

Movie Review Summarization Using Supervised Learning and Graph-Based Ranking Algorithm

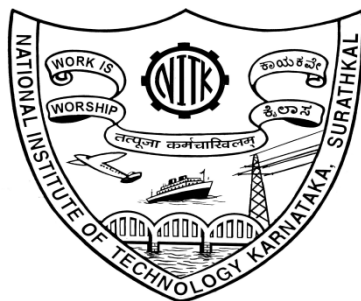
MASTER OF TECHNOLOGY in
COMPUTER SCIENCE AND ENGINEERING

by

Talapally Sandeep kumar-212CS033

Under the guidance of

Dr.Yogendra Jeppanna



Department of Computer Science
National Institute of Technology Karnataka, Surathkal.

24-02-2022

Contents

1	Topic	1
2	Motivation	2
3	Problem Definition	3
4	Literature Survey	4
4.1	Preprocessing:	5
4.2	Feature Extraction:	6
4.3	Summarization of Reviews:	6
5	Design and Implementation	7
5.1	Data Preprocessing	7
5.1.1	Removal Of Html Tags	7
5.1.2	Word Tokenization and Removal of Stop Words:	8
5.1.3	Performing Lancaster Stemming	9
5.2	Feature Extraction using TF-IDF	9
5.2.1	TF-IDF:	10
5.3	Multinomial NaiveBayes Classification	11
5.4	Graph Based Review Summarization	12
5.5	Novelty	14
6	Results	16
7	Conclusion	17
	References	18

List of Figures

5.1	Removal of html tags and Special characters	7
5.2	Example for code snippet to remove stop words	8
5.3	Example for Lancaster stemming code snippet	9
5.4	Unigrams and Bigrams	10
5.5	Code snippet for Unigrams and Bigrams	11
5.6	Conditional Probability	11
5.7	Naive Bayes	12
5.8	Undirected Weighted Graph	12
5.9	Importance of a node in final summary	14
5.10	Procedure of Performing Summarization	15
6.1	Accuracy Of Multi Naive Bayesian Model	16
6.2	Positive Review Summary of Dark Knight Movie	16
6.3	Negative Review Summary of Dark Knight Movie	16

Chapter 1

Topic

Summarizing movie reviews using Supervised Learning using Graph-Based Ranking Algorithm using IMDB dataset

Chapter 2

Motivation

Text summary is the process of reducing lengthy documents to concise paragraphs or phrases. The technique extracts important information while also ensuring that the paragraph's meaning remains consistent. This shortens the time required to comprehend huge documents such as research articles while without omitting critical information. The quantity of information accessible on the internet is unlimited. The primary goal of text summarising is to extract the most precise and helpful information from a huge document while eliminating unnecessary or less important material. Humans are often skilled at distinguishing between what is significant and what is not. As a result, they are adept at summarising lengthy materials. Machines, on the other hand, have no sense of what is significant or not. Text summarization may be done manually, which takes time. Instead, an automated model that can decrease human labour is the most desirable. With the evolution of the Web, which stresses user interaction, more and more websites, such as the Internet Movie Database (IMBD, a movie review website) and Amazon, allow people to publish reviews for things they are interested in. Online merchants frequently ask their customers to give opinions or reviews on the products or services they purchased online in order to satisfy customers and enhance their shopping experience, i.e. the number of reviews received by a product grows rapidly as millions of customers post reviews about a product, resulting in information overload. Because of this information overload, it is difficult for a buyer to read each product evaluation in order to decide whether or not to purchase a product. At the same time, it is difficult for Text summarization simplifies people's job, saves time, and increases work rate efficiency. It makes it easier to remember information. With the increasing availability of information on the internet, online movie reviews are becoming an important source of information for Internet users. Every day, hundreds of movie reviews are posted online, making it difficult to summarise them due to the physical labour required to read them. One of the most difficult problems in natural language processing is movie review mining and summary. As a result, an automated way to summarising long movie reviews is desired, as it will help consumers to immediately detect the good and bad parts of a movie.

Chapter 3

Problem Definition

Several individuals publish bulk reviews on movie review on IMDB on daily basis, which contain user attitude towards a certain movie. IMDB dataset comprising 50K movie reviews for natural language processing or Text analytics. This is a dataset for binary sentiment classification comprising much more data than prior benchmark datasets. This dataset contains a collection of 25,000 highly polar movie reviews for training and 25,000 for testing. This project is meant to summarise all the reviews of a certain movie.

Chapter 4

Literature Survey

Review mining and summarising is divided into two steps: review mining and review summarization. The process of obtaining, analysing, and categorising subjective information in order to determine the sentiment associated with a certain target is known as review mining or opinion mining. Many academics have offered various strategies for the job of review mining. For mining reviews in various areas, a variety of review mining methodologies have been developed, including ML-based and sentiment lexicon-based strategies. Opinion categorization of texts are also done using ML-based algorithms. There are two types of machine learning algorithms: supervised and unsupervised. These methods accomplish sentiment categorization by the extraction and selection of a collection of relevant characteristics. For sentiment categorization of movie review data, supervised machine learning techniques such as SVM are used. The authors classified high/low informative opinion phrases derived from restaurant reviews using decision trees.

Another solution to this issue is the sentiment lexicon-based strategy for review mining, which is divided into two categories: dictionary-based and corpus-based approaches. For polarity classification, researchers presented a dictionary-based technique combined with the WordNet network. This method used polarity ratings from thesaurus such as SentiWordNet with random walk analysis of ideas discovered in movie reviews to arrive at a final result. Because the same word might have distinct meanings in different domains, the dictionary-based method has the drawback of being unable to deal with context and domain-specific orientation. Researchers developed a corpus-based strategy that uses a corpus of manually annotated movie reviews. Parts-of-Speech (POS) tagging was used to extract linguistic elements such as nouns, verbs, adjectives, and adverbs from movie reviews. They also used SentiWordNet, a semantic resource, to compute the polarity score of movie review documents in the corpus. Both corpus and dictionary techniques depend largely on linguistic resources and are confined to terms in the lexicon.

Graph-based approaches for text summarization have received greater attention in recent years, and they have shown to be successful. The PageRank algorithm and its derivatives are used in these approaches to assign a rank/score to graph nodes, which represent phrases or paragraphs. The authors presented a connection graph in which nodes hold relevant information only if they are linked to a large number of other nodes. They proposed a Lex-PageRank strategy based on eigenvector centrality,

in which a sentence connectivity matrix is constructed and a PageRank-like algorithm is used to discover relevant phrases for summary. A PageRank-like system was also proposed, which detects important phrases for summary creation. They introduced a graph-based technique that combines surface data with text content and explores subtopic aspects across numerous texts to include them in the graph-based ranking algorithm. An affinity network-based multidocument summarization technique uses a similar algorithm to PageRank to generate sentence scores in the affinity graph based on information richness. Other authors developed a document-sensitive graph model for multidocument generic summarization, emphasising the importance of global document set information at the sentence level. A weighted graph model for generic multidocument summarization that incorporates sentence ranking and sentence grouping approaches is presented. To rank the essential sentences, they used the standard PageRank method. Some writers have developed a multidocument extractive summarising method based on an event graph. However, the technique requires the creation of handcrafted argument extraction procedures, which is a time-consuming operation that may restrict the approach's use to a certain domain.

4.1 Preprocessing:

Much of text summarization requires preprocessing. The preprocessing of data in computational linguistic is a significant procedure, particularly in review mining and summarization . As the suggested work is related to RMS, the review document needs to be preprocessed so that it can be used in experiment efficiently before giving it as an input to the system. The preprocessing phase involves four steps, sentence segmentation, tokenization, stop words removal, and word stemming.

(a) Sentence segmentation: It is an essential step in NLP applications such as IR, machine translation, semantic role labeling, and summarization. It is the process of boundary detection within a document which splits the document text into sentences. Mostly, full stop/period (.), sign of exclamation (!), or a sign of interrogation (?) is commonly used to signify boundary of a sentence .

(b) Tokenization: In this task, a simple program will be split into the sentences into distinct words by splitting them at whitespaces such as blanks, tabs, and punctuation marks such as period, semicolon, comma, and colon which are the primary cues for splitting the text into tokens.

(c) Stop words removal: words that appear frequently in the document are called stop words. It consists of conjunctions, articles, prepositions, and frequent words like “the,” “I,” “an,” and “a”. Stop words carry very little or no meaning in the document, so it is a good idea to remove them from document set. Eliminating stop words from review documents helps to increase the performance of the system.

(d) Word stemming: it is an important task in the preprocessing phase. Word stemming transforms the derived words to its stem or root word for capturing the similar concept. A well known stemming algorithm named as Porter's stemming is used for word stemming that removes the suffixes of words.

4.2 Feature Extraction:

This phase's goal is to use a well-known feature extraction approach called bag of words to extract features for review categorization (BoW). BoW is a straightforward feature extraction method that uses a vector space model to describe the review text content. The characteristics are made up of a mix of unigrams and bigrams.

Classification of Reviews: This phase's goal is to employ a supervised ML classification algorithm to categorise users' review text. Review categorization divides user feedback into positive and negative categories. When compared to other state-of-the-art classification algorithms, the Naive Bayes (NB) classification method is employed in this research since it is a robust classifier and achieves greater accuracy on scaled datasets. Because of its simplicity and accuracy, the NB classifier has a number of applications in text categorization.

4.3 Summarization of Reviews:

This phase's goal is to compile a summary of the reviews (both favourable and negative). There are three stages in this phase: (1) the building of a network from categorised reviews
(2) the ranking of graph nodes (review sentences)
(3) the selection of top rank sentences.

For text summary, all of these procedures have been suggested in recent years.

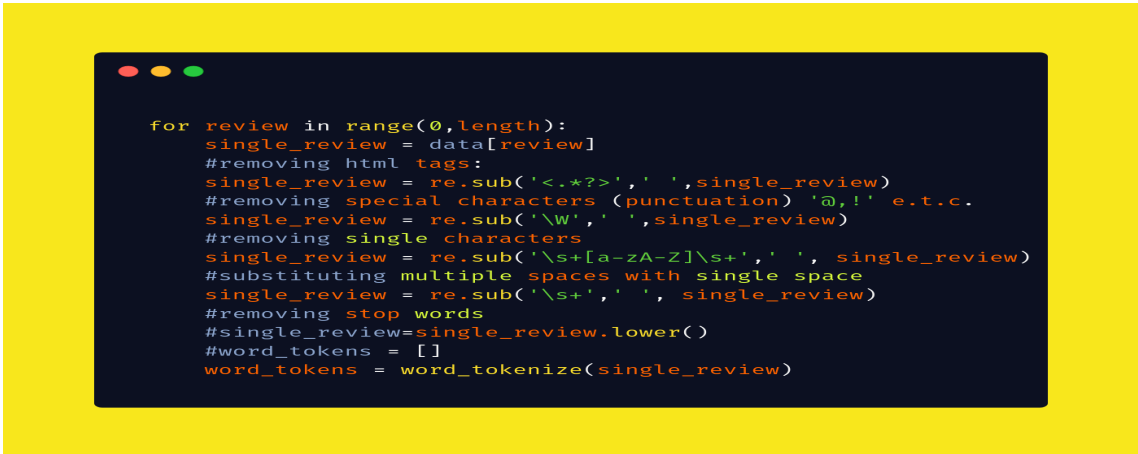
Chapter 5

Design and Implementation

This project employs a feature extraction technique called bag of words (BoW) to extract features from movie reviews and represent the reviews as a vector space model or feature vector. In the next phase it uses Naive Bayes machine learning algorithm to classify the movie reviews (represented as feature vector) into positive and negative. Next, an undirected weighted graph is constructed from the pairwise semantic similarities between classified review sentences in such a way that the graph nodes represent review sentences, while the edges of graph indicate semantic similarity weight. The weighted graph-based ranking algorithm (WGRA) is applied to compute the rank score for each review sentence in the graph. Finally, the top ranked sentences (graph nodes) are chosen based on highest rank scores to produce the extractive summary.

5.1 Data Preprocessing

5.1.1 Removal Of Html Tags



```
for review in range(0,length):
    single_review = data[review]
    #removing html tags:
    single_review = re.sub('<.*?>',' ',single_review)
    #removing special characters (punctuation) '@,! ' e.t.c.
    single_review = re.sub('\W',' ',single_review)
    #removing single characters
    single_review = re.sub('\s+[a-zA-Z]\s+',' ', single_review)
    #substituting multiple spaces with single space
    single_review = re.sub('\s+',' ', single_review)
    #removing stop words
    #single_review=single_review.lower()
    #word_tokens = []
    word_tokens = word_tokenize(single_review)
```

Figure 5.1: Removal of html tags and Special characters

For any sophisticated string extraction operation, regular expressions are the most common and powerful way. Regex, which is widely used in data mining and string matching algorithms, may be utilised

to find string patterns between HTML elements with ease.

In this Project python regex operations are used to remove html tags.

5.1.2 Word Tokenization and Removal of Stop Words:

In this project NLTK Tokenizer Package is used to tokenize text. Tokenizers divide strings into lists of substrings. NLTK also provide standard set of stopwords. These can be downloaded using nltk.corpus. With the help of these stop words one can remove the stop words. One can also add extra stop words that with respect to their domain.



```
from nltk.tokenize import sent_tokenize, word_tokenize
from nltk.corpus import stopwords


data = "All work and no play makes jack dull boy. All work and no play makes jack a dull boy."
stopWords = set(stopwords.words('english'))
words = word_tokenize(data)
wordsFiltered = []

for w in words:
    if w not in stopWords:
        wordsFiltered.append(w)

print(wordsFiltered)
```

Figure 5.2: Example for code snippet to remove stop words

5.1.3 Performing Lancaster Stemming

A code snippet for Lancaster stemming using NLTK. The code is displayed in a dark-themed editor window with a yellow background. The code imports the LancasterStemmer from nltk.stem, creates an instance, and then iterates over a list of words to print their stemmed forms.

```
from nltk.stem import LancasterStemmer
lancaster = LancasterStemmer()
words = ['eating', 'eats', 'eaten', 'puts', 'putting']
for word in words:
    print(word, " —> ", lancaster.stem(word))
```

Figure 5.3: Example for Lancaster stemming code snippet

Lancaster Stemmer is a stemmer developed and presented by Chris Paice from Lancaster University. The Lancaster stemmers are more aggressive and dynamic compared to the other two stemmers. The stemmer is really faster, but the algorithm is really confusing when dealing with small words. But they are not as efficient as Snowball Stemmers. The Lancaster stemmers save the rules externally and basically uses an iterative algorithm. Its rules are more aggressive than Porter and Snowball and it is one of the most aggressive stemmers as it tends to overstem a lot of words.

5.2 Feature Extraction using TF-IDF

In Unigram the occurrence of each word is independent of its previous word. Hence each word becomes a gram(feature) here. In Bigram that each occurrence of each word depends only on its previous word. Hence two words are counted as one gram(feature) here.

Consider the following three review text documents; These are single review sentence from each document for convenience.

Review document 1: "I loved this movie."

Review document 2: "I hated this movie."

Review document 3: "Great acting a good movie."

TABLE 1: BoW vector space model for unigrams.

Review documents	Acting	Good	Great	Hated	Loved	Movie	This	Class
Review Doc1	0	0	0	0	1	1	1	+ve
Review Doc2	0	0	0	1	0	1	1	-ve
Review Doc3	1	1	1	0	0	1	0	+ve

TABLE 2: Bag of bigram vector space model.

Review documents	Acting good	Good movie	Great acting	Hated this	Loved this	This movie	Class
Review Doc1	0	0	0	0	1	1	+ve
Review Doc2	0	0	0	1	0	1	-ve
Review Doc3	1	1	1	0	0	0	+ve

TABLE 3: Vector space model of bag of unigrams and bigrams.

Review documents	Acting	Acting good	Good	Good movie	...	Loved this	Movie	This	This movie	Class
Review Doc1	0	0	0	0	...	1	1	1	1	+ve
Review Doc2	0	0	0	0	...	0	1	1	1	-ve
Review Doc3	1	1	1	1	...	0	1	0	0	+ve

Figure 5.4: Unigrams and Bigrams

5.2.1 TF-IDF:

Phrase frequency analyses the frequency of a term one is interested in relation to the rest of the text.

There are many methods for determining frequency

1. How many times a term occurs in a document (raw count).
2. Term frequency is adjusted for the document's length (raw count of occurrences divided by number of words in the document).
3. Frequency that is logarithmically scaled (e.g. $\log(1 + \text{raw count})$).
4. Frequency of booleans (e.g. 1 if the term occurs, or 0 if the term does not occur, in the document).

The inverse document frequency method examines how frequent (or unusual) a term is in the corpus.

The IDF is computed as follows: t is the term (word) for which the frequency is determined, and N is the number of documents (d) in the corpus (D). The number of papers in which the phrase t occurs is the denominator.

TF-IDF = Term Frequency (TF) x Inverse Document Frequency (IDF)

$$TF - IDF(t, d) = TF(t, d) \times \log_{10}\left(\frac{N}{1 + df}\right)$$

In this project the parameters $\text{mindf} = 2$ and $\text{maxdf} = 0.5$ are used. Here mindf parameter is used to ignore all the words which occur in less than 2 documents. Maxdf parameter is used to ignore all the words occur more than 50 percentage of the documents. The final output matrix shape has number of reviews as rows and all possible unique words and bigrams as columns

```
[ ] #multinomial naive bayes
import sklearn
from sklearn.feature_extraction.text import TfidfVectorizer
def vectorize_train(data):
    tfidf=TfidfVectorizer(min_df=2,max_df=0.5,ngram_range=(1,2))
    text_count_matrix=tfidf.fit_transform(data)
    return text_count_matrix, tfidf

[ ] def vectorize_test(tfidf,test_data):
    test_matrix=tfidf.transform(test_data)
    return test_matrix
```

Figure 5.5: Code snippet for Unigrams and Bigrams

5.3 Multinomial NaiveBayes Classification

Consider creating a new document for evaluation. "I love this movie," says the NB classifier, who categorises it as positive or negative. Here is a brief summary of the document. The review document is first represented as a bag of unigrams and bigrams vector representation. The following equation may be used to determine the likelihood of a review document belonging to a certain class (positive or negative).

$$P(Doc/Class) = \prod_{i=1}^{Doc} P(W/c_i)$$

where Doc is the review document, —Doc— is the length of document, and P(W—ci) is the probability of a term W in a review document's given certain class (+ve or ve).

In order to classify the view document "I love this movie," it is needed to determine the probabilities of all terms(unigrams and bigrams) in the review documents labeled as positive. The probability of each term given class c_i

$$P(w_k/positive) = \frac{(n_k + 1)}{(n + |VOC|)}$$

where n_k is the number of times the term w_k occurs in positive cases and n is the total number of words in positive cases. VOC indicates the number of unique unigrams and bigrams in the review documents. Probability of the above review document's given positive case is estimated based on probabilities of all unigrams and bigrams in the review document.

$$P(\text{Positive}) = \frac{\text{number of positive review cases}}{\text{total number of review cases}},$$

$$P(\text{loved} | \text{positive}) = \frac{\text{number of times "loved" occurs in positive cases} + 1}{\text{total number of words in positive cases} + |VOC|},$$

$$(\text{this movie} | \text{positive}) = \frac{\text{number of times "this movie" occurs in positive case} + 1}{\text{total number of words in positive case} + |VOC|}.$$

Figure 5.6: Conditional Probability

The same process is repeated for negative review document. Based on the following equation, the review document is assigned to a class if the probability value of the review document's given class is maximized.

$$C_{NB} = \operatorname{argmax}_{C_i \in C} P(C_i) \prod_{w \in \text{words}} P\left(\frac{w}{C_i}\right).$$

Figure 5.7: Naive Bayes

5.4 Graph Based Review Summarization

This phase's purpose is to create a graph using categorised reviews. In previous phase categorised reviews broken into sentences. After that, find semantic similarities between review phrases and make a graph out of the pairwise semantic similarities. In this project word2vec model is used to extract word embeddings for each word in sentences in order to calculate pairwise semantic similarities between sentences. To learn word embeddings (word vectors) for each word in all phrases, Google's pretrained word2vec model is used. The word2vec model is a neural network-based implementation that learns distributed vector representations of words from a continuous bag of words, according to Google. The model was built using around 100 billion words from the Google News dataset. The default length of the word vector is 300 features. To represent sentences as vectors, take the average of all word embeddings in word2vec's vocabulary and disregard the words that aren't in the vocabulary. The degree to which two phrase vectors are semantically similar.

Cosine similarity, as shown in equation, is used to determine A and B. Cosine similarity is a dot product of two vectors; it is one if the cosine angle between two phrase vectors is zero, and less than one for any other angle.

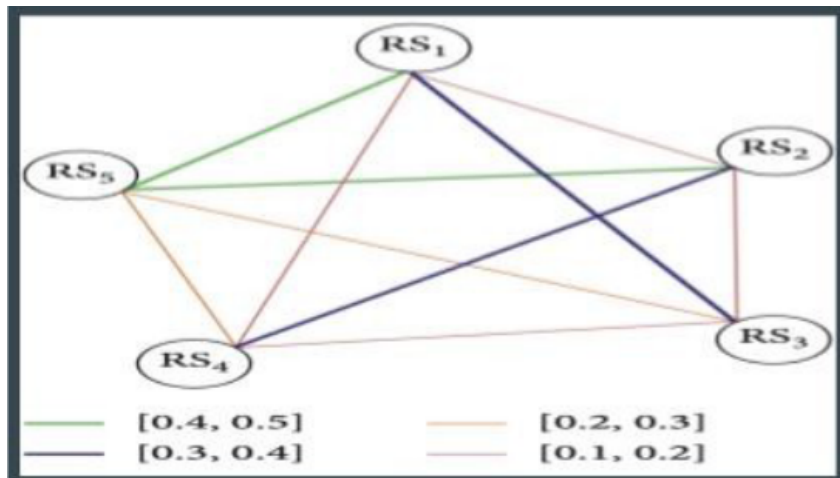


Figure 5.8: Undirected Weighted Graph

$$sim(A, B) = \frac{A.B}{||A||.||B||}$$

Following the computation of the semantic similarity score for each pair of phrases, the similarity scores of the sentences are used to create a semantic similarity matrix M_{ij} . The semantic similarity matrix created in the preceding phase is then used to form an undirected weighted network. The network is constructed in such a manner that if the similarity weight $sim(A_i, B_j)$ between nodes A_i and B_j ($i \neq j$) is larger than 0, a connection is formed between them; otherwise, no link is formed. This project is only interested in substantial phrase similarity in this investigation, thus there is establishment of an empirical similarity criterion of 0.5. As a result, a connection is only made between nodes with a similarity score of 0.5; otherwise, no link is established between nodes. To eliminate phrase repetition in summary creation, two nodes with a similarity score larger than 0.5 are expected to be semantically comparable and are not included to the network. The semantic similarity $sim(A_i, B_j)$ between two nodes A_i, B_j (i, j) is determined using equation . An undirected weighted graph is shown above. Various coloured solid bars on the edges of the graph indicate different ranges of semantic similarity weights. The network nodes relate to the review phrases indicated by RS_i , where i is a number between 1 and n .

Let formally define the classified review document to be D . Assume $G(V, E)$ is an undirected weighted network with n nodes/vertices V linked by edges E , which reflect the relationships between the classified review sentences in document set D . Let V be a collection of vertices, with each vertex v_i representing a classified review phrase in D . Assume that E is a collection of edges, with each edge e_{ij} representing the semantic similarity weight between the two vertices v_i and v_j . Next, apply weighted graph-based ranking algorithm (WGRA) that considers edge weights, which correlate to semantic similarity between sentences. $WGRA(v_i)$ is the significance score of the node/vertex v_i under consideration. The vertex/node salience score is calculated using all linked vertices (sentences) as well as the salience scores of the connected vertices (sentences). It is expressed as follows:

$$\text{WGRA}(v_i) = (1 - d) + d * \sum_{v_j \in \text{In}(v_i)} \frac{\text{WGRA}(v_j) \cdot w_{ji}}{\sum_{v_k \in \text{Out}(v_j)} w_{jk}},$$

Figure 5.9: Importance of a node in final summary

where the damping factor is set at 0.85 in most cases. The number of vertices pointing to the given vertex v_i is provided by $\text{In}(v_i)$. The weight associated with the edge connecting nodes v_i and v_j is w_{ji} , and $\text{out}(v_j)$ is the number of outgoing connections from the vertex v_j . The weights associated with outgoing linkages from vertex v_j are represented by w_{jk} . In terms of implementation, the weighted graph based ranking algorithm (WGRA) begins by assigning a rank score of 1 to all graph nodes/vertices. The programme then calculates the number of related nodes/vertices to the currently selected node. The approach computes the relevance of each linked vertex in two phases after finding the number of related nodes/vertices to the current node/vertex.

The weights associated with non-going links are aggregated once the outgoing links from a connected vertex are counted. The rank score of a specific node/vertex is computed by taking into account the number of nodes/vertices connected to it as well as the salience values of the related vertices. The WGRA employs equation to generate new ranking scores for the nodes/vertices once the salience scores of the related nodes/vertices have been collected. The salience scores for the nodes/vertices are continuously computed by the algorithm until convergence is reached. The iteration/ranking technique achieves convergence when the difference between the rank scores generated for any vertices(sentences) at two different locations is less than a certain threshold 0.0001 in this project. After the algorithm gets converged, the rank scores attained for vertices of the graph are sorted in reverse order. Finally, the rank scores attained for vertices (sentences) of the graph are sorted in reverse order. The next step is to choose the top ranked sentences for extractive summary generation. In this project, top 20 high ranked sentences are chosen for summary.

5.5 Novelty

This is the important phase where Google’s Universal Sentence Encoder(USE) is used for embedding. USE has two models for performing sentence embedding one is Transformer model and other is DAN(Deep Averaging Network). The DAN option computes the unigram and bigram embeddings first and then averages them to get a single embedding. This is then passed to a deep neural network to get a final sentence embedding of 512 dimensions. Two variants of the encoding models allow for trade-offs between accuracy and compute resources. This project uses above DAN variant for text summarization.

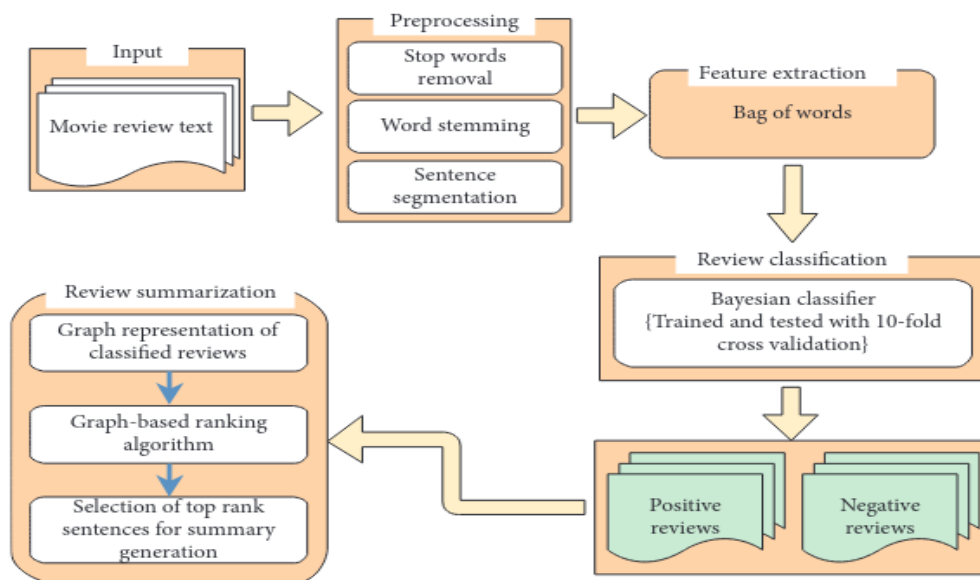


Figure 5.10: Procedure of Performing Summarization

Chapter 6

Results

```
[ ] from sklearn import metrics
    predict=model_highest.predict(test_vector)
    accuracy_score = metrics.accuracy_score(predict, y_test)
    print(accuracy_score)

0.8862
```

Figure 6.1: Accuracy Of Multi Naive Bayesian Model

```
1 compared to heath's other films like '10 things i hate about you' and even 'brokeback mountain' proves that this man could actually act, doesn't n
2 that's how difficult of a role this was, and that's why ledger's performance is so great.this isn't an action movie.
3 many of the imdb top movies deal with the good old battle between good and evil and the world-weariness of the hero or anti-hero, but this film la
4 it is very well made and gripping with some great performances, but personally i would put casablanca, wizard of oz, amadeus and it's a wonderful
5 in my opinion, the best part of christopher nolan's movies has always been his manipulation of time.
```

Figure 6.2: Positive Review Summary of Dark Knight Movie

```
1 i had to watch the original movie afterwards which has a plot, interesting characters and a batman that doesn't sound ridiculous every time he ope
2 i'm all for big surprises but this was just too unbelievable, and i'm saying that about a film that has a man dressing up as a giant bat!
3 i dislike super-heroes (perhaps ..because i have a brain) however--lots of people love this kind of trash..heath ledger won an oscar for his ridic
4 and i could go on but you probably all agree but are too blinded by hype and the fact that ledgers dead to see that this is a pretty average film.
5 it's hard to make out what's happening in the scenes set at night(and there's a lot of them).an ending which had godfather written all over it was
6 the quiet ending, the characters were forced out of the film for some reason, some dialogue that makes no sense.
```

Figure 6.3: Negative Review Summary of Dark Knight Movie

Chapter 7

Conclusion

Movie review mining and summary is a difficult issue, and this work takes a fresh approach to movie review summarization. There have been few study efforts in the field of movie reviews. This study offered a method for classifying and summarising movie reviews using machine learning and graph-based ranking. The suggested technique is universal and may be used to any domain by just giving the training data for that area. When both unigrams and bigrams were utilised as features in the context of movie review sentiment classification, it was discovered that the Naive Bayes classifier performed quite well when compared to the benchmark approach. When the frequency of features (unigrams and bigrams) was weighted with IDF, the classifier's performance increased even more. Finally, in order to present a gist of a massive volume of movie reviews, this research employed a semantic graph-based technique to summarise the classified movie reviews. Based on the empirical data, the suggested technique outperforms current state-of-the-art summarization models.

Bibliography

- [1] Atif Khan , Muhammad Adnan Gul, Mahdi Zareei , R. R. Biswal, Asim Zeb L. “*Movie Review Summarization Using Supervised Learning and Graph-Based Ranking Algorithm*” .
- [2] A. F. Alsaqer and S. Sasi, “*Movie review summarization and sentiment analysis using rapidminer,*” in Proceedings of 2017 International Conference on Networks Advances in Computational Technologies (NetACT), pp. 329–335, Trivandrum, India, July 2017.
- [3] A. Mahajani, V. Pandya, I. Maria, and D. Sharma, “A comprehensive survey on extractive and abstractive techniques for text summarization,” in Ambient Communications and Computer Systems, pp. 339–351, Springer, Berlin, Germany, 2019
- [4] Y. Liu and M. Lapata, “Text summarization with pretrained encoders,” 2019, <https://arxiv.org/abs/1908.08345>
- [5] Pang and L. Lee, “Opinion mining and sentiment analysis,” Foundations and Trends in Information Retrieval, vol. 2, no. 1- 2, pp. 1–135, 2008.
- [6] Y.-H. Tseng, Y.-M. Wang, Y.-I. Lin, C.-J. Lin, and D.-W. Juang, “Patent surrogate extraction and evaluation in the context of patent mapping,” Journal of Information Science, vol. 33, no. 6, pp. 718–736, 2007.