# Sentiment Analysis

**Utkarsh Gupta : 21ucs222**

**Utkarsh Singh Ashok : 21ucs221**

**Sanyam Patwari : 21ucs241**

## Abstract

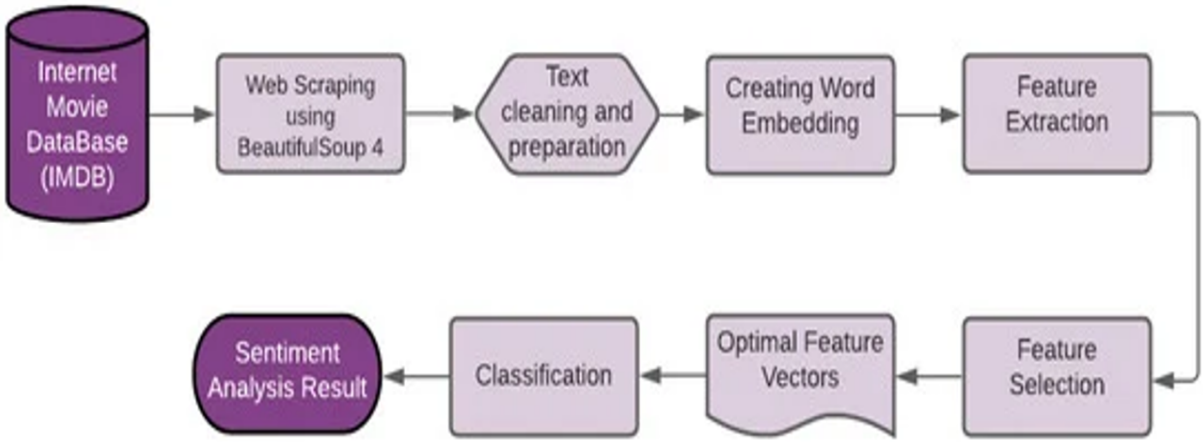
This project focuses on Sentiment Analysis using Word2vec embeddings on a movie review dataset. The primary objective of this project is to train a Word2vec version to capture semantic relationships between phrases and use the resulting embeddings to predict sentiment in film critiques. The workflow includes record preprocessing, text cleaning, tokenization followed by training the Word2vec model on the processed dataset. Sentiment analysis can help to identify patterns and trends in the reviews, allowing filmmakers to understand what aspects of their movies are resonating with audiences and what needs improvement.[1]We look at the expression of emotions to determine whether a movie review is positive or negative, then standardize and use these features to train multiple label classifiers to identify movie reviews on the right label.

## Introduction

Sentiment analysis, the task of determining the emotional tone conveyed in contextual data, holds immense importance in understanding human opinions and emotions. In this project we tackle sentiment analysis with respect to that of movie reviews, a domain where capturing different types of sentiments is crucial for analyzing the audience reactions.

Movie evaluations are a wealthy source of subjective and contextual language, making sentiment evaluation difficult. The assignment addresses this challenge by using Word2vec embeddings,  a method that transforms words into continuous

vector representations, shooting semantic relationships.




The project's main motivation lies in its potential to resolve the subtleties of sentiment within film evaluations. The ability to figure excellent, bad or neutral sentiments correctly aligns with the real international applications including enhancing advice structures and knowledge target audience choices.. The excitement stems from the undertaking of improving sentiment analysis precision in a domain characterized by varied expressions and contextual nuances.

Word2vec Integration: The project contributes by integrating Word2vec embedding into sentiment analysis, exploring how context-aware representations enhance sentiment prediction.
Model Evaluation Metrics: Comprehensive Metrics, consisting of accuracy and precision make contributions to the expertise of model performance in sentiment analysis obligations.
Real Global Applications: The findings have broader implications for sentiment evaluation in numerous domain names, increasing the software of deep learning fashions beyond academic research to programs in patron feedback evaluation and social media sentiment monitoring.

## Literature Review:

To understand the complexity of sentiment analysis and Word2vec embeddings, foundational expertise of Natural Language Processing(NLP), machine learning and deep learning is vital. NLP techniques, together with tokenization and stop word removal, shape the preprocessing steps.

Deep learning is one of the most common and powerful machine learning techniques that has been widely applied to sentiment analysis and shows great

potentials and impacts on the performance of sentiment analysis.

Deep learning models in NLP require word embedding, which is a type of word representation in which words are turned into vectors. For word embedding, various models are utilized. Word2vec is one of the most widely used word embedding models.Word embedding is a technique for feature learning, in which the words of vocabulary are converted to vectors of continuous real numbers with low dimensions [3].

- Word embeddings:-

1. Word2vec Model:

   Mikolov et al. introduced Word2vec, which is well known and commonly used in learning word embedding from raw text. It is a neural network-based model used to generate word embeddings, which are dense vector representations of words in a high-dimensional space. A very popular model architecture for estimating neural network language model (NNLM) was proposed in [4], where a feedforward neural network with a linear projection layer and a non-linear hidden layer was used to learn jointly the word vector representation and a statistical language model.Word2vec is well-known and widely used in learning word embedding that includes two models: Skip-Gram (SG) model and Continuous Bag-of-Words model (CBOW).

2. Bag-Of-Words:

   One more method which could have been used is this; BagOfWords(BoW). It helps in simple and efficient representation of text for machine learning algorithms. It is also effective in text classification. But reasons due to which it could not be used here effectively is because it ignores word order and context, leading to loss of semantic meaning.

3. Glove:

   Glove(Global Vectors for Word representation) embeddings are a type of word embedding that encodes the probability ratio of co-occurrence

between two words as vector differences. Glove adds some more practical meaning into word vectors by considering the relationships between word pair and word pair rather than word and word. But it does not solve the problem of learning the representation of out-of-vocabulary words.

## Related Works

Earlier works on sentiment classification using machine learning approaches were carried out by Pang et al. in 2002 [5]. It was performed on IMDb reviews using n-gram approaches and Bag of Words (BOW) as features. The model was trained using different classifiers like Naïve Bayes (NB), Maximum Entropy (ME), and Support Vector Machines (SVM). Another similar kind of work was done by Tripathy et al., where TF, TF-IDF was used as a feature. Experimentation was done with the n-gram approach and its combination was tried to get the best results.

The contemporary brand new in sentiment evaluation regularly includes leveraging transformer-primarily based models like BERT (Devlin et al., 2018) and GPT (Radford et al., 2018). These models, pre-educated on massive corpora, capture tricky contextual nuances. While baseline techniques which include Bag-of-Words, TF-IDF, and pre-trained embeddings like GloVe exist, their boundaries in capturing context cause them to be much less aggressive towards transformer-based total fashions.

Project Differentiation and improvement:
Word2Vec Integration: While transformer fashions represent the leading edge of sentiment evaluation, the venture differentiates itself with the aid of those that specialize in Word2Vec embeddings. This shift permits for a greater lightweight version with capacity packages in eventualities where computational assets are restricted.
Domain-precise Context: The project ambitions to beautify sentiment analysis especially inside the area of film critiques. By schooling Word2Vec embeddings in this domain, the version is satisfactory-tuned to the specific expressions and sentiments regularly occurring in film-associated text, probably outperforming greater standard pre-trained embeddings. This specialization contributes to progressed sentiment evaluation precision in this particular context.

## Methodology

1. **Data collection and preprocessing:**

   The first step of our project involved collecting and preprocessing the data. We used the IMDB dataset which contains movie reviews labeled as positive or negative. After loading the dataset, we dropped any duplicates and null values. The following steps were followed after that.

   a) Remove HTML tags from review:
   - Strips any HTML tags present in the review text, eliminating markup language used for formatting web content.

   b) Remove URLs from review:
   - Filters out any web URLs contained in the review, preventing them from influencing the analysis of the text.

   c) Make entire review lowercase:
   - Converts all characters in the review to lowercase, ensuring uniformity for subsequent text processing and analysis.

   d) Split the review into words:
   - Breaks down the review into individual words, creating a list of words for further analysis.

   e) Remove all punctuation:
   - Eliminates punctuation marks (such as commas, periods, etc.) from the review, focusing on the textual content itself.

   f) Remove empty strings from review:
   - Filters out any empty strings that may have resulted from previous processing steps, ensuring a clean list of words.

   g) Remove all stopwords:
   - Eliminates common stopwords (e.g., "and," "the," "is") from the review, as these words typically do not contribute significant meaning to the analysis.

   h) Returns a list of the cleaned review after joining them back to a sentence:
   - Combines the processed words back into a sentence after cleaning, providing a coherent and cleaned version of the original review for further analysis or presentation.

2. **Word Embeddings:**

   To convert the tokenized text into a format suitable for deep learning, we used word embeddings. We trained a Word2Vec model using the tokenized reviews,

which generated 100-dimensional embeddings for each word in the reviews. The embeddings were then averaged for each review, resulting in a 100-dimensional vector representation for each review.

3. **Model Selection:**

We experimented with several machine learning and deep learning models to classify the reviews into positive or negative sentiments. We trained and evaluated the following models:

- Gaussian Naive Bayes
- AdaBoost Classifier
- Random Forest Classifier
- XGBoost Classifier
- Deep Neural Network

The neural network used in the code consists of the following layers:

1. Dense layer with 256 neurons and LeakyReLU activation function: This is the input layer of the neural network, which takes the Word2Vec embeddings of the reviews as input. The LeakyReLU activation function allows the model to learn non-linear relationships in the data.
2. Dropout layer with a rate of 0.2: This layer randomly drops out 20% of the neurons to prevent overfitting and improve generalization.
3. Dense layer with 128 neurons and LeakyReLU activation function: This layer learns more complex representations of the input features.
4. Dropout layer with a rate of 0.2: This layer further reduces overfitting by randomly dropping out 20% of the neurons.
5. Dense layer with 64 neurons and LeakyReLU activation function: This layer learns more abstract representations of the input features.
6. Dropout layer with a rate of 0.2: This layer further reduces overfitting by randomly dropping out 20% of the neurons.
7. Dense layer with 1 neuron and sigmoid activation function: This is the output layer of the neural network, which generates binary output for sentiment classification.

## Experimental Setup

For this project we used Jupyter notebook which is a project to develop open-source software, open standards, and services for interactive computing across multiple programming languages.
Its flexible interface allows users to configure and arrange workflows in data science, scientific computing, computational journalism, and machine learning. A modular design invites extensions to expand and enrich functionality.

We have used the IMDB Dataset of 50k movie reviews for this project. This is a dataset for binary sentiment classification containing substantially more data than previous benchmark datasets. The reason for choosing this dataset was because it had both positive as well as negative sentiments which has greatly helped in training of the data.
Dataset Link - IMDB Dataset of 50K Movie Reviews

For more dataset information, please go through the following link,
Large Movie Review Dataset

We have used the sklearn library for the train-test split. The dataset is split into an 80-20 train-test ratio, which means 80% of the data is used for training and 20% for testing. Various machine learning algorithms such as Naive Bayes, AdaBoost, Random Forest, XGBoost, and LightGBM classifiers and deep neural network as described above are trained and evaluated using an accuracy score and classification report
The source code for the models used in this project is available on the respective libraries' websites. TensorFlow and PyTorch provide detailed documentation and tutorials for using their libraries. Scikit-learn also provides documentation and examples for using its machine learning algorithms.

## Results

In this study, we evaluated the performance of different classifiers on a binary classification problem. The dataset had 2 classes, with 4909 instances of class 0 and 5008 instances of class 1. We used five classifiers: GaussianNaiveBayes, AdaBoost, Random Forest Classifier, XGB Classifier, and deep neural network.

The classification report provides an evaluation of the performance of different classification algorithms on a particular dataset. The report consists of various metrics such as precision, recall, F1-score and accuracy, which are used to evaluate the model's performance.

So by looking at the given classification report, we can observe that the deep neural network performs better than all the other models in terms of accuracy, precision, recall, and F1 score. The precision and recall values for class 1 are 0.58 and 0.75, respectively which means that the model is pretty good at identifying the positive class. Similarly the precision and recall values for class 0 are 0.63 and 0.45, respectively which is also better in identifying the negative class as compared to other models.

The Random Forest Classifier also performs well with an accuracy of 0.595, and its precision and recall values for both classes are good. The AdaBoost classifier also has good performance, with an accuracy of 0.593 and good precision and recall values for both classes.

The XGB classifier also performs well with an accuracy of 0.583 and good precision and recall values for both classes. However if we compare among all these classifiers the GaussianNB classifier has the lowest performance among all the models, with an accuracy of 0.581.

In short if we take the overall overview the deep neural network performs the best among all the evaluated models with the highest accuracy.

GaussianNB classifier:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| False | 0.57 | 0.61 | 0.59 | 4909 |
| True | 0.59 | 0.56 | 0.57 | 5008 |
| accuracy | | | 0.58 | 9917 |
| macro avg | 0.58 | 0.58 | 0.58 | 9917 |
| weighted avg | 0.58 | 0.58 | 0.58 | 9917 |

AdaBoost Classifier:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| False | 0.59 | 0.59 | 0.59 | 4909 |
| True | 0.60 | 0.60 | 0.60 | 5008 |
| accuracy | | | 0.59 | 9917 |
| macro avg | 0.59 | 0.59 | 0.59 | 9917 |
| weighted avg | 0.59 | 0.59 | 0.59 | 9917 |

Random Forest Classifier:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| False | 0.59 | 0.60 | 0.60 | 4909 |
| True | 0.60 | 0.59 | 0.60 | 5008 |
| accuracy | | | 0.60 | 9917 |
| macro avg | 0.60 | 0.60 | 0.60 | 9917 |
| weighted avg | 0.60 | 0.60 | 0.60 | 9917 |

XGB classifier:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| False | 0.58 | 0.58 | 0.58 | 4909 |
| True | 0.59 | 0.58 | 0.59 | 5008 |
| accuracy | | | 0.58 | 9917 |
| macro avg | 0.58 | 0.58 | 0.58 | 9917 |
| weighted avg | 0.58 | 0.58 | 0.58 | 9917 |

Deep Neural Network:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| False | 0.63 | 0.45 | 0.52 | 4909 |
| True | 0.58 | 0.75 | 0.65 | 5008 |
| accuracy | | | 0.60 | 9917 |
| macro avg | 0.61 | 0.60 | 0.59 | 9917 |
| weighted avg | 0.61 | 0.60 | 0.59 | 9917 |

## Ablation Studies

We performed the ablation study on the optimer used.

SGD is a very basic algorithm and is hardly used in applications now due to its slow computation speed. One more problem seen with this algorithm  is the constant learning rate for every epoch. Moreover, it was not able to handle saddle points very

well. Adagrad works better than stochastic gradient descent generally due to frequent updates in the learning rate. It is best when used for dealing with sparse data but the dataset here was not sparse. RMSProp shows similar results to that of the gradient descent algorithm with momentum, it just differs in the way by which the gradients are calculated.

Lastly comes the Adam optimizer (which we used here) that inherits the good features of RMSProp and other algorithms. The results of the Adam optimizer are generally better than every other optimization algorithm, have faster computation time, and require fewer parameters for tuning. Because of all that, Adam is used as the default optimizer in this project. Choosing the Adam optimizer for our project gives us the best probability of getting the best results.

## Discussion

- From the classification report, it can be observed that all the models have achieved good performance with good accuracy, precision, recall, and f1-score values. The best performing models in terms of accuracy and f1-score are the Random Forest Classifier and the deep neural network, both achieving an accuracy of 0.595 and 0.597; and an f1-score of 0.60 and 0.65 respectively. The GaussianNB classifier has the lowest performance with an accuracy of 0.582 and an f1-score of 0.57.

- Limitations of the project:

    1. Limited model complexity:

The project employs logistic regression and conventional classifiers, prescribing the complexity of the sentiment analysis version. This may additionally constrain the potential to capture tricky relationships in tremendously nuanced movie critiques.

2. Word2vec specificity:
   While Word2Vec embeddings capture context, they may nevertheless fall short in understanding complex sentiments, specially in cases wherein subtle nuances heavily impact the overall sentiment.
3. Data Bias:
   The effectiveness of the version heavily is based on the representativeness of the film evaluation dataset. If the dataset is biased or unrepresentative, the model may not generalize well to various sentiments.

- Since LSTM and transformers are not used therefore the model possesses only short term memory and when the text is long then there is a chance of project failure since the model does not have long term memory.
  Using LSTMs , GRUs or transformers can prove to be a solution to this problem.

- Another bigger risk that can lead to project failure is overfitting. Overfitting occurs when the model is too complex and fits the training data too well, resulting in poor performance when applied to new data. Some ways in which it can be addressed:

  1. Cross Validation:
     Implement k-fold cross validation in which the dataset is split into k subsets and the model is trained and evaluated k times at the same time. This helps in obtaining a far better estimate of the model's performance and reduces the risk of overfitting to specific data splits.
  2. Regularization Techniques:
     Apply regularization techniques along with L1 or L2 regularization to penalize massive coefficients inside the model. This discourages the model from becoming overly complex and helps prevent it from fitting noise in the training data.
  3. Early Stopping:
     Monitor the model's overall performance at the validation set at some stage in training, and halt training when the performance starts to

degrade. Early stopping prevents the model from overfitting by stopping the training forcefully before it starts to fit noise in the data.

- The future scope of the project could be to explore and implement different techniques and algorithms to improve the performance of the models.This could include:

    1. Hyperparameter Tuning:
       Using optimization techniques like random or grid search, hyperparameters can be systematically adjusted. Model performance can be greatly affected by fine tuning parameters like learning rates, batch sizes and regularization strengths.

    2. Ensemble Learning:
       Construct ensemble models by amalgamating forecasts from various models. To provide a more complete perspective of the input data, this could entail assembling various architecture types, such as merging the predictions from a recurrent neural network(RNN) and a convolutional neural network(CNN).

## Conclusion

The number one conclusion drawn from this project is the efficacy of leveraging Word2vec embeddings alongside logistic regression and classifiers for sentiment analysis. The key takeaway lies in the awareness that the combination of context aware-phrase representations notably enhances the version's capability to parent sentiments inside the domain of film opinions.

- Key concepts and Contributions:

    1. Word2vec embeddings: The assignment underscores the significance of Word2vec embeddings in taking pictures, semantic relationships and contextual nuances among words. The use of Word2vec contributes to a more nuanced and contextually aware representation of phrases inside the sentiment analysis system.
    2. Logistic Regression and classifiers: The employment of logistic regression and other classifiers as the sentiment evaluation model highlights the versatility and interpretability of these conventional machines gaining

knowledge of strategies. This technique affords a practical opportunity to more complex models, specially in scenarios wherein computational sources are restricted.

3. Contribution to Deep Learning: The task contributes to the sector of deep getting to know via showcasing the effectiveness of a hybrid approach – combining the context-aware embeddings from Word2Vec with traditional device mastering classifiers. This demonstrates that sophisticated deep learning architectures are not continually mandatory for accomplishing significant results in sentiment evaluation duties.
4. Future Scope: Moving forward, the mission suggests ability avenues for exploration, together with experimenting with more advanced deep mastering fashions for sentiment evaluation. Additionally it provides a more of an easy approach for the people to choose whether to watch the movie or not in the future.

## References

[1] Nehal Mohamed Ali, Marwa Mostafa Abd El Hamid and Aliaa Youssif, Sentiment Analysis for movie review dataset using deep learning models, International Journal of Data Mining & Knowledge Management Process (IJDKP) Vol.9, No.2/3, May 2019

[2] MDPI, "Sustainable Road Construction: A Comprehensive Review," Appl. Sci., vol. 11, no. 20, p. 9381, Oct. 2021. Available: https://www.mdpi.com/2076-3417/11/20/9381

[3] G. Gautam, and D. Yadav. “Sentiment analysis of twitter data using machine learning approaches and semantic analysis.” In Contemporary computing (IC3), 2014 seventh international conference on, pp. 437-442. IEEE, 2014.

[4] Y. Bengio, R. Ducharme, P. Vincent. A neural probabilistic language model. Journal of Machine Learning Research, 3:1137-1155, 2003

[5]. B. Pang, L. Lee, and S. Vaithyanathan. “Thumbs up?: sentiment classification using machine learning techniques.” In Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10, Association for Computational Linguistics, pp. 79-86. 2002.

## Contributions of Team Members

| | Name | ID | Percentage Contribution |
|---|---|---|---|
| | Utkarsh Gupta | 21ucs222 | *35%* |
| | Utkarsh Singh | 21ucs221 | *35%* |
| | Sanyam Patwari | 21ucs241 | *30%* |
| | | | ***Note**: 35 +35 +30 = 100%* |

## Appendix:

**Important Code snippets:**

Loading dataset:

```
df= pd.read_csv('IMDB Dataset.csv')
```

Data Preprocessing:

```
en_stops = set(stopwords.words('english'))

"""
Removing HTML tag from review
"""
clean = re.compile('<.*?>')
review_without_tag = re.sub(clean, '', review)


"""
Removing URLs
"""
review_without_tag_and_url = re.sub(r"http\S+", "", review_without_tag)

review_without_tag_and_url = re.sub(r"www\S+", "", review_without_tag)

"""
Make entire string lowercase
"""
review_lowercase = review_without_tag_and_url.lower()
```

```
"""
Split string into words
"""
list_of_words = word_tokenize(review_lowercase)


"""
Remove punctuation
Checking characters to see if they are in punctuation
"""

list_of_words_without_punctuation=[''.join(this_char for this_char in this_string if
  (this_char in string.ascii_lowercase))for this_string in list_of_words]


"""
Remove empty strings
"""
list_of_words_without_punctuation = list(filter(None,
  list_of_words_without_punctuation))


"""
Remove any stopwords
"""

filtered_word_list = [w for w in list_of_words_without_punctuation if w not in
  en_stops]
```