb_rating:[7 TO 10]'

results = solr.search(query, **{
    "fl": "movie_name, imdb_rating, release_date, body",
    "rows": 100,
    "group": "true",
    "group.field": "movie_name",
    "group.limit": 1,
    "group.sort": "imdb_rating desc"
})

# Parse grouped results
groups = results.raw_response['grouped']['movie_name']['groups']

print(f"Found {len(groups)} distinct movies with relevant reviews.")

for group in groups:
    doc = group['doclist']['docs'][0]
    print("Movie:", doc.get("movie_name", "N/A"))
    print("Rating:", doc.get("imdb_rating", "N/A"))
    print("-" * 40)
```

```
Found 3 distinct movies with relevant reviews.
Movie: The Wild Robot
Rating: 8.2
----------------------------------------
Movie: Dune: Part Two
Rating: 8.5
----------------------------------------
Movie: Furiosa: A Mad Max Saga
Rating: 7.5
----------------------------------------
```

Figure 3 : Example of “climate change” AND “imdb_rating : 7 TO 10



```python
query = 'movie_name_autocomplete:"dea"'

results = solr.search(query, **{
    "fl": "movie_name_autocomplete, imdb_rating",
    "rows": 100,
    "group": "true",
    "group.field": "movie_name",
    "group.limit": 1,
    "group.sort": "imdb_rating desc"
})

groups = results.raw_response['grouped']['movie_name']['groups']

print(f"Found {len(groups)} distinct movies with relevant reviews.")

for group in groups:
    doc = group['doclist']['docs'][0]
    movie_names = doc.get("movie_name_autocomplete", ["N/A"])
    movie = ", ".join(sorted(set(movie_names)))  # dedupe & join
    print("Movie:", movie)
    print("Rating:", doc.get("imdb_rating", "N/A"))
    print("-" * 40)
```

```
Found 4 distinct movies with relevant reviews.
Movie: Dead Dead Demon's Dededede Destruction
Rating: 7.0
----------------------------------------
Movie: Deadpool & Wolverine
Rating: 7.6
----------------------------------------
```

Figure 4 : Example of Autocomplete, with Query being ‘dea’

Together, these retrieval capabilities highlight the practical richness and adaptability of the constructed corpus for both research and real-world applications.

3.2 Solr: Backend

To enable effective search and retrieval across the IMDb movie review corpus, Apache Solr was adopted as the backend indexing and querying engine. Solr was chosen due to its flexible schema design, real-time search performance, and support for features such as autocomplete, faceted search, and fuzzy matching. The heterogeneous nature of the dataset—comprising both unstructured review text and structured metadata—necessitated a carefully engineered schema and a preprocessing pipeline to ensure consistency and relevance during both indexing and querying.

The schema, defined in Solr’s managed-schema, captures a wide range of fields. Textual fields such as title, body, and movie_name were assigned the text_general type, which employs a standard analyzer incorporating tokenization, synonym expansion (via *microtext.txt* and *synonyms_movies.txt*) (Hansonrobotics, n.d.), (*A List of English Stopwords*, n.d.), case normalization, stopword removal, and Porter stemming. These transformations ensured that terms were semantically aligned and linguistically normalized, supporting recall-oriented full-text search. During query time, synonym expansion was reinforced using **SynonymGraphFilterFactory**, allowing queries such as “sci-fi” to match “science fiction” seamlessly.

To implement real-time search suggestions, the schema defined several autocomplete fields using the *text_autocomplete* type. These fields—including *movie_name_autocomplete*, *directors_autocomplete*, *stars_autocomplete*, *writers_autocomplete*, *country_autocomplete*, and *release_country_autocomplete*—utilized an **EdgeNGram** tokenizer on the index side with a minimum and maximum gram size of 2 and 15, respectively. Lowercase filtering was applied to support case-insensitive prefix matching. Corresponding copyField directives ensured that autocomplete fields were automatically populated from their source fields (e.g., *movie_name* to *movie_name_autocomplete*), without needing to explicitly set values at ingestion time.

Structured numerical data, such as *rating*, *imdb_rating*, *budget*, *gross_worldwide*, *runtime_minutes*, *metascore*, and *user_ratings* (positive and negative), were indexed using Solr's *pfloat* type to support numerical filtering and range queries. Fields like release_date were parsed into ISO 8601 format and indexed using the *pdates* type for temporal queries. MultiValued metadata fields such as *directors*, *writers*, *stars*, *language*, and *country_of_origin* allowed multiple entries per field and were assigned the *text_general* type for analyzable, facetable search.

During the indexing process, the raw IMDb data underwent a series of preprocessing steps to ensure format compatibility and semantic clarity. Textual fields were cleaned by removing HTML tags, escaping special characters, and stripping excess whitespace. Fields such as runtime were parsed from strings like "2 hours 13 minutes" into numeric values in minutes. Similarly, budgets and revenues with currency symbols and comma separators were converted into float values. A unique document ID was generated for each review using an MD5 hash of the concatenated movie_name, body, and a timestamp, ensuring consistent document-level uniqueness, even for repeated or similar entries.

Documents were indexed in batches of 500 to balance throughput and reduce API load on the Solr server. The ingestion process validated required fields like id and ensured each document conformed to schema constraints.

On the querying side, Solr's ***edismax*** query parser was employed to enhance interpretability of user queries. This parser allowed for weighted field-level querying, meaning that certain fields such as stars could be boosted to prioritize matches where actor names were involved. Query preprocessing included case normalization and whitespace trimming, while fuzzy search with the ~1 operator enabled tolerance for typographical errors. Queries involving autocomplete fields matched partial prefixes using the movie_name_autocomplete and related fields, producing real-time dropdown suggestions based on user input.

Altogether, the schema design, preprocessing logic, and query handling pipeline combined to form a robust information retrieval system, capable of delivering precise, fast, and semantically informed results across the IMDb movie review corpus.

3.3 File Location

The files that are used in the Indexing component of this project are organized within the "solr-9.8.1" folder available via the GitHub repository. This directory contains the following key files:

- **moviereviewscore/conf/managed-schema:**

This file defines the Solr schema used to support the full-text indexing and querying of user review content. This includes fields such as *title*, *body*, and *rating*, in addition to structured metadata like *movie_name, imdb_rating*, and *runtime_minutes*. Notably, it also includes autocomplete support via **EdgeNGram** filters on fields like *movie_name*, *directors*, *stars*, and *writers*. The schema employs the text_general type with tokenization, stemming, lowercasing, and synonym expansion to ensure robust handling of sentiment and emotion classification tasks over user-generated review content. This schema supports *Part B – Indexing, Question 2,* by enabling fine-grained retrieval and analysis across both structured metadata and unstructured review texts.

- **metadatacore/conf/managed-schema:**

This file defines a more streamlined schema focused solely on the structured metadata of movies. It indexes core fields such as *movie_name*, *imdb_rating*, *directors*, *writers*, *stars*, *release_date*, *budget*, and *gross_worldwide*, without including review text. Like the moviereviewscore schema, it supports autocomplete through **EdgeNGram**-enhanced fields. This schema serves metadata-centric search and retrieval needs, providing efficient access to key factual attributes for aggregation, filtering, and faceted browsing, also supporting *Part B – Indexing, Question 2*, by enabling efficient indexing and retrieval of structured movie metadata, facilitating classification and evaluation tasks that rely on factual attributes such as cast, crew, runtime, and commercial performance—independent of user-generated review content.

3.4 User Interface (UI) for Querying

Question 2: Design a simple UI to allow users to access your system in a simple way.

3.4.1 Tools used

For querying, we built a simple web application using ReactJS for the frontend and NodeJS for the backend. The frontend communicates with the backend via HTTP requests, and the backend connects to the Solr server using Node’s solr-client package. The frontend is designed with a component-based architecture for extendability.

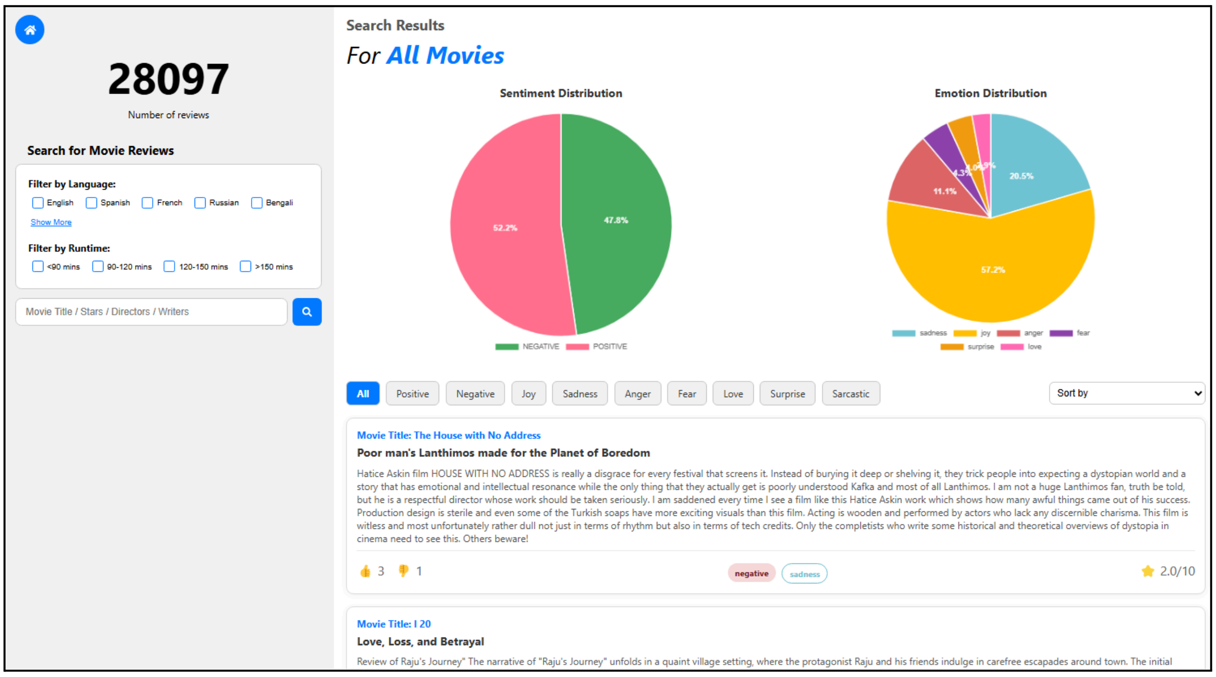
For more advanced retrieval, the application leveraged Solr's JSON Facet API, allowing multi-dimensional exploration of the dataset. Users could filter and pivot results across facets such as language, director, stars, and runtime—providing a responsive and exploratory interface over the indexed data.

To test the functionality of the opinion search engine, users may refer to the list of available movies provided in the **Movie List.pdf** (stored in Github Repository). These titles can be used as sample queries to explore and evaluate the system's sentiment and emotion retrieval capabilities.

3.4.2 Design overview

Figure 6 shows the homepage of the movie reviews web application. Main functions include:

- Querying movie titles, directors, writers or stars
- Filtering search results based on movie language and runtime
- Displaying all movie reviews
- Summarising movie reviews' sentiments and emotions in pie charts
- Filtering displayed reviews based on sentiment or emotion categories
- Sorting reviews based on ratings, upvotes and downvotes



Movie Title: The House with No Address

Poor man's Lanthimos made for the Planet of Boredom

Hatice Askin film HOUSE WITH NO ADDRESS is really a disgrace for every festival that screens it. Instead of burying it deep or shelving it, they trick people into expecting a dystopian world and a story that has emotional and intellectual resonance while the only thing that they actually get is poorly understood Kafka and most of all Lanthimos. I am not a huge Lanthimos fan, truth be told, but he is a respectful director whose work should be taken seriously. I am saddened every time I see a film like this Hatice Askin work which shows how many awful things came out of his success. Production design is sterile and even some of the Turkish soaps have more exciting visuals than this film. Acting is wooden and performed by actors who lack any discernible charisma. This film is witless and most unfortunately rather dull not just in terms of rhythm but also in terms of tech credits. Only the completists who write some historical and theoretical overviews of dystopia in cinema need to see this. Others beware!

3 1 2.0/10

Movie Title: I 20

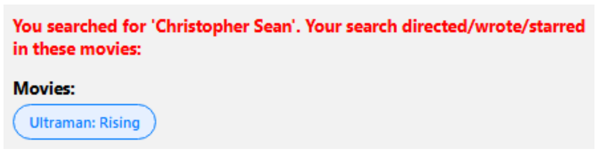
Love, Loss, and Betrayal

Review of Raju's Journey" The narrative of "Raju's Journey" unfolds in a quaint village setting, where the protagonist Raju and his friends indulge in carefree escapades around town. The initial

Figure 6: Homepage of web application

Feature 1: Search Form

The user can search for reviews by querying either movie title, stars, directors or writers. If a movie star, director or writer is queried, the application will retrieve the movie titles that the target starred in, directed, or wrote for respectively. If a movie title is queried, the application will directly return the movie reviews for that movie.



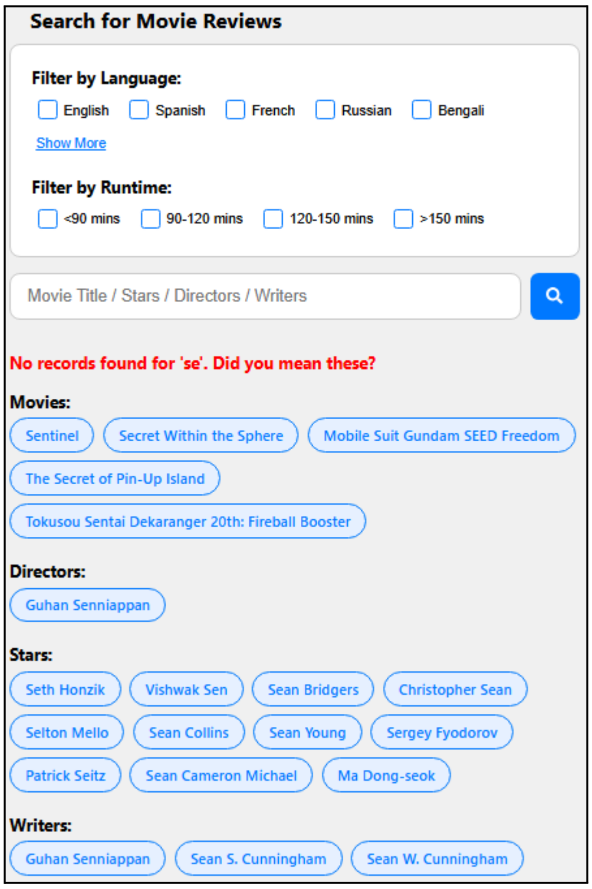
You searched for 'Christopher Sean'. Your search directed/wrote/starred in these movies:

Movies:

Ultraman: Rising

Figure 7: UI showing relevant movies for movie star 'Christopher Sean'

The search results can be narrowed down using search filters for movie language and runtime. Figures 8 and 9 show the difference between the suggestions before and after filters are applied.



Search for Movie Reviews

Filter by Language:
☐ English ☐ Spanish ☐ French ☐ Russian ☐ Bengali
Show More

Filter by Runtime:
☐ <90 mins ☐ 90-120 mins ☐ 120-150 mins ☐ >150 mins

Movie Title / Stars / Directors / Writers

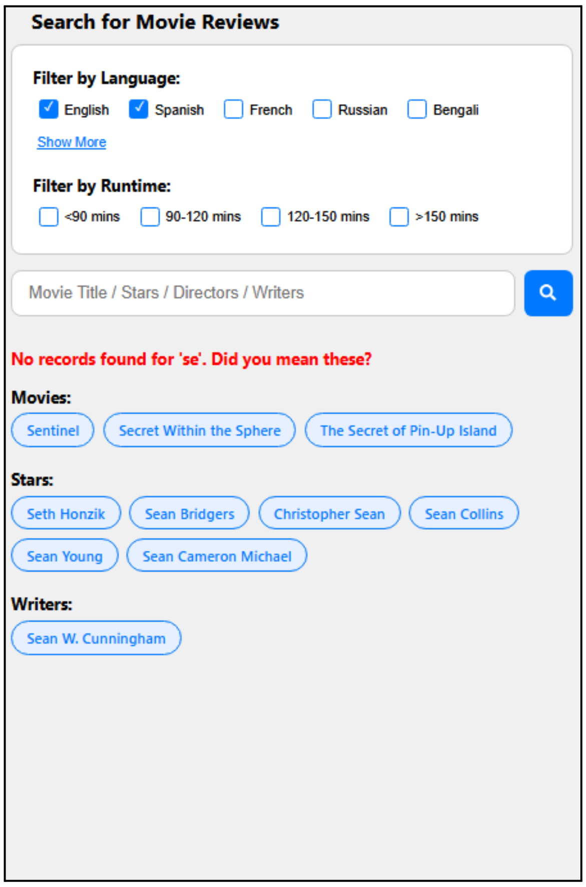
No records found for 'se'. Did you mean these?

Movies:
Sentinel | Secret Within the Sphere | Mobile Suit Gundam SEED Freedom | The Secret of Pin-Up Island | Tokusou Sentai Dekaranger 20th: Fireball Booster

Directors:
Guhan Senniappan

Stars:
Seth Honzik | Vishwak Sen | Sean Bridgers | Christopher Sean | Selton Mello | Sean Collins | Sean Young | Sergey Fyodorov | Patrick Seitz | Sean Cameron Michael | Ma Dong-seok

Writers:
Guhan Senniappan | Sean S. Cunningham | Sean W. Cunningham



Search for Movie Reviews

Filter by Language:
☑ English ☑ Spanish ☐ French ☐ Russian ☐ Bengali
Show More

Filter by Runtime:
☐ <90 mins ☐ 90-120 mins ☐ 120-150 mins ☐ >150 mins

Movie Title / Stars / Directors / Writers

No records found for 'se'. Did you mean these?

Movies:
Sentinel | Secret Within the Sphere | The Secret of Pin-Up Island

Stars:
Seth Honzik | Sean Bridgers | Christopher Sean | Sean Collins | Sean Young | Sean Cameron Michael

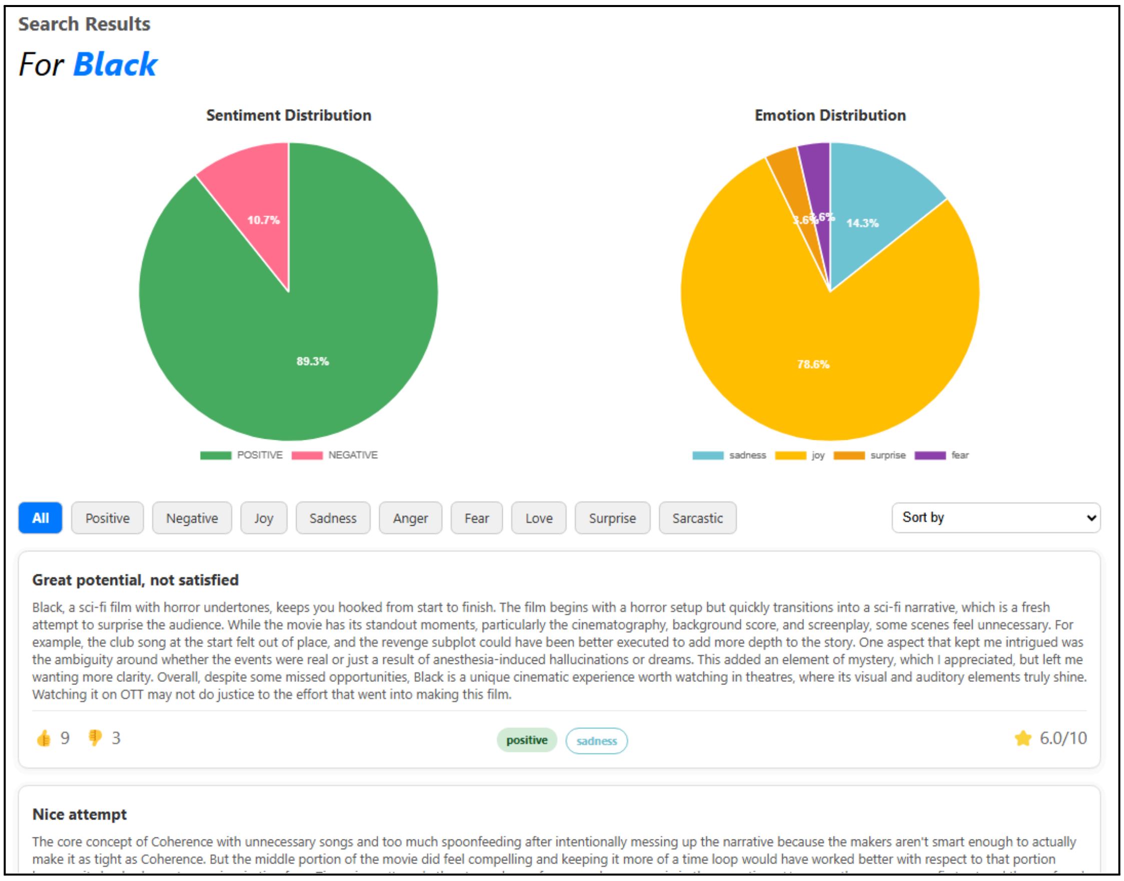
Writers:
Sean W. Cunningham

Figure 8 (Left): Search results for ‘se’ when no filters are applied

Figure 9 (Right): Search results for ‘se’ when language filters (English and Spanish) are applied

<u>Feature 2: Displaying Movie’s Reviews, Sentiments and Emotions</u>

Figure 10 below shows an example of the UI when a specific movie is chosen (in this case, ‘Black’). The sentiment and emotion distributions for the movie are shown on the pie charts. Each review is also labelled with the predicted sentiment and emotion.



Great potential, not satisfied

Black, a sci-fi film with horror undertones, keeps you hooked from start to finish. The film begins with a horror setup but quickly transitions into a sci-fi narrative, which is a fresh attempt to surprise the audience. While the movie has its standout moments, particularly the cinematography, background score, and screenplay, some scenes feel unnecessary. For example, the club song at the start felt out of place, and the revenge subplot could have been better executed to add more depth to the story. One aspect that kept me intrigued was the ambiguity around whether the events were real or just a result of anesthesia-induced hallucinations or dreams. This added an element of mystery, which I appreciated, but left me wanting more clarity. Overall, despite some missed opportunities, Black is a unique cinematic experience worth watching in theatres, where its visual and auditory elements truly shine. Watching it on OTT may not do justice to the effort that went into making this film.

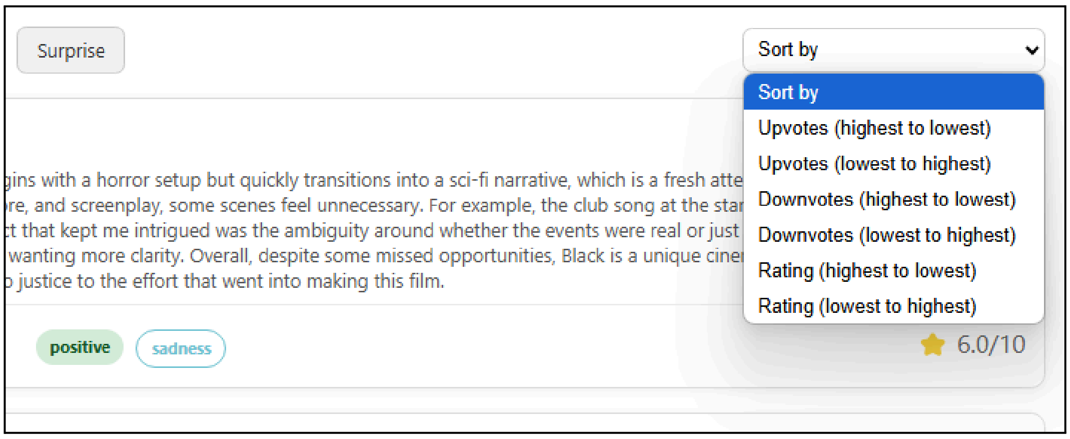
👍 9 👎 3 positive sadness ⭐ 6.0/10

Nice attempt

The core concept of Coherence with unnecessary songs and too much spoonfeeding after intentionally messing up the narrative because the makers aren't smart enough to actually make it as tight as Coherence. But the middle portion of the movie did feel compelling and keeping it more of a time loop would have worked better with respect to that portion

Figure 10: UI showing reviews when movie 'Black' is queried

Users can sort the reviews by Upvotes, Downvotes and Movie Ratings.



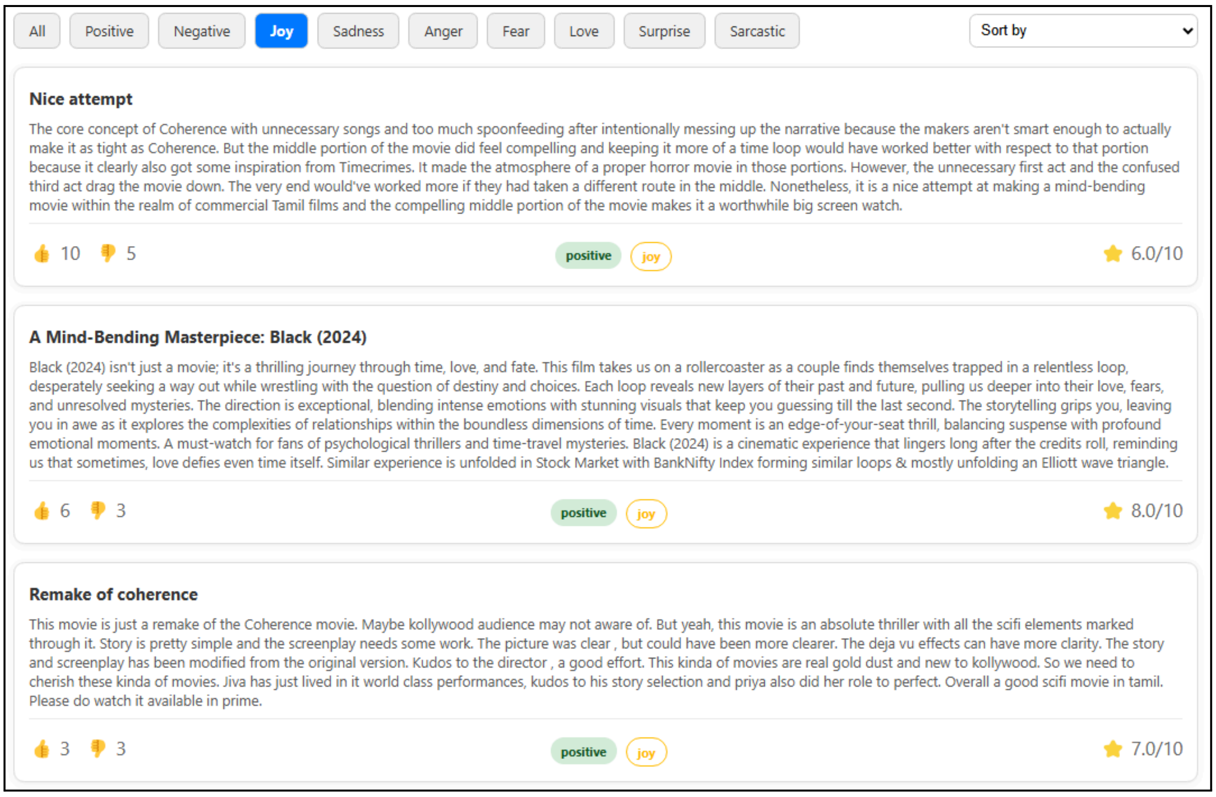
Surprise
Sort by
Sort by
Upvotes (highest to lowest)
Upvotes (lowest to highest)
Downvotes (highest to lowest)
Downvotes (lowest to highest)
Rating (highest to lowest)
Rating (lowest to highest)

gins with a horror setup but quickly transitions into a sci-fi narrative, which is a fresh atte
re, and screenplay, some scenes feel unnecessary. For example, the club song at the star
t that kept me intrigued was the ambiguity around whether the events were real or just
wanting more clarity. Overall, despite some missed opportunities, Black is a unique cine
justice to the effort that went into making this film.

positive sadness ⭐ 6.0/10

Figure 11: Sorting options for movie reviews

The sentiment/emotion tabs allow the user to filter reviews according to what they are interested in.



Nice attempt

The core concept of Coherence with unnecessary songs and too much spoonfeeding after intentionally messing up the narrative because the makers aren't smart enough to actually make it as tight as Coherence. But the middle portion of the movie did feel compelling and keeping it more of a time loop would have worked better with respect to that portion because it clearly also got some inspiration from Timecrimes. It made the atmosphere of a proper horror movie in those portions. However, the unnecessary first act and the confused third act drag the movie down. The very end would've worked more if they had taken a different route in the middle. Nonetheless, it is a nice attempt at making a mind-bending movie within the realm of commercial Tamil films and the compelling middle portion of the movie makes it a worthwhile big screen watch.

👍 10 👎 5 positive joy ⭐ 6.0/10

A Mind-Bending Masterpiece: Black (2024)

Black (2024) isn't just a movie; it's a thrilling journey through time, love, and fate. This film takes us on a rollercoaster as a couple finds themselves trapped in a relentless loop, desperately seeking a way out while wrestling with the question of destiny and choices. Each loop reveals new layers of their past and future, pulling us deeper into their love, fears, and unresolved mysteries. The direction is exceptional, blending intense emotions with stunning visuals that keep you guessing till the last second. The storytelling grips you, leaving you in awe as it explores the complexities of relationships within the boundless dimensions of time. Every moment is an edge-of-your-seat thrill, balancing suspense with profound emotional moments. A must-watch for fans of psychological thrillers and time-travel mysteries. Black (2024) is a cinematic experience that lingers long after the credits roll, reminding us that sometimes, love defies even time itself. Similar experience is unfolded in Stock Market with BankNifty Index forming similar loops & mostly unfolding an Elliott wave triangle.

👍 6 👎 3 positive joy ⭐ 8.0/10

Remake of coherence

This movie is just a remake of the Coherence movie. Maybe kollywood audience may not aware of. But yeah, this movie is an absolute thriller with all the scifi elements marked through it. Story is pretty simple and the screenplay needs some work. The picture was clear , but could have been more clearer. The deja vu effects can have more clarity. The story and screenplay has been modified from the original version. Kudos to the director , a good effort. This kinda of movies are real gold dust and new to kollywood. So we need to cherish these kinda of movies. Jiva has just lived in it world class performances, kudos to his story selection and priya also did her role to perfect. Overall a good scifi movie in tamil. Please do watch it available in prime.

👍 3 👎 3 positive joy ⭐ 7.0/10

Figure 12: Sentiment and Emotion categorisation for each review

3.5 Test Queries and System Performance Evaluation

Question 2: Write five queries, get their results, and measure the speed of the querying.

To measure the speed of querying, we used Postman software to send HTTP get requests to our backend. The time taken from sending the query, to searching on Solr, to processing on the backend is considered as part of the querying time. Results for 5 test queries are shown in the following table:

Test No.	Query Type	Query	Time Taken (ms)
1	Exact movie match, movie can be found in database	Black	10.89
2	Exact ‘director’ match, director found in database	Steve Garel	11.54
3	Incomplete query, query cannot be found in database	se	16.78
4	Incomplete query, query cannot be found in database	james	18.76
5	Query with language filters applied	se Filters: English, Spanish	14.44

Table 1: Querying Results

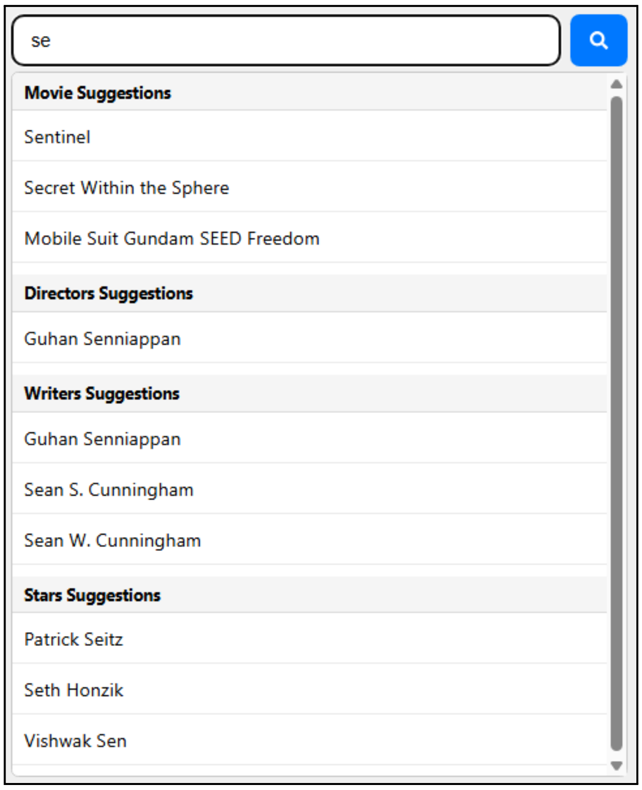
3.6 Innovations to Enhance Search

Question 3: Explore some innovations for enhancing the indexing and ranking.

In this section, we introduce the enhancements that we implemented for searching.

3.6.1 Autocomplete with Typeahead

To facilitate querying, the application suggests results while the user types. It saves time by allowing users to select from relevant suggestions instead of typing full queries, while also giving them insight into what's available in the database. Moreover, the feature helps the system by guiding users to make relevant queries, reducing the chances of invalid or broad searches. This improves performance, minimizes unnecessary load, and ensures more efficient use of backend resources.


se

Movie Suggestions
Sentinel
Secret Within the Sphere
Mobile Suit Gundam SEED Freedom

Directors Suggestions
Guhan Senniappan

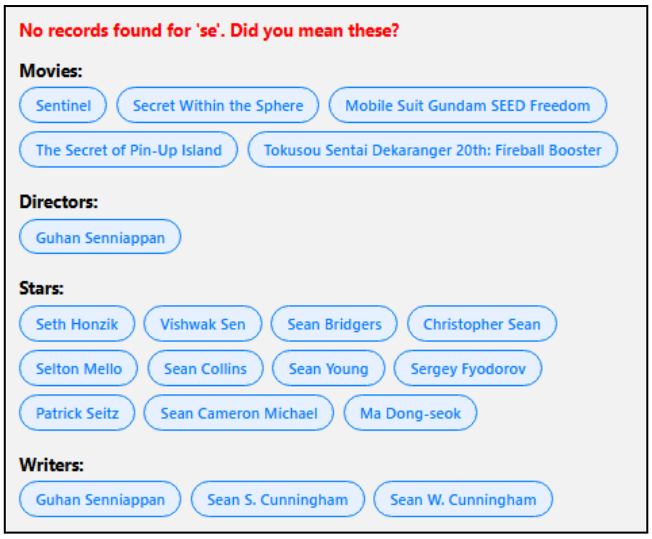
Writers Suggestions
Guhan Senniappan
Sean S. Cunningham
Sean W. Cunningham

Stars Suggestions
Patrick Seitz
Seth Honzik
Vishwak Sen

Figure 13: Categorised search suggestions while user types

3.6.2 Search Suggestions

When the user enters an incomplete query that does not match either a movie title, star, director or writer in the Solr core, the UI will show a list of suggestions that includes the user's query. The user can click on the suggestions to get their intended search results.



No records found for 'se'. Did you mean these?

Movies:

Sentinel | Secret Within the Sphere | Mobile Suit Gundam SEED Freedom | The Secret of Pin-Up Island | Tokusou Sentai Dekaranger 20th: Fireball Booster

Directors:

Guhan Senniappan

Stars:

Seth Honzik | Vishwak Sen | Sean Bridgers | Christopher Sean | Selton Mello | Sean Collins | Sean Young | Sergey Fyodorov | Patrick Seitz | Sean Cameron Michael | Ma Dong-seok

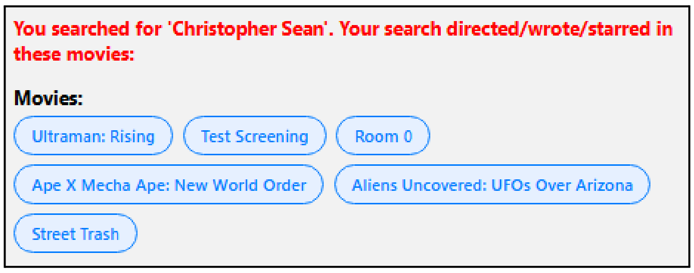
Writers:

Guhan Senniappan | Sean S. Cunningham | Sean W. Cunningham

Figure 14: Search suggestions when user's input is incomplete

Depending on which suggestions the user clicks on, the user may get routed to different results:

- If the user clicks on either a director, star or writer, the UI will display movie suggestions that the selected choice had directed for, starred in or participated as a writer (Figure 15).
- If the user clicks on a 'movie' suggestion, the UI will show all the reviews of the movie selected, along with the sentiments and emotions (Figure 16).

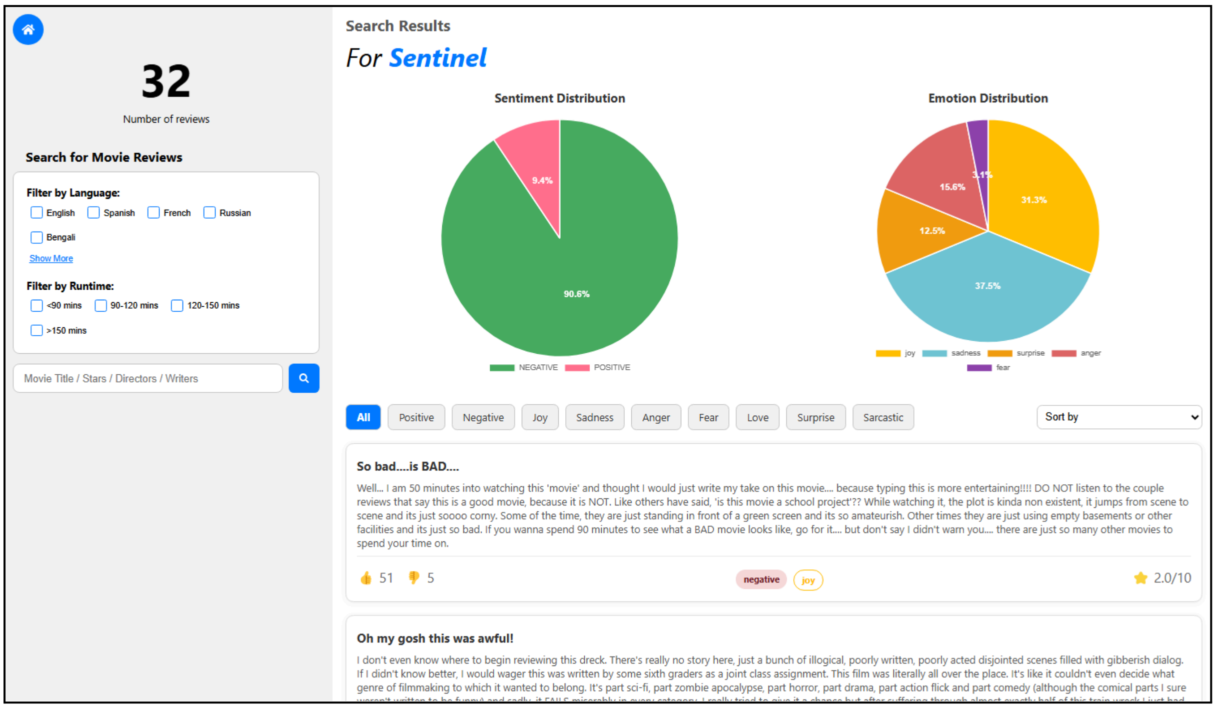


You searched for 'Christopher Sean'. Your search directed/wrote/starred in these movies:

Movies:

Ultraman: Rising | Test Screening | Room 0 | Ape X Mecha Ape: New World Order | Aliens Uncovered: UFOs Over Arizona | Street Trash

Figure 15: Search results if "Christopher Sean" is clicked on. UI displays movie suggestions that "Christopher Sean" starred in.



So bad....is BAD....

Well... I am 50 minutes into watching this 'movie' and thought I would just write my take on this movie.... because typing this is more entertaining!!!! DO NOT listen to the couple reviews that say this is a good movie, because it is NOT. Like others have said, 'is this movie a school project'?? While watching it, the plot is kinda non existent, it jumps from scene to scene and its just soooo corny. Some of the time, they are just standing in front of a green screen and its so amateurish. Other times they are just using empty basements or other facilities and its just so bad. If you wanna spend 90 minutes to see what a BAD movie looks like, go for it.... but don't say I didn't warn you.... there are just so many other movies to spend your time on.

51 5 negative joy 2.0/10

Oh my gosh this was awful!

I don't even know where to begin reviewing this dreck. There's really no story here, just a bunch of illogical, poorly written, poorly acted disjointed scenes filled with gibberish dialog. If I didn't know better, I would wager this was written by some sixth graders as a joint class assignment. This film was literally all over the place. It's like it couldn't even decide what genre of filmmaking to which it wanted to belong. It's part sci-fi, part zombie apocalypse, part horror, part drama, part action flick and part comedy (although the comical parts I sure

Figure 16: Search results if "Sentinel" is clicked on. UI displays movie reviews.

3.6.3 Search Result Visualisation Enhancement using Pie Charts

The sentiment and emotion distributions of a movie is shown using pie charts. The number of reviews that exist under each category will be displayed when the cursor is hovered over the respective category on the chart.

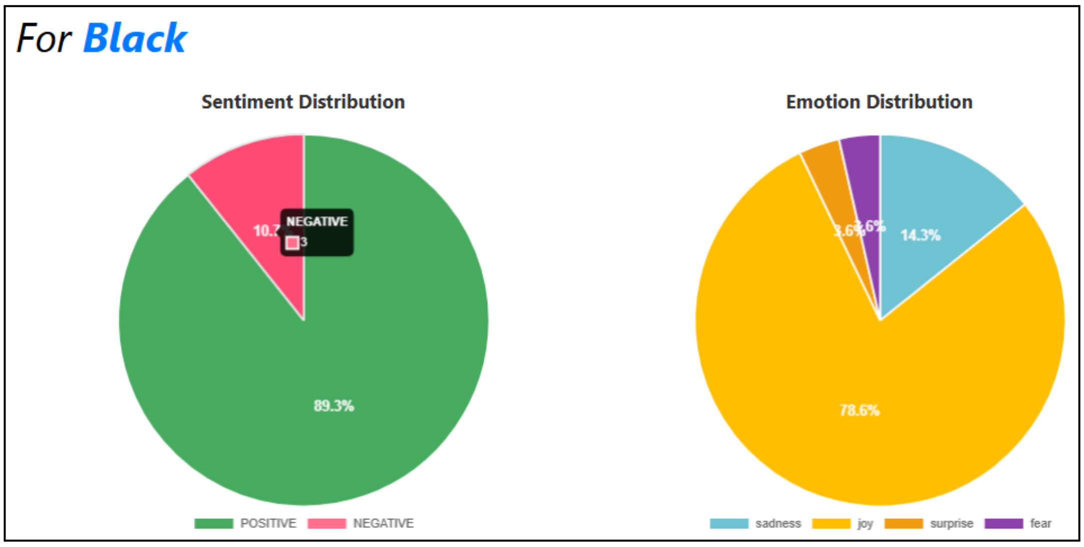


Figure 17: Pie Charts displayed for the Movie: Black

3.6.4 Search Filters

As introduced earlier in Section 3.2.2, search filters are implemented to give the user more flexibility in querying for the movies. Specifically, language and runtime filters can be applied to the search to narrow down the search results. Furthermore, as can be seen from the querying performance in section 3.3 (test cases 3 and 5), querying time reduced from 16.78ms to 14.44ms when filters were applied.

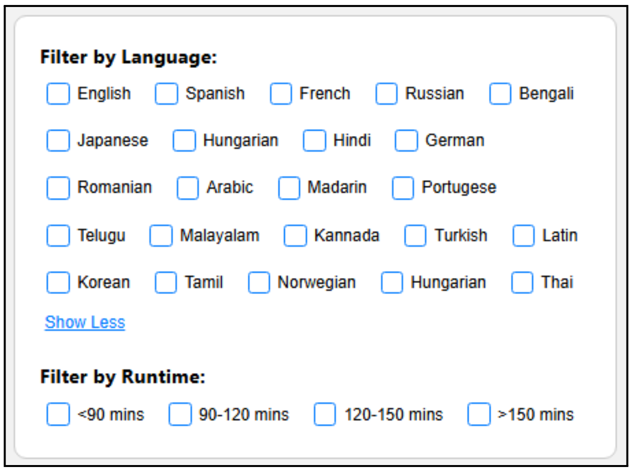


Figure 18: Search Filters Available

Test No.	Query	Time Taken (ms)
3	**se**	16.78
5	**se**, Filters: English, Spanish	14.44

Table 2: Querying performance for Test cases 3 and 5

4. Classification

4.1 Approaches to classification problems

Question 4: Justify your classification approach choice in relation to the state of the art

In this project, transformer-based models from the Hugging Face Transformers library were used to perform both **sentiment classification** and **emotion detection**. The selected model, accessed via AutoModelForSequenceClassification, is grounded in state-of-the-art transformer architectures like BERT or RoBERTa, which have achieved leading performance across a variety of NLP tasks.

This choice is justified by several key advantages:

1. **Contextualized Language Understanding:** Transformers can capture nuanced sentiment and emotional tone in a review, including subtle variations such as sarcasm or ambiguity.
2. **Dual Capability:** By fine-tuning or using models pretrained on emotion datasets (e.g., GoEmotions), the architecture supports multi-class emotion detection (e.g., joy, anger, sadness) alongside binary sentiment classification.
3. **Scalability & Flexibility:** Hugging Face's unified API supports multiple classification heads, allowing sentiment and emotion prediction to be handled within the same inference pipeline.
4. **Modern Best Practice:** Compared to rule-based or traditional machine learning models, transformer models achieve superior performance, adaptability across domains, and improved robustness on noisy or informal text such as user-generated movie reviews.

This approach ensures the model can not only distinguish between positive and negative sentiments, but also detect specific emotional expressions, offering richer insights into viewer responses. Given the nuanced linguistic patterns often found in movie reviews — such as sarcasm, irony, or emotional layering — a BERT-style classifier proves to be a highly suitable, informed, and scalable solution. Its strong accuracy, contextual interpretability, and ability to generalize across both sentiment and emotion tasks further justify its adoption over conventional or rule-based approaches.

4.2 Defining the classification task

This project defines two closely related classification tasks:

1. **Sentiment Classification**

   Goal: Determine whether a movie review expresses a positive or negative opinion.

   Type: Binary classification

   Input: Pre-processed movie reviews (e.g., "A stunning performance by the lead actor.")

   Output: Positive / Negative label

2. **Emotion Detection**

   Goal: Identify specific emotions conveyed in the review, such as joy, anger, sadness, disgust, or neutral.

   Type: Multi-class classification (single-label or multi-label depending on implementation)

   Model Behavior: The model generates emotion probabilities for predefined emotion categories, and the one with the highest score is selected.

3. **Data Characteristics:**

   The movie reviews were crawled and stored in a .pkl file, after which the relevant "body" text was extracted through preprocessing. These preprocessed entries were then passed through the classification pipeline for both sentiment analysis and emotion detection.

   To enable evaluation, a subset of 1,000 reviews was manually labeled, including both sentiment polarity (positive or negative) and emotion categories. The dataset contains a wide range of linguistic expressions, often blending multiple sentiments or emotional undertones. This complexity highlights the need for deep contextual modeling, as provided by transformer-based architectures.

This dual-task framework allows the system to provide both high-level sentiment polarity and fine-grained emotional tone, enabling a deeper understanding of user feedback on the movie.

4.3 Preprocessing Movie Reviews

Question 4: Discuss whether you had to preprocess data and why

4.3.1 Preprocessing Strategy and Scope

In this project, preprocessing was applied at two different stages, each serving a distinct purpose:

1) **For indexing and retrieval:**
   We utilized Apache Solr filters to normalize the text before building the search index. This included text transformations such as synonym mapping, case folding, stopword removal, and stemming. These filters enhanced the quality of document matching and retrieval during search operations.

2) **For sentiment and emotion classification:**
   We relied entirely on the built-in preprocessing mechanisms of the Hugging Face models. The tokenizer associated with each pre-trained model automatically handled all necessary formatting and normalization tasks during inference.

4.3.2 Built-In Tokenizer Preprocessing (Classification Phase)

The classification models used — **distilbert-base-uncased-finetuned-sst-2-english (sentiment)** and **bhadresh-savani/distilbert-base-uncased-emotion (emotion)** — are both based on DistilBERT's uncased architecture, and include the following tokenization steps as part of the Hugging Face AutoTokenizer:

- Lowercasing: Text is automatically converted to lowercase during tokenization.
- Subword Tokenization: Inputs are tokenized using the WordPiece algorithm.
- Special Tokens: Tokens like [CLS] and [SEP] are added according to the model's expected input format.
- Padding and Truncation: Inputs are padded and truncated to ensure consistent input length (e.g., max_length=512).

This allows raw IMDb reviews to be passed directly into the model pipeline without manual text cleaning — ensuring consistency with the model's original training procedure.

4.4 Classification strategies

Overview of classification strategy — BERT-based, Hugging Face, sentiment & emotion handled together.

4.4.1 Training dataset

- Crawled dataset in .pkl
- Manual labeling of 1,000 reviews with inter-annotator agreement
- Evaluation and random test setup

4.4.2 Dual-Model Classification: Sentiment and Emotion

To address both sentiment polarity and emotional nuance in movie reviews, this project implemented a dual-model classification pipeline, where two specialized models were used independently:

- **Sentiment Classification Model**

For binary sentiment analysis (positive / negative), we used the model: ***distilbert-base-uncased-finetuned-sst-2-english.*** This model is fine-tuned on the Stanford Sentiment Treebank (SST-2) and provides reliable sentiment classification for short-to-medium text.

- **Emotion Detection Model**

For multi-class emotion classification (e.g., joy, anger, sadness, etc.), we used: ***bhadresh-savani/distilbert-base-uncased-emotion.*** This model was trained on the GoEmotions dataset from Google and returns probabilities for a wide range of fine-grained emotional categories.

Each model was loaded using an initialize_model() function and tokenized independently before inference.

<u>**Justification for Separate Models**</u>

- Using task-specific models allows for **better performance and interpretability** within each domain.

- Emotion classification is inherently more complex and multi-dimensional, requiring a model trained on emotionally labeled datasets.
- Sentiment polarity can be more effectively captured using a binary classifier trained on sentiment-specific corpora.

4.4.3 File Location

The files that are used in the Classification component of this project are organized within the "Sentiment Analysis" folder available via the GitHub repository. This directory contains the following key files:

- **Final Sentiment Analysis.ipynb:** This Jupyter Notebook includes the complete classification workflow and is used for both sentiment & emotion classification, and the evaluation of the manually labeled dataset. This file addresses the requirements of *Part C – Classification*, specifically *Question 4* of the assignment.
- **analysis_all_movies (JSON and CSV formats):** These output files consist of movie reviews that have been annotated with both sentiment (positive, neutral, negative) and emotion tags, generated through the classification pipeline.
- **manual labelled data.csv:** This file contains a manually annotated dataset of 1,000 movie reviews. It serves as the ground truth for evaluating model performance and conducting the random accuracy assessment on unlabeled data.

4.5 Evaluations

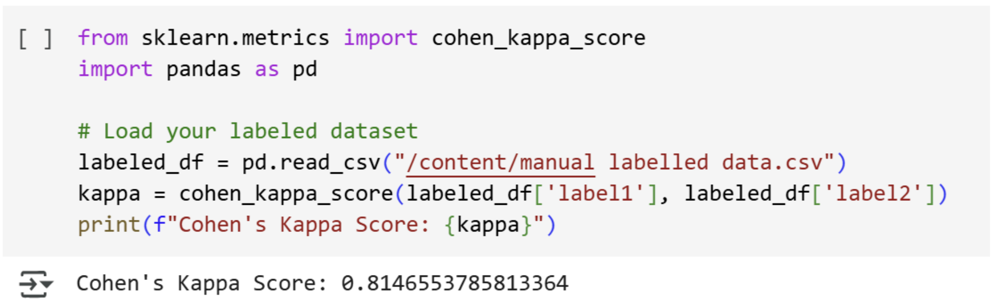
Question 4: Build an evaluation dataset by manually labeling at least 1,000 records with an inter-annotator agreement of at least 80%

4.5.1 Evaluation dataset

Since no official ground truth labels were available, we created an evaluation dataset by:

1. Randomly extracting 1,000 rows from the dataset;
2. Having 2 independent annotators manually annotate the data with sentiment (positive/negative) and emotion tags;
3. Achieve at least 80% of inter-annotator agreement using cohen_kappa_score to ensure reliability of dataset (Figure 19);
4. Retaining only data where both annotators agreed (Figure 20). Reviews with mismatched annotations were excluded to maintain evaluation integrity.

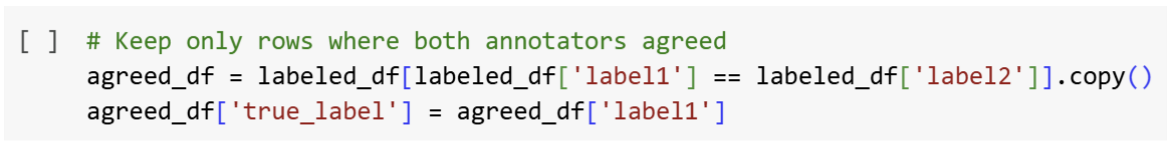
This subset serves as a proxy ground truth for evaluating model predictions. Examples of the evaluation test cases can be found in manual_labelled_data.csv.



```
[ ]  from sklearn.metrics import cohen_kappa_score
     import pandas as pd

     # Load your labeled dataset
     labeled_df = pd.read_csv("/content/manual labelled data.csv")
     kappa = cohen_kappa_score(labeled_df['label1'], labeled_df['label2'])
     print(f"Cohen's Kappa Score: {kappa}")
```

```
Cohen's Kappa Score: 0.8146553785813364
```

Figure 19: Cohen’s Kappa score for Evaluation set

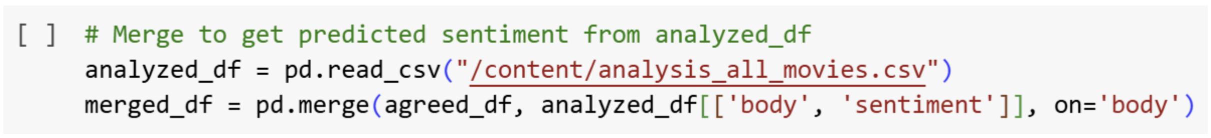


```
[ ]  # Keep only rows where both annotators agreed
     agreed_df = labeled_df[labeled_df['label1'] == labeled_df['label2']].copy()
     agreed_df['true_label'] = agreed_df['label1']
```

Figure 20: Code snippet showing that rows are only kept where annotators agreed

4.5.2 Evaluation: Metrics & Performance

To evaluate performance, the model’s predictions are compared against the final evaluation dataset.



```
[ ]  # Merge to get predicted sentiment from analyzed_df
     analyzed_df = pd.read_csv("/content/analysis_all_movies.csv")
     merged_df = pd.merge(agreed_df, analyzed_df[['body', 'sentiment']], on='body')
```

Figure 21: Code snippet showing merging of model's predictions to evaluation dataset

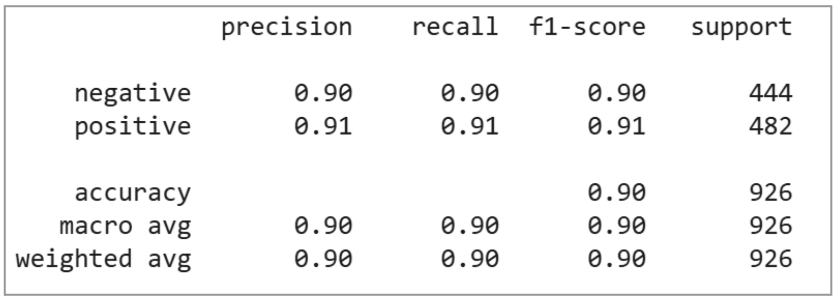
Question 4: Provide metrics such as precision, recall, and F-measure on such dataset

4.5.2.1 Evaluation Metrics

For evaluation, we used the following from sklearn.metrics**:**

- **Confusion matrix (**Provides breakdown of correct and incorrect predictions for each class)
- **Classification metrics**: Precision, recall, F1-score, and accuracy

The results were visualised using ConfusionMatrixDisplay.



```
              precision    recall  f1-score   support

    negative       0.90      0.90      0.90       444
    positive       0.91      0.91      0.91       482

    accuracy                           0.90       926
   macro avg       0.90      0.90      0.90       926
weighted avg       0.90      0.90      0.90       926
```

Figure 22: Sentiment Analysis Evaluation Results

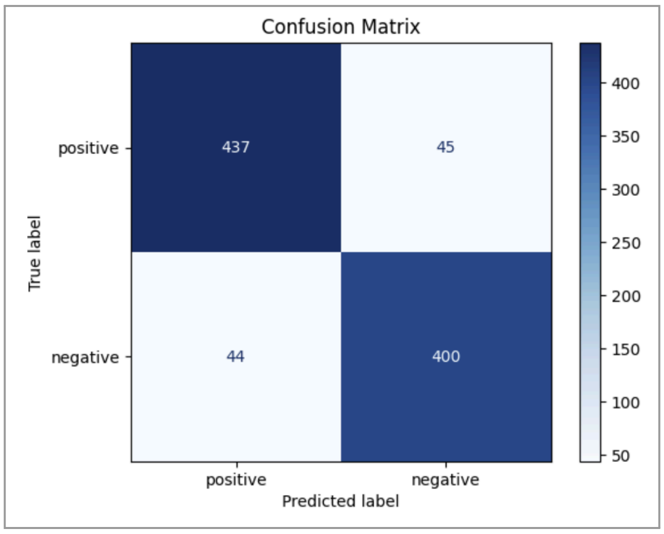


Figure 23: Confusion Matrix of Evaluation Results

4.5.2.2 Evaluation Performance

Our results show:

- The model achieved an overall accuracy of 90%, indicating strong alignment with human judgment.
- Both precision and recall were balanced across the positive and negative classes, with minimal disparity (0.90 vs 0.91).
- The confusion matrix shows near-symmetric performance:
    - Only 45 positive reviews were misclassified as negative
    - Only 44 negative reviews were misclassified as positive

The near-equal number of misclassifications in both directions suggests that the model is not significantly biased toward any one sentiment class.

The macro and weighted averages further confirm consistent performance across classes, supporting the model's generalizability.

4.5.3 Random Accuracy Test and Comparison

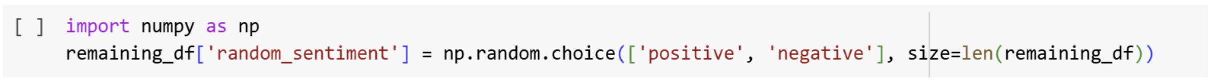
Question 4: Perform a random accuracy test on the rest of the data and discuss results

To further demonstrate that the model performs meaningfully and not by chance, we performed a random accuracy test on the remaining data — i.e., reviews that were not part of the manually labeled evaluation set.

Methodology

The manually labeled dataset (eval_df) was excluded from the full analyzed dataset to obtain remaining_df. We compared:

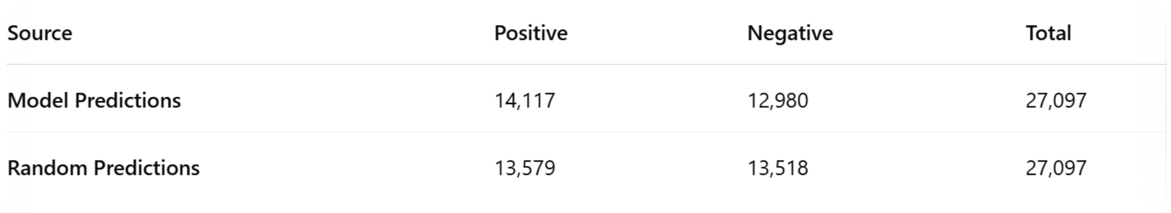
1. Model-predicted sentiment (from your BERT-based classifier)
2. Random sentiment predictions, assigned uniformly using NumPy



```
[ ]  import numpy as np
     remaining_df['random_sentiment'] = np.random.choice(['positive', 'negative'], size=len(remaining_df))
```

Figure 24: Code snippet showing random labelling of remaining dataset

Distribution Contribution



Source	Positive	Negative	Total
Model Predictions	14,117	12,980	27,097
Random Predictions	13,579	13,518	27,097

Figure 25: Sentiment Distribution of Model output and Randomly labelled

Since we lack ground truth labels for the remaining dataset, we cannot compute exact accuracy. However, based on our model's previously evaluated accuracy of 90%, we expect it to substantially outperform the random baseline on unseen data. The random predictor outputs an almost equal number of 'positive' and 'negative' predictions, but offers no real interpretability or reliability, unlike our model, which was trained on meaningful features and validated against human-labeled examples.

4.5.4 Result discussion and analysis

Question 4: Discuss performance metrics, e.g., speed, and scalability of the system

1. Does our model have a balanced sentiment prediction, or is it biased toward one class?

The model's sentiment predictions on the remaining unlabeled dataset show:

- Positive: 14,117 reviews
- Negative: 12,980 reviews

This indicates a slight skew toward the positive class, but the distribution is still relatively balanced, with only ~1,100 more positive predictions out of over 27,000 reviews. Therefore, the model does not exhibit significant class imbalance or bias in prediction.

2. How does the random prediction differ?

In contrast, the randomly generated sentiment labels show:

- Positive: 13,579
- Negative: 13,518

This distribution is almost perfectly balanced (~50/50 split), as expected from a random generator with equal class probability. However, this balance is purely coincidental and not based on any actual content or signal in the reviews.

3. Why is our model better than random (e.g., more informed, consistent, interpretable)?

While the random classifier produces a superficially balanced result, it completely lacks:

- Context-awareness: It does not consider the actual review content.
- Predictive value: It cannot capture tone, sentiment cues, or emotional signals.
- Consistency: Predictions are non-repeatable and unpredictable.

In contrast, the fine-tuned model uses pre-trained language understanding (via BERT) to make data-informed predictions. The model's earlier evaluation on 926 manually labeled reviews achieved:

- 90% accuracy
- F1-scores of 0.90+ for both classes

This confirms that the model significantly outperforms random guessing, providing interpretable and trustworthy outputs for downstream tasks like review analysis or sentiment tracking.

4.6 Enhanced Classification

Question 5: Explore some innovations for enhancing classification

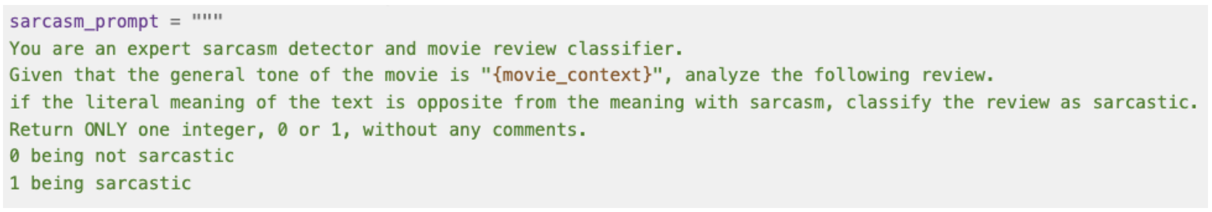
In this section, we will introduce more methods to enhance classification to improve user experience with higher accuracy. We utilize the output file *(analysis_all_movies)* generated from the Sentiment Analysis classification pipeline as the foundation for further enhancement in our innovative classification approach.

4.6.1 Enhanced classification: Sarcasm Detection

One of the common strategies people use when posting comments online is sarcasm. Sarcasm is the use of irony to say or do the opposite of what they really mean in order to mock or insult someone. This will cause the detected meaning to be opposite of the real meaning of what the user is trying to express. This report tried 2 methods to detect sarcasm.

Mistral model

Firstly, we tried to use a large language model 'mistral' using Ollama LLM. the prompt given to the model is:

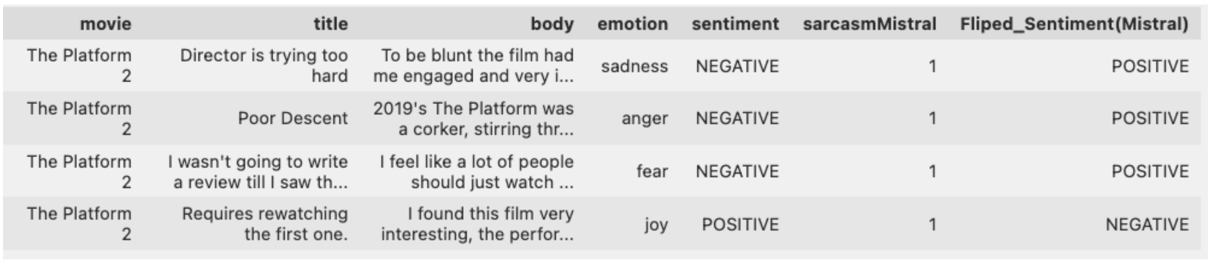


```
sarcasm_prompt = """
You are an expert sarcasm detector and movie review classifier.
Given that the general tone of the movie is "{movie_context}", analyze the following review.
if the literal meaning of the text is opposite from the meaning with sarcasm, classify the review as sarcastic.
Return ONLY one integer, 0 or 1, without any comments.
0 being not sarcastic
1 being sarcastic
```

Figure 26: Prompt used for sarcasm detection using Mistral

Where the model will analyse the review two times, with and without sarcasm. When the literal meaning is opposite from the meaning with sarcasm, the review will be marked as sarcastic.

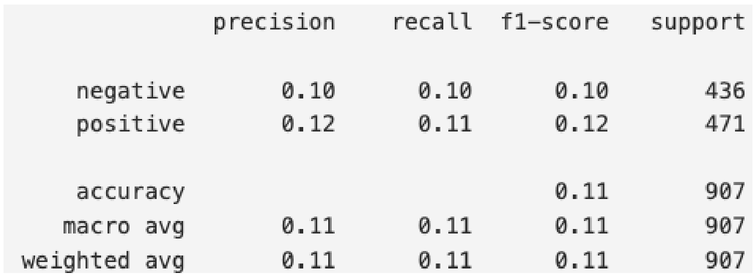
The output if this model is either 0 or 1, with 0 being not sarcastic and 1 being sarcastic. The next question is how can we utilize this data. As the output of the sentiment analysis only has 2 possibilities, positive and negative, we could just flip the sentiment result. For example, when a comment is detected as 'NEGATIVE' and with sarcasm, we will add another column 'Fliped_Sentiment(Mistral)' which will state 'POSITIVE'.



movie	title	body	emotion	sentiment	sarcasmMistral	Fliped_Sentiment(Mistral)
The Platform 2	Director is trying too hard	To be blunt the film had me engaged and very i...	sadness	NEGATIVE	1	POSITIVE
The Platform 2	Poor Descent	2019's The Platform was a corker, stirring thr...	anger	NEGATIVE	1	POSITIVE
The Platform 2	I wasn't going to write a review till I saw th...	I feel like a lot of people should just watch ...	fear	NEGATIVE	1	POSITIVE
The Platform 2	Requires rewatching the first one.	I found this film very interesting, the perfor...	joy	POSITIVE	1	NEGATIVE

Figure 27: Output for Sarcasm 1 - Mistral

With this, we performed the accuracy test again on 'Fliped_Sentiment(Mistral)' and the result is shown below.



```
              precision    recall  f1-score   support

    negative       0.10      0.10      0.10       436
    positive       0.12      0.11      0.12       471

    accuracy                           0.11       907
   macro avg       0.11      0.11      0.11       907
weighted avg       0.11      0.11      0.11       907
```

Figure 28: Classification report for Sarcasm 1 - Mistral

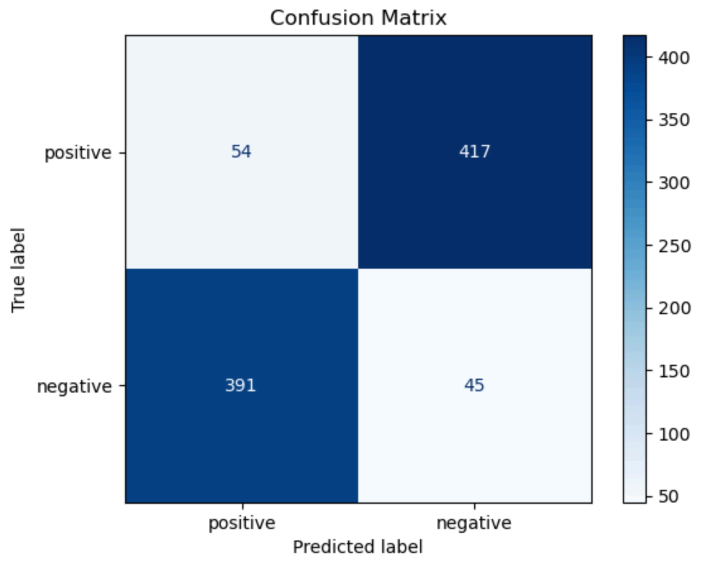
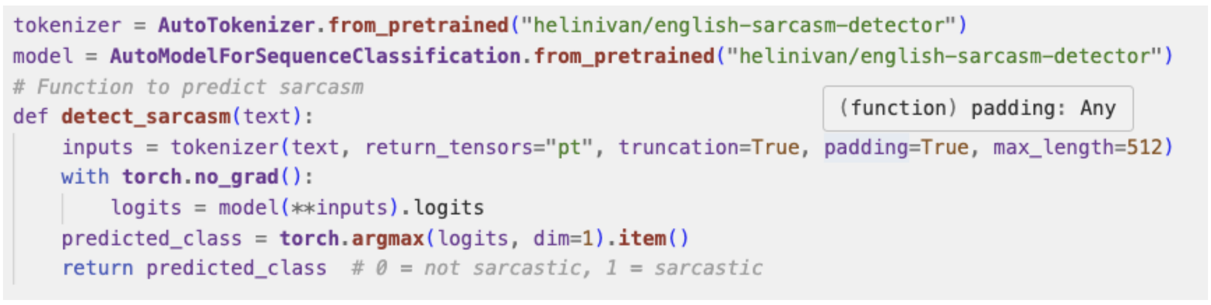


Figure 29: Confusion Matrix for Sarcasm 1 - Mistral

It can be seen that the F1 score dropped significantly from 0.90 and 0.91 to 0.10 and 0.12 for negative and positive groups respectively. The confusion matrix also shows low accuracy of 54 correctly detected as positive and 45 correctly detected as negative. This result shows that the model does not perform well when detecting sarcasm.

<u>Helinivan Sarcasm Model</u>

From the previous section, we had established that 'Mistral' is not a good model in detecting sarcasm due to the low accuracy. Thus, we tried another model 'helinivan/english-sarcasm-detector' from huggingface as shown below.

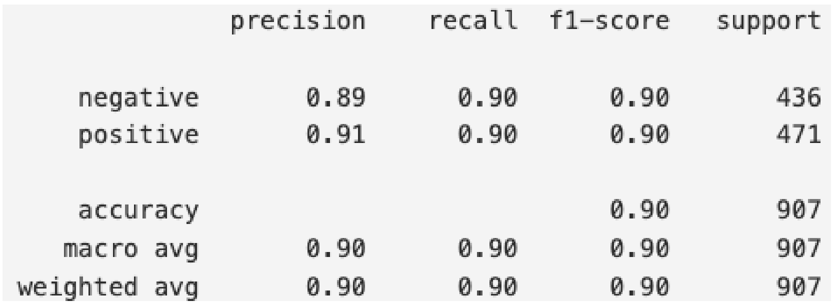


```
tokenizer = AutoTokenizer.from_pretrained("helinivan/english-sarcasm-detector")
model = AutoModelForSequenceClassification.from_pretrained("helinivan/english-sarcasm-detector")
# Function to predict sarcasm
def detect_sarcasm(text):                                            (function) padding: Any
    inputs = tokenizer(text, return_tensors="pt", truncation=True, padding=True, max_length=512)
    with torch.no_grad():
        logits = model(**inputs).logits
    predicted_class = torch.argmax(logits, dim=1).item()
    return predicted_class  # 0 = not sarcastic, 1 = sarcastic
```

Figure 30: Sarcasm 2 Model

This model follows the same output where it will return 0 if sarcasm is not detected and 1 if it is detected. After an output is produced for all reviews, we will flip the sentiment result again.

Next, accuracy testing is performed on the flipped sentiment.



```
              precision    recall  f1-score   support

    negative       0.89      0.90      0.90       436
    positive       0.91      0.90      0.90       471

    accuracy                           0.90       907
   macro avg       0.90      0.90      0.90       907
weighted avg       0.90      0.90      0.90       907
```

Figure 31: Classification report for Sarcasm 2

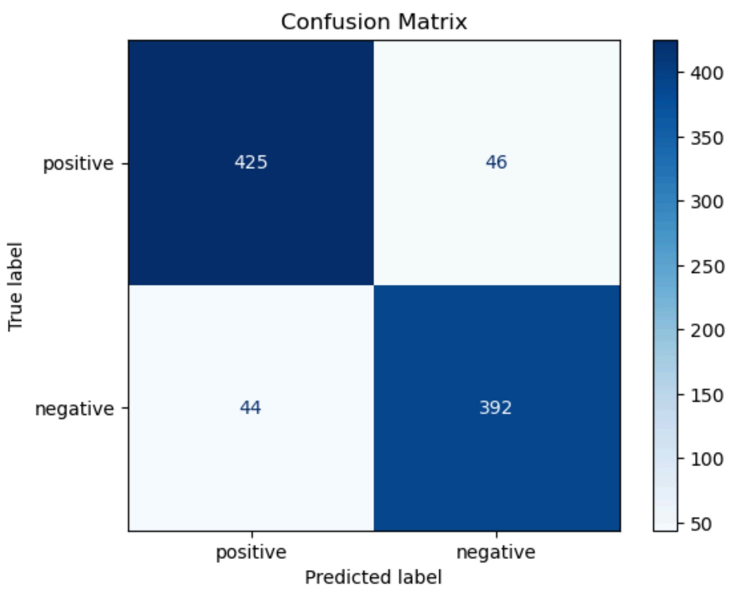


Figure 32: Confusion Matrix for Sarcasm 2

As seen from the result above, the F1 score for negative is 0.90, which is the same as the result without sarcasm detection. However, F1 score with sarcasm for positive is 0.90, slightly lower than 0.91, the score without sarcasm. The confusion matrix follows this finding as one more positive is detected as negative when using sarcasm as compared to without.

This shows that although the model is able to detect sarcasm more accurately than the 'mistral' mode, it does not improve the accuracy of the sentiment analysis. However, as there are only 1000 samples for accuracy testing, the result may not be precise and applicable to the whole population. Thus, as the accuracy does not vary much, we decided to label comments with sarcasm in our page, such that the user is able to quickly identify which comments are sarcastic and are opposite of its literal meaning.

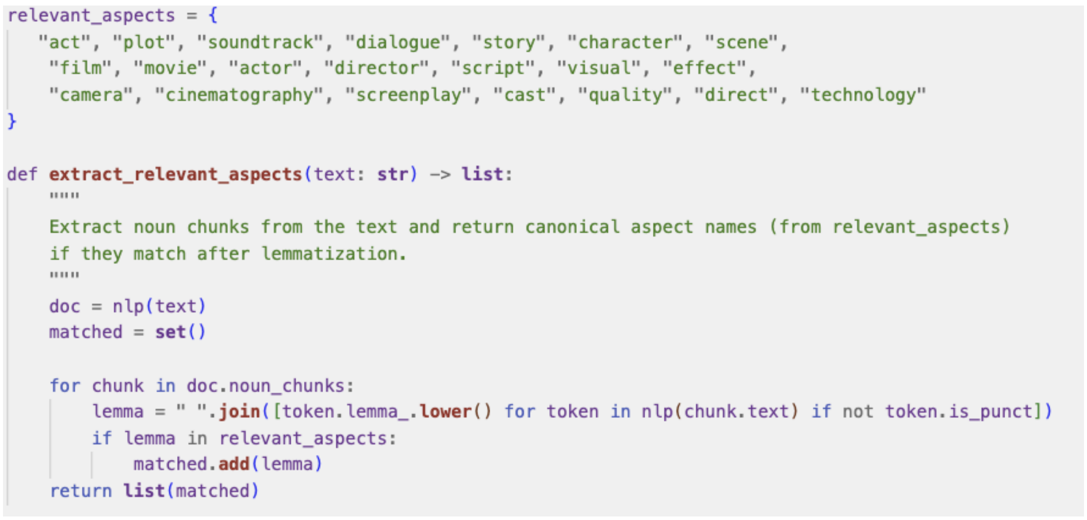
4.6.2 Fine-grained classification: ABSA

Apart from the emotion detection, we are also interested in the reason for users giving positive or negative reviews. This can be done by using Abstract Base Sentiment Analysis (ABSA).

<u>Deberta-v3-base-absa model</u>

In this model, our aim is to extract the aspects from each comment and analyse the sentiment of the extracted aspect. To ensure coherency, we have predefined some of the common aspects for movie reviews and will only perform analysis on these aspects.

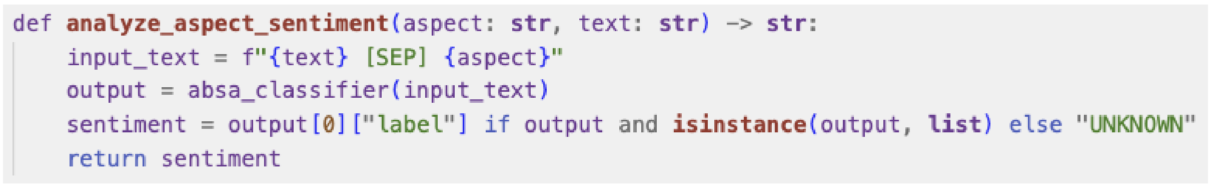
With this, the first step is to extract the aspect and only return if it is inside the predefined aspects. However, in order to match the aspect, we also need to lemmatize it before matching as shown below.



```
relevant_aspects = {
   "act", "plot", "soundtrack", "dialogue", "story", "character", "scene",
    "film", "movie", "actor", "director", "script", "visual", "effect",
    "camera", "cinematography", "screenplay", "cast", "quality", "direct", "technology"
}

def extract_relevant_aspects(text: str) -> list:
    """
    Extract noun chunks from the text and return canonical aspect names (from relevant_aspects)
    if they match after lemmatization.
    """
    doc = nlp(text)
    matched = set()

    for chunk in doc.noun_chunks:
        lemma = " ".join([token.lemma_.lower() for token in nlp(chunk.text) if not token.is_punct])
        if lemma in relevant_aspects:
            matched.add(lemma)
    return list(matched)
```

Figure 33: ABSA 1 preparation

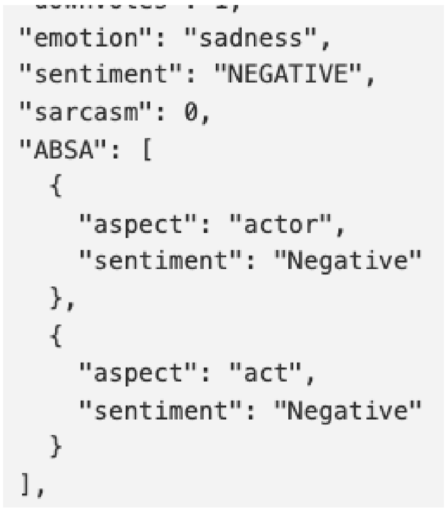
From the code, we can see that noun chunks are first extracted from the text using spaCy then compared with the predefined aspects. After aspects are identified, we will then perform sentiment analysis of the aspect as shown below.



```
def analyze_aspect_sentiment(aspect: str, text: str) -> str:
    input_text = f"{text} [SEP] {aspect}"
    output = absa_classifier(input_text)
    sentiment = output[0]["label"] if output and isinstance(output, list) else "UNKNOWN"
    return sentiment
```

Figure 34: ABSA 1 model

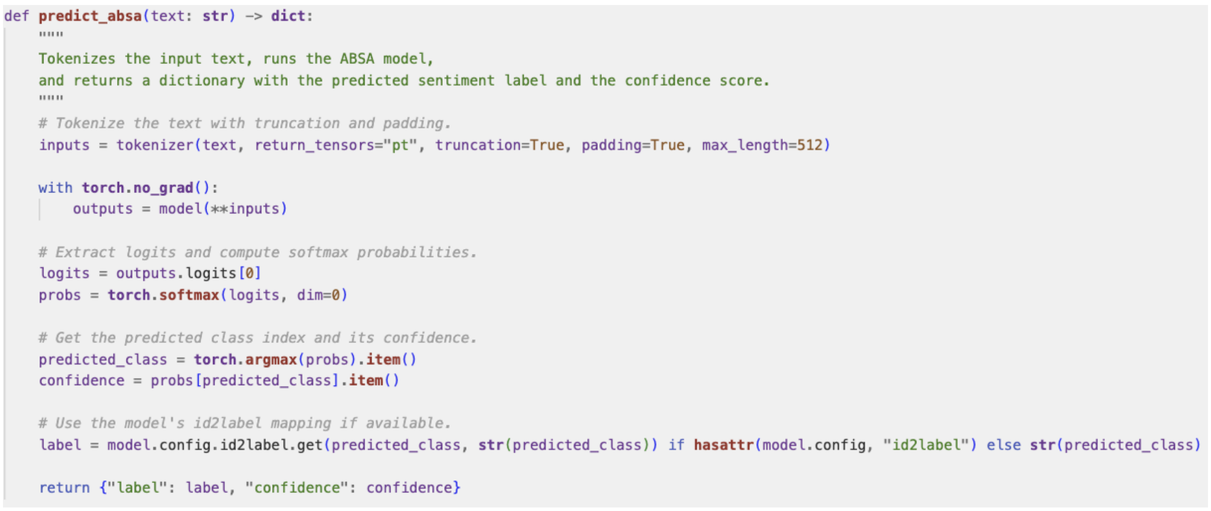
ABSA classifier will be run for the given aspect. The output for each aspect will be either positive or negative.



```
"emotion": "sadness",
"sentiment": "NEGATIVE",
"sarcasm": 0,
"ABSA": [
  {
    "aspect": "actor",
    "sentiment": "Negative"
  },
  {
    "aspect": "act",
    "sentiment": "Negative"
  }
],
```

Figure 35: Example output for ABSA 1

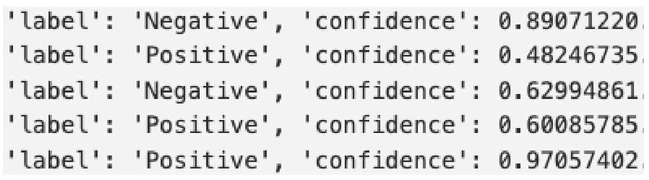
This gives more insight to the viewer as to the reason for the positive or negative reviews. For example for the above review, the negative comments given might be due to the actor and acting being disappointing. Furthermore, this model can be used for confirming sentiment analysis in section 4.5. The model could predict the sentiment class of the review and give confidence in the classification.



```python
def predict_absa(text: str) -> dict:
    """
    Tokenizes the input text, runs the ABSA model,
    and returns a dictionary with the predicted sentiment label and the confidence score.
    """
    # Tokenize the text with truncation and padding.
    inputs = tokenizer(text, return_tensors="pt", truncation=True, padding=True, max_length=512)

    with torch.no_grad():
        outputs = model(**inputs)

    # Extract logits and compute softmax probabilities.
    logits = outputs.logits[0]
    probs = torch.softmax(logits, dim=0)

    # Get the predicted class index and its confidence.
    predicted_class = torch.argmax(probs).item()
    confidence = probs[predicted_class].item()

    # Use the model's id2label mapping if available.
    label = model.config.id2label.get(predicted_class, str(predicted_class)) if hasattr(model.config, "id2label") else str(predicted_class)

    return {"label": label, "confidence": confidence}
```

Figure 36: ABSA 2 model

However, the output does not give the aspect of the class but only the confidence level.

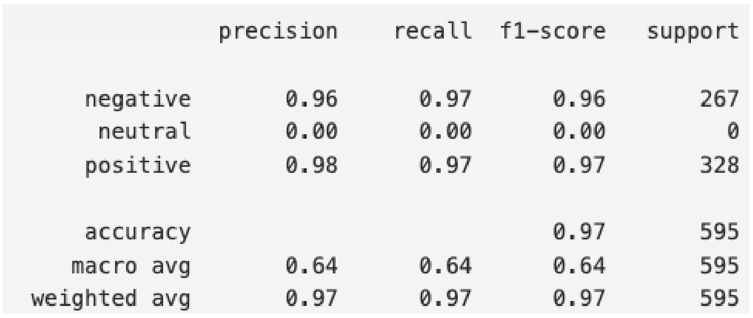


```
'label': 'Negative', 'confidence': 0.89071220
'label': 'Positive', 'confidence': 0.48246735
'label': 'Negative', 'confidence': 0.62994861
'label': 'Positive', 'confidence': 0.60085785
'label': 'Positive', 'confidence': 0.97057402
```

Figure 37: Example output for ABSA 2

Given the information on the confidence, we will now extract the label only when the confidence is more than 80%. The reviews which are not marked as positive or negative will be stated as 'EMPTY' In this case, we have identified the labels that are stated we are not confident in.

Given the new sentiment, we now perform the accuracy test again but a bit different. Previously we have performed accuracy testing on all 1000 labeled data. Now, due to the presence of an additional group 'EMPTY', we will ignore these reviews and only perform accuracy tests on the rest to ensure accuracy.



```
              precision    recall  f1-score   support

    negative       0.96      0.97      0.96       267
     neutral       0.00      0.00      0.00         0
    positive       0.98      0.97      0.97       328

    accuracy                           0.97       595
   macro avg       0.64      0.64      0.64       595
weighted avg       0.97      0.97      0.97       595
```

Figure 38: Classification report for ABSA 2

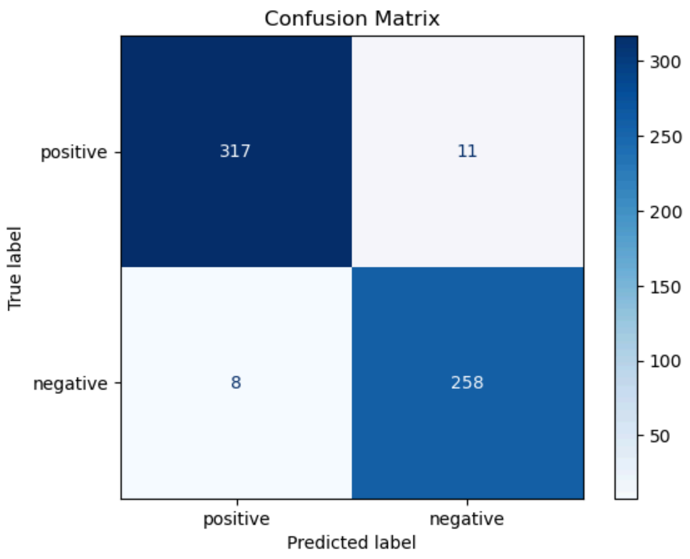


Figure 39: Confusion Matrix for ABSA 2

It can be seen that the F1 score for both positive and negative increase dramatically to 0.97 and 0.96. There are also fewer false positives and false negatives as seen in the confusion matrix. However, we could not directly compare with the accuracy test result from above as the number of data used is 595, around half of what we had used earlier. Regardless of the comparison, we realised the importance of identifying the sentiment labels we are not confident in as many of the times reviews are neutral with no strong opinion or emotion.

4.6.3 File Location

The files relevant to this section are organized into two main components within the designated folder: a **Code** Folder and a **Data Output** Folder.

Code folder	1. **ABSA.py**: Code used to generate output for the second method of ABSA result. 2. **sarcasm2.py**: Code used to generate output for the second method of sarcasm result. 3. **Q5.ipynb**: Code used to generate the first method of ABSA and sarcasm. Accuracy test code and result are included for all methods.
Output folder	1. **Finals.json** ● Used for powering the website interface ● Contains sentiment results from: ○ Basic sentiment analysis ○ Second method of sarcasm detection ○ Flipped sentiment for second method of sarcasm ○ Second ABSA model results 2. **manualLabel_Final.json**: Include all 1000 manually labeled data with ● Second method of sarcasm ● Flip sentiment result for second method of sarcasm ● Second method of ABSA 3. **AllSarcasmAllABSAresults.csv**: Includes all 1000 manually labeled dataset ● First and second method of sarcasm ● Flip sentiment result for first and second method of sarcasm ● First and second method of ABSA ● Second method of ABSA result used for accuracy testing

5. Submission Links

3. Youtube Link

Upon completing the SC4021 Information Retrieval Course Assignment, our team produced a video presentation to showcase the practical applications, impact, and innovative aspects of our project. You can view the presentation at the following YouTube link:
https://youtu.be/FBzbrYpglho

4. A Dropbox (or similar, e.g., Google Drive or OneDrive) link to a compressed (e.g., zip) file with crawled text data, queries and their results, evaluation dataset, automatic classification results, and any other data for Questions 3 and 5

https://entuedu-my.sharepoint.com/:f:/g/personal/kxing003_e_ntu_edu_sg/EgNs2ikiUG5HuNh9QEg5tYoBY90-M4zBISu_t7RSE51n3A?e=4vK3Pc

The files in side the zipped folder are as follows:

1. **all_review_information.json**
   This file contains all the crawled text data from IMDB for our group's Information Retrieval System, which is used as an input for our Classification model.
2. **analysis_all_movies (in CSV and JSON)**
   These are the prediction outputs from our model in Final Sentiment Analysis, where movie reviews are labelled with sentiments and emotions.
3. **manual labelled data.csv**
   This file contains 1000 manually annotated text data to be used for the performance evaluation for classification model.
4. **query_results.pdf** (queries and their results)
5. **final.json**
   These are the prediction outputs after part of the enhance classification method (second method of ABSA and second method of Sarcasm Detection), where movie reviews are further labelled with the aspects and sarcasm output. It contains the whole dataset.
6. **AllSarcasmAllABSAresults.csv**
   These are the prediction outputs after all enhance classification method (ABSA and Sarcasm Detection) where movie reviews are further labelled with the aspects and

sarcasm output for only the 1000 manually labelled reviews, which will be used for evaluation purpose only.

5. A Dropbox (or similar, e.g., Google Drive or OneDrive) link to a compressed (e.g., zip) file with all your source codes and libraries, with a README file that explains how to compile and run the source codes

The **OneDrive link** to the compressed file with all source codes and libraries: https://entuedu-my.sharepoint.com/:f:/g/personal/jtan573_e_ntu_edu_sg/EsuYlYpkPnRHsWmUdbdK_AcB03xOFxxr4VPqx4_MJ7Vpmw?e=JXByeg

For the readme file with images for the frontend source codes, it can be found in the **Github link**: https://github.com/jtan573/SC4021-Information-Retrieval-Assignment/tree/main#

6. References

For Solr Package:

https://www.npmjs.com/package/solr-client

Hansonrobotics. (n.d.). *hr-solr/synonyms_en.txt at master*

hansonrobotics/hr-solr. GitHub.

https://github.com/hansonrobotics/hr-solr/blob/master/synonyms

A list of English stopwords. (n.d.). Gist.

https://gist.github.com/rg089/35e00abf8941d72d419224cf

5b5925d