

**NANYANG
TECHNOLOGICAL
UNIVERSITY**
SINGAPORE

**CZ4034 Information Retrieval
Course Assignment Report**

Group 32

No	Name	Matriculation Number
1	Chen Wei May	U2020687E
2	Aye Myint Maung	U1920344G
3	Nyi Ye Htut	U1920263D

1. Introduction	3
1.1 Topic Selection	3
1.2 Problem Statement	3
1.3 Aim	3
2. Web Crawling of Text Data	4
2.1 Methodology for Crawling of Data	4
2.2 Analysis of Crawled Data in Text Corpus	4
3. Preprocessing Steps:	5
3.1 Sample Queries	5
4. Indexing and Querying of Text Corpus	7
4.1 Inverted Index	7
4.2 Document Frequency	7
4.3 TF-IDF	7
4.4 Cosine Similarity Score	8
5. Evaluation of Retrieval Results	9
6. UI	11
7. Classification for Sentiment Analysis	15
7.1 Classification Problem:	15
7.2 Approaches to Classification Problem	15
7.3 Classification Approach	16
7.4 Training Data	16
7.5 Justification for No Polarity Detection	17
7.6 Preprocessing of Reviews for Classification	17
7.6.1 Preprocessing Functions	17
7.6.2 Processed and Cleaned Reviews	18
7.7 Building Classification Models	18
7.8 Ensemble Classifiers	19
7.9 Discussion of Classifiers Actual Performances on Crawled Data	20
7.10 Evaluation of Classification Results	24
8. Submission	25
8.1 Video Presentation Link	25
8.2 Database Link	25
8.3 UI+source codes Link	25

1. Introduction

There are a plethora of movies already available, and the movie industry is ever-growing. As a result, consumers are spoilt for choice whenever they want to explore movie options. The existing online databases of information related to movies offers typical information such as cast, directors, movie title, rating, genre, etc. Movie reviews contain much useful information about the movie. The main value of movie reviews are the descriptive information and the sentiment information. The conventional ways of filtering for a selection of movies (such as by overall movie rating, alphabetical order) has been widely accepted for a long time.

1.1 Topic Selection

User descriptions of a movie are considerably different from the language used in official movie descriptions on movie listings. The official movie descriptions are often written by movie experts or the creators of the movies for the purpose of promoting the movies.

However, it is known that movie reviews contain much useful descriptive and sentiment information expressed by the average consumer. Furthermore, the descriptions in user movie reviews may often be more natural and colloquial and can sometimes be longer than the general movie description. Additionally, the current numeric ‘star’ ratings of movie reviews are relatively uninformative.

Hence, our team wanted to explore the possibility of creating a different, innovative and better way to search for and sort movies according to how a user would describe it in layman terms, rather than the conventional sorting methods of numeric ‘star’ rating of a movie and alphabetical order.

Thus, it is worthwhile to explore a written user review-based movie search engine for users, based on the movie descriptions written by other general consumers (as opposed to movie descriptions written by movie-makers). This is further illustrated in the classification section where sentiment analysis is utilised to create a more informative rating metric from the user written reviews.

1.2 Aim

We aim to create a ‘user-friendly’ movie search engine that fits the average user’s natural language, so that users can obtain a useful and relevant list of movies that match most closely with the user’s desire as specified by their natural language query. We also hope to create a more informative rating metric from the user written reviews because we believe that users will find this useful.

2 Web Crawling of Text Data

2.1 Methodology for Crawling of Data

BeautifulSoup python library was used for crawling of data from IMDb website listing the top 1000 movies of all time, sorted by user rating.

(https://www.imdb.com/search/title/?groups=top_1000&sort=user_rating.desc&count=100&start=000&ref_=adv_next)

2.2 Analysis of Crawled Data in Text Corpus

First, the data fields of title, genre, duration, rating, year, certification, url (to each movie), director and cast were scraped for 1000 movies. Later, using the url data of each movie, the first 25 movie reviews were scraped for each movie, resulting in a total of 25000 movie review records. (Each movie review had a minimum of 10 words.)

3 Preprocessing Steps:

The same preprocessing steps were performed (in the specified order) on the **review text data** and the 5 sample **user queries**.

Note: numbers were not removed due to our judgement about the movie query context. Because some movies contained numbers in the titles, hence the numbers could contribute significantly to the relevance of a text to a query.

Step	Preprocessing Description
1	Convert to lowercase : convert all text to lowercase form
2	Strip html tags : remove all html tags from the text
3	Expand contractions : expand contracted words (e.g. “don’t” → “do not”)
4	Remove punctuation : remove all punctuation from the text
5	Spell checking : check spelling of words in the reviews and correct misspelled words
6	Tokenization : tokenize the text
7	Remove stopwords : remove stopwords from the text (assumed in this context that stopwords are unimportant for user querying)
8	Stemming : stemming on the tokenized text.

3.1 Sample Queries

	Query
1	"Movie about aliens that is an absolute sci-fi thriller"
2	"Moviee abt a fast car with amaze police actin"
3	"Family-friedly animtion with strong emphasis 32 on values and commnity, that teach lessons."
4	"Drama mafia guns not for the feint hearted"
5	"best drama of all time, so romantic and lovely, hopeful girls and heartwrenching."

Spell checking for isolated spell correction was done using pyspellchecker python library. Spell checking in this manner was done for the review texts and queries respectively.

Spell Checking

Isolated spell checking was carried out using the spellchecker from pyspellchecker python library,

```
Mispelled Query 1 :      Movie about alieens that is an absolut sci-fi thriller
Spell-Corrected Query 1 : movie about aliens that is an absolut scifi thriller

Mispelled Query 2 :      Moviee abt a fast car with amaze police actin
Spell-Corrected Query 2 : movie abt a fast car with amazed police actin

Mispelled Query 3 :      Family-friedly animtion with strong emphasis 32 on vaulues and commuunityy, that teache lesssons.
Spell-Corrected Query 3 : familyfriedly animation with strong emphasis 32 on values and community that teach lessons

Mispelled Query 4 :      Dramaa mufia guns not for the feint hearted
Spell-Corrected Query 4 : drama mafia guns not for the feint hearted

Mispelled Query 5 :      bests drama of all time, so romanticc and lovly, hopefull girlss and heartwrenching.
Spell-Corrected Query 5 : bests drama of all time so romantic and lovely hopefully girls and heartwrenching
```

Details on stemming for the review text is shown below:

Before Stemming	After Stemming
<pre>movie_reviews_df['term_dict'].head() 0 [serious, upvoting, film, imdb, weighted, aver... 1 [soorarai, pottru, 2020, nbrief, review, major... 2 [shawshank, redemption, without, doubt, one, b... 3 [today, bothered, review, godfather, everyone,... 4 [first, need, point, generally, hate, action, ... Name: term_dict, dtype: object</pre>	<pre>movie_reviews_df['term_dict_stemmed'].head() 0 [seriou, upvot, film, imdb, weight, averag, fi... 1 [soorarai, pottru, 2020, nbrief, review, major... 2 [shawshank, redempt, without, doubt, one, best... 3 [today, bother, review, godfath, everyon, pret... 4 [first, need, point, gener, hate, action, film... Name: term_dict_stemmed, dtype: object</pre>

There are 4106036 words in the corpus and 126388 unique words in the corpus. The number of words were counted from the stemmed term dictionary collection.

```
# Total number of terms in corpus:
corpus_total = 0
for i in range(len(movie_reviews_df)):
    corpus_total += len(movie_reviews_df['term_dict_stemmed'][i])
corpus_total
```

4106036

```
# Total number of unique terms in corpus
# corpus_total_unique = 0
corpus = []
for i in range(len(movie_reviews_df)):
    # to append each stemmed collection from all rows
    corpus.extend(movie_reviews_df['term_dict_stemmed'][i])
corpus_total_unique = len(set(corpus))
corpus_total_unique
```

126388

4 Indexing and Querying of Text Corpus

The ranked retrieval was done based on doc(i.e. movie) similarity to the query input. The similarity metric used was cosine similarity.

4.1 Inverted Index

First, an inverted index was created for each term. An inverted index was created as a dictionary of terms with a postings list of the indexes of all the documents (movies) that each term appeared in.

Snippet of terms and their postings lists:

seriou	{0, 2, 3, 4, 5, 6, 8, 9, 10, 11, 12, 14, 16, 1...
upvot	{0}
film	{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13,...
imdb	{0, 2, 3, 4, 5, 6, 7, 8, 10, 11, 12, 15, 16, 1...
weight	{0, 517, 519, 7, 12, 13, 14, 525, 528, 24, 27,...
	...

4.2 Document Frequency

Document frequency is the number of documents in which a term appears

Then the document frequency was obtained as the length of the posting list for each term.

```
{'seriou': 589,  
  'upvot': 1,  
  'film': 998,  
  'imdb': 507,  
  'weight': 218,  
  'averag': 487,  
  'rate': 784,  
  ...}
```

4.3 TF-IDF

Counter from the python collections library was used to iterate over all documents to obtain the frequency of the tokens (term frequency), and calculate idf. In this manner, a dictionary was created with (document, token) pair as a key, with tf_idf as the value.

The TF-IDF weights were calculated similarly for documents and user query (ddd.qqq = n't'c.n't'c). The TF-IDF weight calculated is a variation of the standard ntc variant, where 't' in

'ntc' is $\log \frac{N+1}{DF+1}$ instead of $\log \frac{N}{DF}$.

	Formulae
Term Frequency (TF)	$TF(t, d) = \frac{\text{count of the number of terms } t \text{ in document } d}{\text{number of words in document } d}$
Document Frequency (DF)	DF(t) = occurrence of t in N, where N is number of documents
Inverse Document Frequency (IDF)	$IDF(t, d) = \log \frac{N+1}{DF+1}$
Term Frequency-Inverse Document Frequency (TF-IDF)	$TF\text{-}IDF(t, d) = TF * IDF$ The TF-IDF was weighted using α , where $\alpha = 0.3$ Hence, $TF\text{-}IDF(\text{weighted}) = TF\text{-}IDF(t, d) * 0.3$

4.4 Cosine Similarity Score

The tf-idf weights of document vectors were in a document vector matrix, which is a numpy matrix of dimensions (number of docs (ie. movies), total_vocab) where total_vocab is the total number of unique words in the text corpus.

The tf-idf weight for a query is also calculated in a similar way as earlier described.

A cosine similarity score was calculated via:

Dot product

Unit vectors (normalized)

$$\cos(\vec{q}, \vec{d}) = \frac{\vec{q} \bullet \vec{d}}{|\vec{q}| |\vec{d}|} = \frac{\vec{q}}{|\vec{q}|} \bullet \frac{\vec{d}}{|\vec{d}|} = \frac{\sum_{i=1}^{|V|} q_i d_i}{\sqrt{\sum_{i=1}^{|V|} q_i^2} \sqrt{\sum_{i=1}^{|V|} d_i^2}}$$

where q_i is the tf-idf weight of term i in the query, d_i is the tf-idf weight of term i in the document.

Each cosine similarity score was calculated for the queries and the movies, then the movies returned to the user were ranked in decreasing order of cosine similarity score with respect to the user query. (ie. the first-ranked movie is the one with the highest score similarity score)

5.Evaluation of Retrieval Results

The time taken for each query to be processed was obtained via the python timeit library, using the % timeit function.

Hand-labelled relevance values for 100 movies were done for each of the 5 sample queries for the purpose of evaluating our movie search engine information retrieval system. (Note only 100 movies were labelled due to time and resource constraints.)

For each of the queries, an evaluation of the movie search engine information retrieval system was conducted via the plotting of a graph showing how precision, recall, f1 score and arithmetic mean changes with the number of documents. Note: it is unsurprising that the precision is generally the highest, and recall is always the lowest, with arithmetic mean and f1 score in between.

Precision =	$\frac{\text{Total number of documents retrieved that are relevant}}{\text{Total number of documents that are retrieved}}$
Recall =	$\frac{\text{Total number of documents retrieved that are relevant}}{\text{Total number of relevant documents in the database}}$

From the graphs below, it is evident that our movie information retrieval system works well. This is because precision remains high for at least the first 10 movies retrieved. This means that the user is likely to find our movie search engine useful because it results in a high number of relevant documents being retrieved at the start.

Instances where less relevant documents are retrieved often occur only after 10 or sometimes even after 20 movies are retrieved, and it would be reasonable to believe that the user would have found something relevant to his search query within around the first 15 results that are displayed.

	Sample Query	Movies Retrieved	Graph of Results (Recall, Precision, F1-measure, Arithmetic mean)
1	"Movie about aliens that is an absolute sci-fi thriller"	Alien Aliens Ratsasan Star_Wars The_Matrix Drishyam_2 Interstellar Se7en The_Empire_Strikes_Back Back_to_the_Future	<p>27.6 ms \pm 10.9 ms per loop (mean \pm std. dev. of 7 runs, 10 loops each)</p>
2	"Moviee abt a fast car with amaze police actin"	Jai_Bhim Tengoku_to_jigoku Ratsasan The_Departed The_Usual_Suspects The_Lives_of_Others Vikram_Vedha Fight_Club Drishyam_2 Sen_to_Chihiro_no_kamikakushi	<p>24.3 ms \pm 1.24 ms per loop (mean \pm std. dev. of 7 runs, 10 loops each)</p>

3	"Family-friendly animation with strong emphasis on values and community, that teach lessons."	Chhichhore 3 Idiots Taare Zameen Par Léon Dangal Seppuku K.G.F: Chapter 1 Whiplash Idi i smotri 96	
21 ms \pm 1.52 ms per loop (mean \pm std. dev. of 7 runs, 10 loops each)			
4	"Drama mafia guns not for the feint hearted"	The Godfather Goodfellas The Godfather: Part II The Departed Braveheart The Dark Knight Once Upon a Time in America Joker Nuovo Cinema Paradiso Pulp Fiction	
21.3 ms \pm 2.25 ms per loop (mean \pm std. dev. of 7 runs, 100 loops each)			
5	"best drama of all time, so romantic and lovely, hopeful girls and heartwrenching."	Ayla: The Daughter of War Hotaru no haka 96 Soorarai Pottru Coco 3 Idiots Idi i smotri La vita è bella The Lord of the Rings: The R Avengers: Endgame	
23.8 ms \pm 3.71 ms per loop (mean \pm std. dev. of 7 runs, 100 loops each)			

6. UI

For the UI, we are using python3 Streamlit package.

Streamlit library supports numpy and panda which are used to get data retrieval.

Notes on running the application on a local machine:

Pip install python streamlit library via 'pip install streamlit'

Then navigate to the folder containing cosine_find.py

Run the command 'streamlit run cosine_find.py'

Open the link to a local host in a browser to start using the movie information search engine application.

Fig: UI user input query page (Default setting)

Movie Search Engine App

Please describe the movie and list the director/cast

Submit

Filter option?

By Query Similarity (Cosine)

By Query Similarity (Cosine)

By IMDB Rating (Desc)

By Written Review Rating (Desc)

Fig: UI using cosine similarity search

Movie Search Engine App


Please describe the movie and list the director/cast

Movie about aliens that is an absolute sci-fi thriller

Submit

Filter option?

By Query Similarity (Cosine)

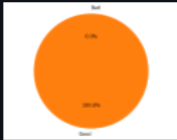


Alien


<https://www.imdb.com/title/tt0078748/>

Imdb rating: 8.5

Written review rating: 10.0



written rating pie chart

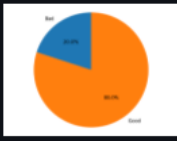


Aliens


<https://www.imdb.com/title/tt0090605/>

Imdb rating: 8.4

Written review rating: 8.0



written rating pie chart

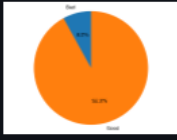


Arrival


<https://www.imdb.com/title/tt2543164/>

Imdb rating: 7.9

Written review rating: 9.2



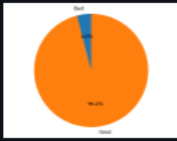
written rating pie chart



District 9

<https://www.imdb.com/title/tt1136608/>

Imdb rating: 7.9



written rating pie chart

Fig: UI sorted by IMDb rating

Movie Search Engine App


Please describe the movie and list the director/cast

Movie about aliens that is an absolute sci-fi thriller

Submit

Filter option?

By IMDB Rating (Desc)

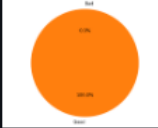


Alien


<https://www.imdb.com/title/tt0078748/>

Imdb rating: 8.5

Written review rating: 10.0



written rating pie chart

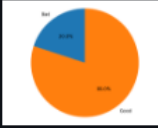


Aliens


<https://www.imdb.com/title/tt0090605/>

Imdb rating: 8.4

Written review rating: 8.0



written rating pie chart

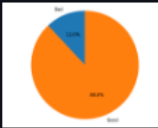


The Thing


<https://www.imdb.com/title/tt0084787/>

Imdb rating: 8.2

Written review rating: 8.8



written rating pie chart

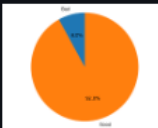


Arrival

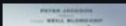
<https://www.imdb.com/title/tt2543164/>

Imdb rating: 7.9


Written review rating: 9.2



written rating pie chart



District 9



written rating pie chart

Fig: UI sorted by Written review rating

Movie Search Engine App


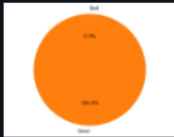

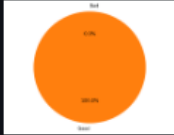

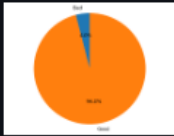

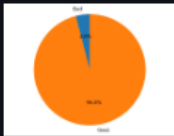
Please describe the movie and list the director/cast

Movie about aliens that is an absolute sci-fi thriller

Submit

Filter option?

By Written Review Rating (Desc) ▾

	<p>Alien</p> <p>https://www.imdb.com/title/tt0078748/</p> <p>Imdb rating: 8.5</p> <p>Written review rating: 10.0</p>	 <p>written rating pie chart</p>
	<p>The Day the Earth Stood Still</p> <p>https://www.imdb.com/title/tt0043456/</p> <p>Imdb rating: 7.8</p> <p>Written review rating: 10.0</p>	 <p>written rating pie chart</p>
	<p>District 9</p> <p>https://www.imdb.com/title/tt1136608/</p> <p>Imdb rating: 7.9</p> <p>Written review rating: 9.6</p>	 <p>written rating pie chart</p>
	<p>Edge of Tomorrow</p> <p>https://www.imdb.com/title/tt1631867/</p> <p>Imdb rating: 7.9</p> <p>Written review rating: 9.6</p>	 <p>written rating pie chart</p>

7. Classification for Sentiment Analysis

7.1 Classification Problem:

The conventional ranking of movies is by the average of all numeric user ratings, often in the form of 'star ratings' selected by users without justification for their choice of rating.'

The currently available numeric user rating filters commonly used on movie platforms are not sufficiently justified. Different people have different perceptions of what determines a 'good' or 'bad' and they translate these perceptions into their own (distinct) sense of an appropriate score on a numeric scale (which they may determine differently).

For example, a consumer (critical rater) may give the movie he deems to be 'excellent' a numeric 'star' rating of '7' and give a movie he deems to be 'bad' a numeric 'star' rating of '3' respectively. However, a different (lenient rater) consumer who deems the same movie to be 'excellent' may give the movie a numeric rating of '9' or give the movie a numeric rating of '6' if he deems it to be 'bad'. Hence it is difficult to determine whether a movie with an average rating of 6.5 is an 'excellent' movie rated mostly by critical raters or whether it is a 'bad' movie rated mostly by lenient raters.

What is more informative instead, is the written reviews which can more aptly capture the general sentiment of consumers about a movie. If these sentiments can be mapped into a numeric form via the same procedure, it would result in 'fairer' numeric ratings that are reflective of the true audience sentiment about the movies, across all movies. Hence, the aim of our classification problem is to map the general sentiment of reviews of a movie to a numeric value that can be compared.

It is difficult to determine the 'fairness' of ratings due to different user perceptions of what constitutes a 'good' or 'bad' movie on a numeric scale.

7.2 Approaches to Classification Problem

Our team explores how relevant movie retrieval can be enhanced by offering a different assessment of 'user rating', which will henceforth be referred to as 'written reviews rating'. This 'written reviews rating' will be a numeric value generated from the analysis of movie review sentiments. In providing this new 'written reviews rating' metric, our team aims to offer a better movie information retrieval search engine offering that is more sensitive to user's needs as specified by their query.

Example for raw review data labelled with positive '1' and negative: '0' sentiment is given below.

7.3 Classification Approach

As mentioned above, data was crawled from IMDb website listing the top 1000 movies of all time, sorted by user rating.

(https://www.imdb.com/search/title/?groups=top_1000&sort=user_rating_desc&count=100&start=000&ref=adv_next)

Later, using the url data of each movie, the first 25 movie reviews were scraped for each movie, resulting in a total of 25000 movie review records. (Each movie review had a minimum of 10 words.)

Movie reviews for 10% of the movies were labelled. (Each movie has 25 reviews, 100 movies = $25 * 100 = 2500$ movie reviews were manually labelled.) Manual labelling of sentiment (1 for 'Positive', 0 for 'Negative') was done for a total of 2500 movie reviews.

The classification models created were trained on the training data from were evaluated based on the performance of the class predictions of the results based on these models.

First, Two Classifier models were used: Logistic Regression and Linear Support Vector Classifier(SVC).

Later, Two StackingClassifier models were used in an attempt to obtain enhanced classification results.

We will choose one classifier out of these two based on the accuracy score and use it to classify the remaining crawled data of $(25000 - 2500 =) 22500$ reviews.

7.4 Training Data

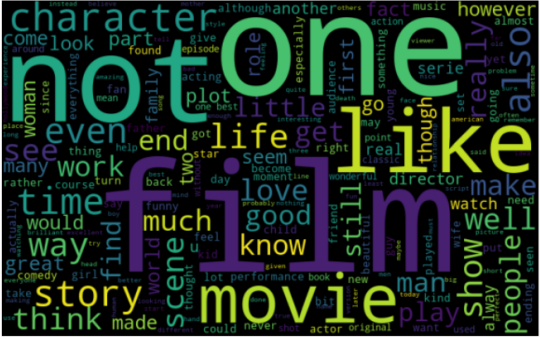

To train our classifier, we got training data from Kaggle Dataset which includes a total of 50k movie reviews with sentiment analysis included.

(<https://www.kaggle.com/datasets/lakshmi25npathi/imdb-dataset-of-50k-movie-reviews>)

This is to save time for manually labelling a large portion of the crawled data set. Training the classifier models on this large labelled dataset will make the models more robust.

	review	sentiment
0	One of the other reviewers has mentioned that ...	1
1	A wonderful little production. The...	1
2	I thought this was a wonderful way to spend ti...	1
3	Basically there's a family where a little boy ...	0
4	Petter Mattei's "Love in the Time of Money" is...	1

Fig1: Snippet of 50k Movie reviews dataset from Kaggle

Word cloud of the reviews labelled 'positive' in the 50k movie reviews.	Word cloud of the reviews labelled 'negative' in the 50k movie reviews.
	

7.5 Justification for No Polarity Detection

The model will be trained on movie reviews. As it is a reasonable assumption that reviewers of movies are either 1. emotionally invested enough in the movie to be motivated to give a review, or 2. they have been requested by someone to give their 'non-neutral' opinion/review on the movie. In either case, their review is reasonably believed to be opinionated or emotionally-charged. Hence, it was judged that polarity detection should not be carried out for this particular classification problem.

7.6 Preprocessing of Reviews for Classification

7.6.1 Preprocessing Functions

1. Punctuation Removal
2. Lowering the Text
3. Spell Checking
4. Tokenization
5. Remove Stopwords
6. Lemmatization

7.6.2 Processed and Cleaned Reviews

The results of the preprocessing steps.

	review	sentiment	preprocessed_review
0	One of the other reviewers has mentioned that ...	1	one reviewer mentioned watching 1 oz episode h...
1	A wonderful little production. The...	1	wonderful little production filming technique ...
2	I thought this was a wonderful way to spend ti...	1	thought wonderful way spend time hot summer we...
3	Basically there's a family where a little boy ...	0	basically family little boy jake think zombie ...
4	Petter Mattei's "Love in the Time of Money" is...	1	petter mattei love time money visually stunnin...

7.7 Building Classification Models

The 50k reviews 80% train set and 20% test set. Both the training and testing dataset will include a train and test set with equal number of randomised positive and negative reviews. We use the same method as in Logistic Regression and used the 50k data set for training and testing to train the classifiers.

Linear Support Vector Classifier (SVC)					
The classification report obtained using scikit-learn python library is shown below.					
	precision	recall	f1-score	support	
0	0.88	0.87	0.87	20028	
1	0.87	0.88	0.88	19972	
accuracy			0.88	40000	
macro avg	0.88	0.88	0.88	40000	
weighted avg	0.88	0.88	0.88	40000	
Fig1: Classification report of our trained LinearSVC classifier.					

Logistic Regression

The classification report obtained using scikit-learn python library is shown below.

	precision	recall	f1-score	support
0	0.89	0.87	0.88	20028
1	0.87	0.89	0.88	19972
accuracy			0.88	40000
macro avg	0.88	0.88	0.88	40000
weighted avg	0.88	0.88	0.88	40000

Fig2: Classification report of our trained LogisticRegression classifier

7.8 Ensemble Classifiers

Stacking Classifier using LinearSVC and Logistic Regression

The classification report obtained using scikit-learn python library is shown below.

	precision	recall	f1-score	support
0	0.88	0.87	0.88	20028
1	0.88	0.88	0.88	19972
accuracy			0.88	40000
macro avg	0.88	0.88	0.88	40000
weighted avg	0.88	0.88	0.88	40000

Fig3: Classification report of our trained Stacking Classifier using LinearSVC and Logistic Regression

Stacking Classifier using SGDClassifier and RidgeClassifier	
The classification report obtained using scikit-learn python library is shown below.	

	precision	recall	f1-score	support
0	0.88	0.87	0.88	20028
1	0.88	0.88	0.88	19972
accuracy			0.88	40000
macro avg	0.88	0.88	0.88	40000
weighted avg	0.88	0.88	0.88	40000

Fig4: Classification report of our trained Stacking Classifier using SGDClassifier and RidgeClassifier

It was observed that all four classifiers produced very similar results, with precision scores around 0.8, recall scores around 0.88, f1-score around 0.88 for both negative reviews and positive reviews.

Thus, all four models are applied to the crawled data for evaluation of actual performance on unlabelled data.

7.9 Discussion of Classifiers Actual Performances on Crawled Data

To evaluate our classifier, we have a total of 2500 reviews which are manually analysed for sentiment. Using this dataset, we will run all the reviews with both of the classifiers and predict sentiment analysis on all the reviews. Then, the comparison will be made between the predicted sentiment values and the manually-labelled sentiment values.

Step 1: Load the manual data

```
df1 = pd.read_csv('manual_data_1.csv')
df1.head(5)
```

```
.....
```

	Review	Sentiment
0	This is one of the greatest films ever made.....	1
1	There are a few wonderful courtroom dramas out...	1
2	When I was younger I thought 12 Angry Men was ...	0
3	A gripping single-location drama and one of th...	1
4	12 Angry Men (1957) **** (out of 4) A poor slu...	1

Step 2: Preprocess the reviews (Include data cleaning, lowercase, tokenization, stop-words removal, lemmatization)

```
df1['preprocessed_review'] = df1['Review'].apply(lambda review: data_preprocessing(review))
df1.head()
```

	Review	Sentiment	preprocessed_review
0	This is one of the greatest films ever made.....	1	one greatest film ever made period much attrib...
1	There are a few wonderful courtroom dramas out...	1	wonderful courtroom drama anatomy murder kill ...
2	When I was younger I thought 12 Angry Men was ...	0	younger thought 12 angry men near perfect ense...
3	A gripping single-location drama and one of th...	1	gripping single location drama one best ever f...
4	12 Angry Men (1957) **** (out of 4) A poor slu...	1	12 angry men 1957 4 poor slum kid trial murder...

Step 3: Predict sentiment data respective Classifiers

Step 4: Classification report on predicted data compared with manually-labelled data on the (2500) movie reviews of the top 100 movies.

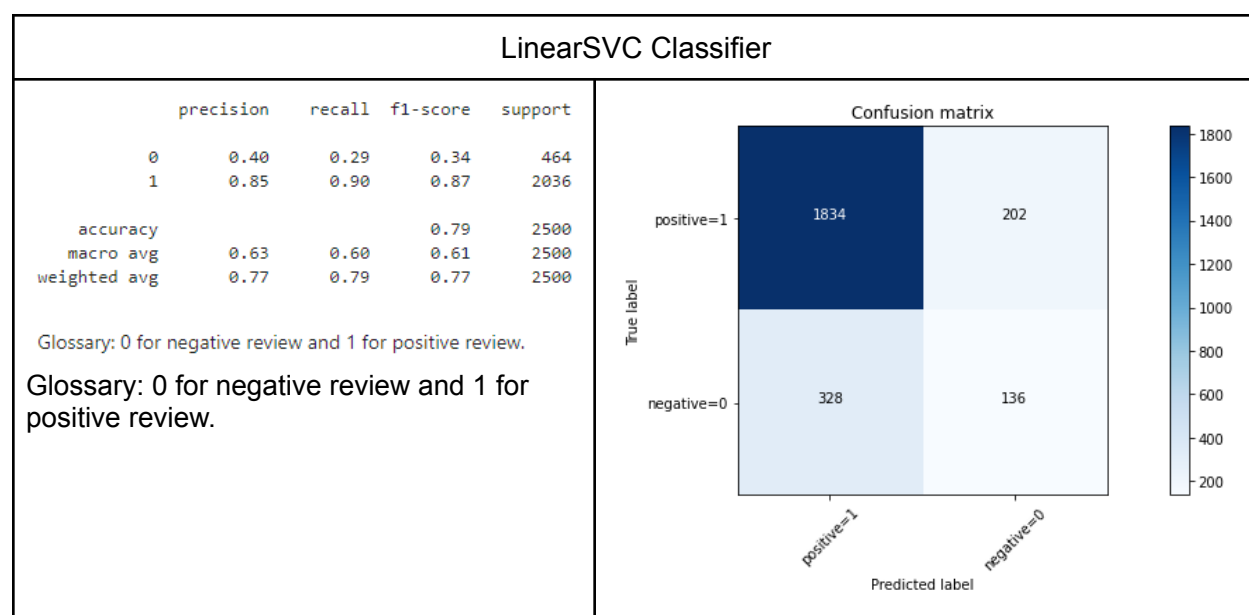


Fig 1. Classification report on how the predicted data compares with the manually labelled data when classified using the LinearSVC Classifier

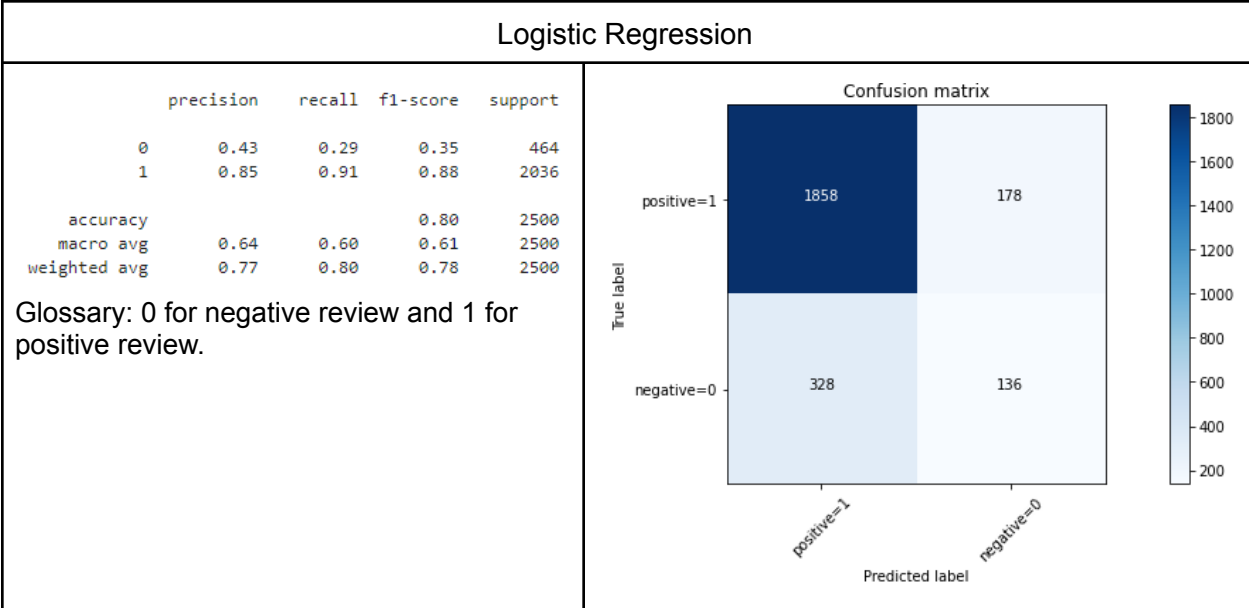


Fig 2. Classification report on how the predicted data compares with the manually labelled data when classified using Logistic Regression

Ensemble Classifiers

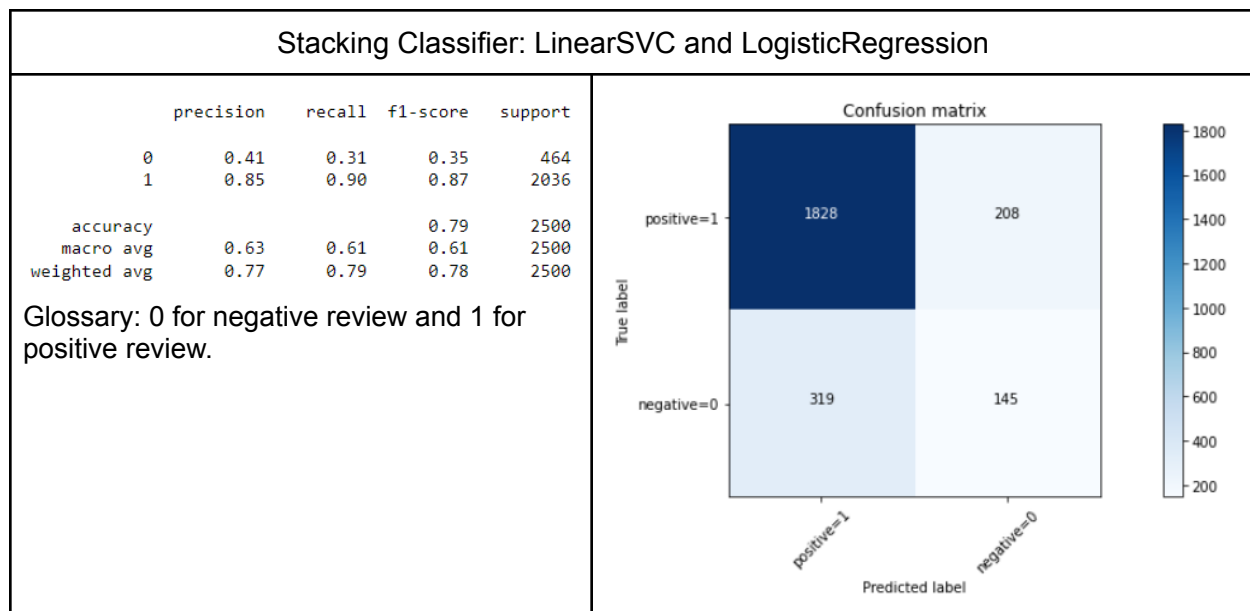


Fig 3. Classification report on how the predicted data compares with the manually labelled data when classified using Stacking Classifier using LinearSVC and LogisticRegression.

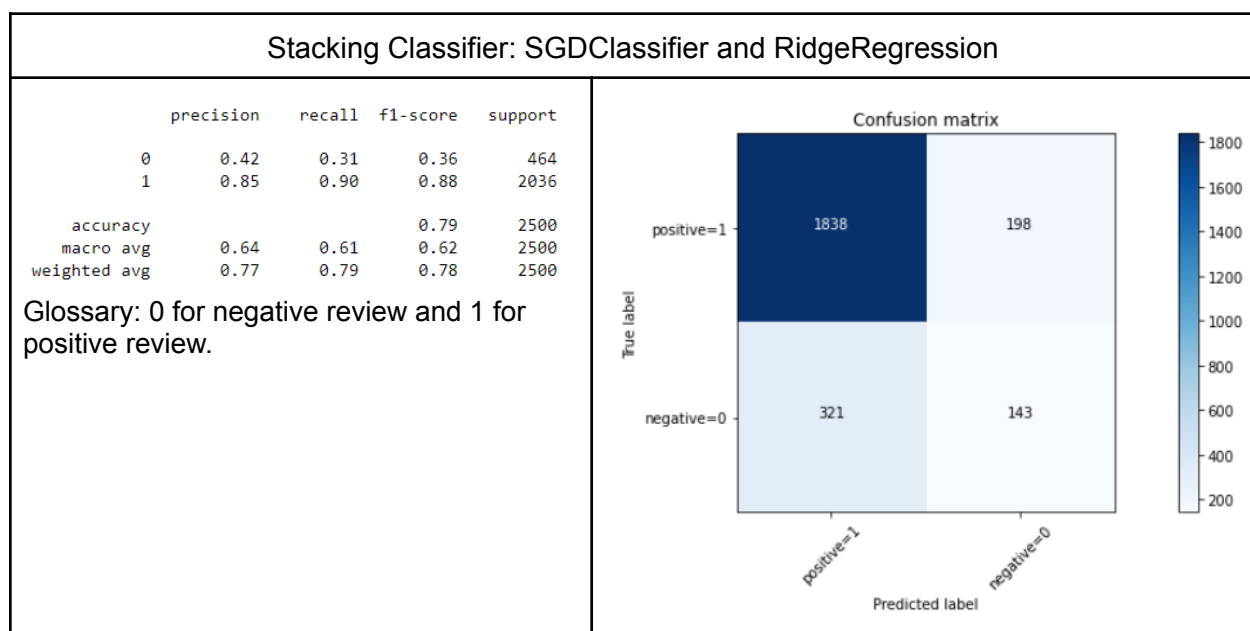


Fig 3. Classification report on how the predicted data compares with the manually labelled data when classified using Stacking Classifier using SGDClassifier and RidgeRegression.

No	Review	Sentiment	preprocessed_review	Predicted_Sentiment
0	1 This is one of the greatest films ever made.....	1	one greatest film ever made period much attrib...	1
1	2 There are a few wonderful courtroom dramas out...	1	wonderful courtroom drama anatomy murder kill ...	1
2	3 When I was younger I thought 12 Angry Men was ...	0	younger thought 12 angry men near perfect ense...	1
3	4 A gripping single-location drama and one of th...	1	gripping single location drama one best ever f...	1
4	5 12 Angry Men (1957) **** (out of 4) A poor slu...	1	12 angry men 1957 4 poor slum kid trial murder...	1
5	6 In a NYC courtroom, a first degree murder case...	1	nyc courtroom first degree murder case poor 18...	0
6	7 This movie came out of an era that I really lo...	1	movie came era really loved ensemble acting st...	1
7	8 Theater at its best is practically impossible ...	1	theater best practically impossible get film c...	1
8	9 A young ethnic kid from a rough area is up on ...	1	young ethnic kid rough area murder charge jury...	1
9	10 I could go on, but I thought I'd stop there be...	1	could go thought stop summary line get big cla...	1

7.10 Evaluation of Classification Results

According to the data above, Negative Reviews '0' have a very low score of precision, recall, f1-score and support due to less sampling data available in our crawled data for negative review compared to positive ones and each unmatched classified data has a huge impact on the score. Hence, the macro avg score has also a low score of around 0.65 because of the imbalance sampling data. However, the score that matters the most is the weighted avg which balances out the weight of all the sampling data and generates the actual classifier score which is 0.8.

It also appears that the Stacking Classifiers did not improve the recall and precision scores much.

Logistic regression model had an accuracy score of 0.80. The stacking classifiers had accuracy scores of 0.79 and 0.80 respectively.

Limitations and Other Considerations

Also, as the 1000 movies selected to be included in this search engine are the top 1000 movies most highly rated 'numerically', it might have been reasonable to hypothesise that most of the written reviews would also be positive in nature.

The first 25 movie reviews is a very small sample size and may not be a good indicator of the overall sentiment for the movies. It was an assumption we made for this project in consideration of the processing time and compute power accessible to us. Theoretically with unlimited compute power, we should process all movie reviews for each movie to calculate an accurate 'written review rating score'.

8.Submission Links

8.1 Video Presentation Link: <https://youtu.be/0v4kRQ-QSII>

8.2 Database Link(Crawl,classification,report.zip) :

<https://drive.google.com/file/d/164zmKlqeiSvtxcQqW5Yf8QEIcRA3C4zS/view?usp=sharing>)

8.3 UI + source codes Link (cz4034_movie_app_final.zip):

https://drive.google.com/file/d/1q620MLasVb5aHwiXUa2_JRId0BToMYnE/view?usp=sharing