A Report On

Sentiment Analysis
with Neural Networks

By

Abhirath Anupam Joshi	2019A7PS1136P
Amulya Gupta	2019D2TS1278P
Yash Gupta	2019A7PS1138P

Prepared in Partial Fulfillment of the Course

CS F469 Information Retrieval

Birla Institute of Technology and Science (BITS), Pilani

April, 2022

Table of Contents

Introduction

With the recent advances in neural networks and the hardware technology needed to support the computations they demand, the field of Information Retrieval - which has traditionally been associated with search engines and index construction - is witnessing an upward trend in the research toward Emotion AI, where the term *Emotion AI* refers to the subset of artificial intelligence that measures, understands, simulates, and reacts to human emotions [1].

Sentiment Analysis, sometimes also referred to as opinion mining, finds itself at the junction of Emotion AI and traditional IR, where the efficient index construction and text vectorization techniques meet the more recent neural networks - including, the increasingly popular convolutional neural networks and RNN-based sequence models - "to systematically identify, extract, quantify, and study affective states and subjective information" [2] from the user input.

Although a pipeline for sentiment analysis may benefit from inclusion of visual - for instance, a photograph of the tweeter's face, showcasing their emotions at the time of posting the tweet - and audio - variations in amplitude and frequency of the speaker - cues, we shall focus solely on the more traditional view of sentiment analysis where it is applied on textual data. Particularly, we shall take up the ideas in (Haque et al) [3] to build neural network based classifiers for binary sentiment analysis for movie reviews, designating them as *positive* or *negative* reviews. We will explore the three different neural network architectures - LSTM, CNN, and LSTM-CNN - used for the task and benchmark them against each other and classifiers used in related literature.

1. Problem Statement

The general classification problem in IR is formulated as follows: given a set of training documents, D, and the corresponding labels, c_D, use a supervised learning method, Γ, to learn a classifier, γ: $X \rightarrow C$ - where X is the document space and

$C = \{c_1, c_2, ..., c_N\}$ is the set of all classes involved - that maps a document $d \in X$ to its corresponding class c [4].

In our version of the sentiment analysis problem, we define X to be the set of all IMDb movie reviews, D to be our training partition of the dataset, and $C = \{pos, neg\}$ for *positive* and *negative* reviews. Furthermore, we use three supervised learning methods Γ_1 LSTM, Γ_2 CNN, and Γ_3 LSTM-CNN to learn three instances of the classifiers $\gamma_1 = \Gamma_1(D)$, $\gamma_2 = \Gamma_2(D)$, $\gamma_3 = \Gamma_3(D)$ to classify any new - or test - movie reviews as *positive* and *negative*. Thus, the sentiment analysis task has been posed as a binary classification problem.

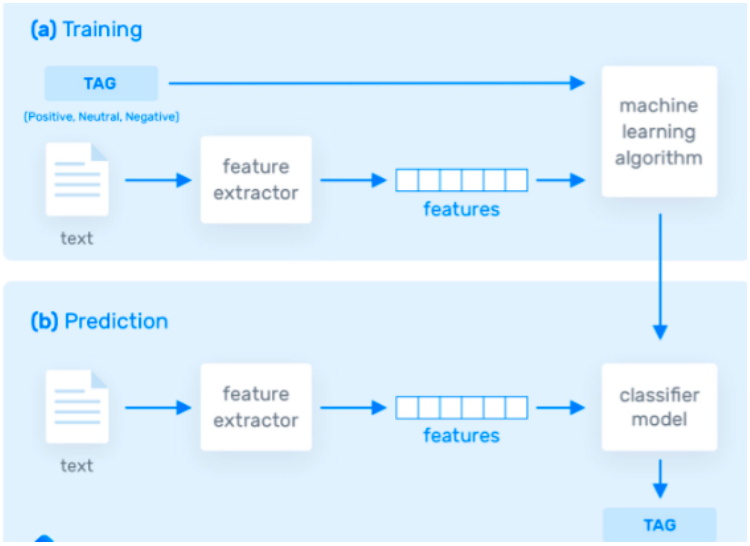


Figure 1.1 Basic organization of the sentiment analysis task as a classification problem.
Image Credit: MonkeyLearn.

To put it in simpler terms, three neural network based models are trained on the IMDb movie reviews dataset to identify the overall sentiment - *positive* or *negative* - contained in the review. Figure 1.1 succinctly encapsulates the problem, except that we do not consider *neutral* sentiment in the present study, for the IMDb dataset is known to be highly *polar* [5].

The three models will be benchmarked against each other as per the evaluation metrics discussed in Section 6.

2. Problem Background

2.1 The Domain of Sentiment Analysis

As stated earlier, sentiment analysis is a natural language processing (NLP) task wherein the sentiment - *positive* or *negative*; *sarcasm*; *hate*; *joy* or *grief;* and the like - contained data is determined. A simple example would be to identify the sentiment in the sentence - "I am highly unsatisfied with the cab service". *Positive* or *negative?* Negative. *Anger, joy,* or *sorrow?* Anger. And so on.

The applications of sentiment analysis are numerous - often combined into other information retrieval systems.

- It often finds use in monitoring brands and products through analysis of customer feedback and product reviews.
- It can be a valuable tool in systematizing the study of the otherwise complex social networks through post/tweet analysis.
- Search engines might be augmented using sentiment analysis by providing blog posts that contain the sentiment relevant to the user's query.
- Netflix and similar platforms may use sentiment analysis to rank movies on their top pages based on the sentiments conveyed in the corresponding reviews.
- Linguists and anthropologists can greatly benefit from sentiment analysis tools for detection of sarcasm and irony to study trends in human languages.

A few of the more important subfields in sentiment analysis are listed below.

- **Emotion Detection:** This refers to the extraction of emotion(s) from a text body, such as happiness, frustration, anger, or grief. Such type of analysis can benefit from incorporation of *emoticons* and *emojis* in the analysis.

- **Graded Sentiment Analysis:** Not only is the sentiment (*positive* or *negative*) identified, but also is attached to it to a degree to measure its polarity. For instance, an online merchandise store might want to combine the analysis of the star-rating system with the written reviews, and thus,

might be interested in labels like *very bad, bad, neutral*, *good*, and *very good*, instead of just *good* or *bad* for the product reviews.

- **Aspect Based Sentiment Analysis:** Rather than just assigning an overall sentiment to the text, it identifies the sentiment associated with a certain aspect of the text. For instance, *"The movie had good action sequences, but the dialogue was poor."* will be assigned the label *bad* when we focus on the aspect *dialogue*; the same review, however, will be given the label *good* when we shift our focus to *action* instead.

- **Multilingual sentiment analysis:** Since more than 40% [4] of the textual web content is in languages other than English, it becomes imperative to develop techniques to analyze content using multiple languages. A general approach might be to identify the language associated with the text and then employ the corresponding model. With multiple languages simultaneously, however, the process complicates significantly and requires a lot of resources and preprocessing.

2.2 Problem Motivation

The problem of sentiment analysis might appear simple if we restrict ourselves to classification based on occurrence of common emotion-signaling words - *angry*, *bad*, *nice, kudos*, and the like. It is, however, not remotely as simple. Consider the tweet appearing in Figure 2.1, for instance. Despite not containing any of such flag words, the tweet clearly expresses a *negative*, or more precisely *dissatisfied*, sentiment towards Uber.

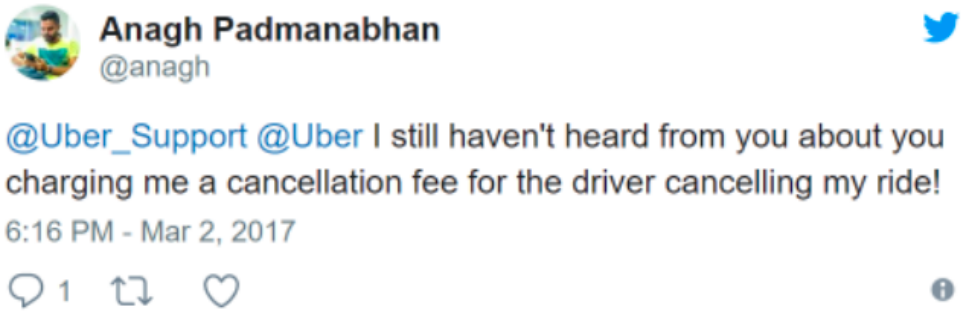


Figure 2.1: Sentiment Analysis can help companies identify popular opinions that should be addressed to maintain a favorable brand reputation.
Image Credit: Towards Data Science.

To address such problems, modern sentiment analyzers most often employ neural networks to capture not just the lexical structure of the text, but also the syntactic - sentence construction might contain cues about the sentiments - and semantic - the overall meaning of the sentence is key to the analysis - structures of the text.

The exact architecture of the model is most often specific to the problem at hand; however, we aim to present some fruitful findings about generic and most common networks used in the field of natural language processing as they are applied to the extremely popular IMDb movie reviews dataset. Specifically, we benchmark the neural networks - LSTM, CNN, and LSTM-CNN, each widely used in NLP - based on the scheme used in [3]. Such an analysis can greatly facilitate the model selection process for future endeavors in more complex sentiment analysis - and by extension, natural language processing - tasks.

2.3 Technical Issues

We develop the pipeline for preprocessing the IMDb movie reviews dataset of 50,000 reviews and training the models under question with the same using the Google Colab open environment. In the absence of a Pro or Pro+ account, the GPUs and TPUs at our disposal are severely restricted, thereby inhibiting the use of larger datasets - such as around million tweets or stackoverflow queries for similar analysis. Therefore, we stick to the moderately sized yet standard IMDb dataset. The behavior of the models with much larger data remains to be investigated.

The process of hyperparameter optimization is highly empirical and iterative. In absence of faster computing resources and manpower, we have performed practically sufficient but yet theoretically limited hyperparameter optimization to select values like the kernel size, stride sizes, dropout probabilities, and the like.

We have employed caching using Tensorflow modules to avoid long disk seeks. Furthermore, the numpy and tensorflow modules used are known to perform highly efficient vector and tensor computations. However, the effect of further parallelization with CUDA based technology remains to be investigated due to the small scale of the project and insufficient expertise in low-level programming.

3. Literature Review

The machine learning and natural language processing community has exhibited a decisive shift towards neural networks-based, especially deep networks-based, models for solving sentiment analysis and similar problems, as evidenced by a relative dearth of publications with more traditional techniques - such as Naive Bayes and Support Vector Machines - in the recent IEEE literature. We present some of the more prominent publications since 2017 in the field of sentiment analysis before proceeding to discuss our pipeline.

Sentiment Analysis is a Big Suitcase (E. Cambria et al, 2017) [6]: The authors note that most works in sentiment analysis approach it as a simple categorization problem, where it is actually a trove containing multiple NLP tasks, which collectively endeavor to capture the *'jumbled idea'* that humans use about their minds conveying emotions and opinions in natural language. The authors propose a three-layer structure, inspired by "jumping NLP curves", to achieve human-like performance in sentiment analysis.

The authors argue that a sentiment analysis system should undertake in order the following more elementary tasks (shown in Figure 3.1) to provide a response more likely to be given by an actual human being - normalization, sentence boundary disambiguation, part-of-speech tagging, text chunking, lemmatization (*the syntactics layer ends here*), word sense disambiguation, concept extraction, named entity recognition, anaphora resolution, subjectivity detection *(semantics layer ends here),* personality recognition, sarcasm detection, metaphor understanding, aspect extraction, and polarity resolution *(the pragmatics layer ends here).*

Figure 3.1: The suitcase of NLP tasks to be tackled in sentiment analysis.
Image Credit: *Please see Reference 6.*

Deep Convolutional Neural Networks for Twitter Sentiment Analysis (Zhao Jianqiang et al, 2018) [7]: The authors use the Twitter sentiment analysis data to survey public emotions about the events or products. They use a word embedding method obtained by unsupervised learning based on a large twitter corpus. This embedding uses latent contextual semantic relationships and co-occurrence statistical characteristics between words in tweets. The embeddings are combined with n-grams features and word sentiment polarity score features to form a sentiment feature set of tweets, which is fed into a deep CNN for training and predicting sentiment classification labels, achieving an average F_1-score of 0.84. Their research highlights the benefits of incorporating embeddings into the model for sentiment analysis.

Sentiment and Sarcasm Classification With Multitask Learning (N. Majumder et al, 2019) [8]: The authors argue that the tasks of sentiment classification and sarcasm detection - which are often taken up independently - are actually highly correlated and thus can be highly beneficial for each other. Sentiment is often coupled with sarcasm where intensive emotion is expressed. They build a multitask learning-based framework (Figure 3.2) using deep learning that models this correlation to improve the performance of both tasks in a multitask learning

setting, beating the stoa models on the same dataset of 994 samples in F_1-score by 3-4%.

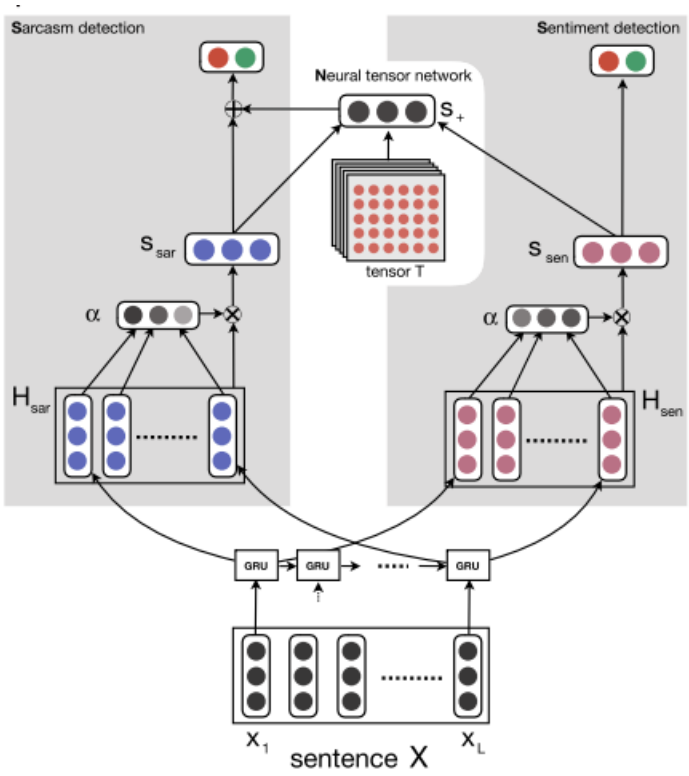


Figure 3.2: Multitask model exploiting the correlation between sentiment classification and sarcasm detection.
Image Credit: *Please see Reference 8.*

Effective LSTMs for Target-Dependent Sentiment Classification (Duyu Tang et al, 2017) [9]: The authors take up the problem of target-dependent sentiment classification. Given a sentence and a target mention, the task aims to find the sentiment polarity - *positive, negative,* or *neutral* - of the sentence towards the target. For instance the sentence, *"I bought a new camera. The picture quality is amazing but the battery life is too short"* has a *positive* polarity toward picture quality but a *negative* polarity toward battery life.

Different context words have different influences on determining the sentiment polarity of a sentence towards the target. To integrate these connections between target and context words, they develop two target dependent LSTM models, where target information is automatically taken into account and train and test them on

the Twitter sentiment dataset. Even without using syntactic parser or external sentiment lexicons, the target-dependent LSTM models achieve state-of-the-art performances - accuracy of 0.72.

Sentiment Analysis of IMDb Movie Reviews Using Long Short-Term Memory (Saeed Mian Qiaseer, 2020) [10]: The paper uses an LSTM based model to perform sentiment classification - *positive* or *negative* - on the IMDb movie reviews dataset. The author achieves a best classification accuracy of 89.9%. The python module `genism` has been used to perform text vectorization and embedding instead of the more widely used Doc2Vec and Glove models. It might be the variable responsible for a higher performance with a similar model than [3].

Sentiment Analysis Using Convolutional Neural Network, (Pan Zhou, 2015 IEEE) [13]: The authors apply a convolutional neural network to solve the sentiment analysis problem. They propose an interesting Word2vec + CNN - which might be better than other deep learning models like the RNNs and Matrix-Vector RNN - framework on a publicly available dataset of movie reviews. The strategy of using a pre-trained network with fine-tuning was adopted to train the CNN. Rather than labeling each word, the authors needed only label the whole sentences. The larger dataset, along with the proposed deep CNN and the associated training strategies, results in a better generalizability of the trained model and higher confidence of such generalizability.

Convolutional Neural Network Based Sentiment Analysis using Adaboost Combination (Yazhi Gao, Wenge Rong, 2016 IEEE) [14]: The authors of the paper propose a boosted CNN model to deal with the problem of sentiment analysis. Different from previous work, which integrates different features generated from different filters together, the proposed model separates the features by parsing the documents from different filters, and then boosts the classifiers trained on these representations and modulates them to reach high performance for binary classification tasks. Although the model shows improvements over the pre-existing methods, to better investigate the role of N-gram features in the neural network, the authors plan to integrate a sighting system into the classifier itself. Moreover, inspired by the work on DCNN, the authors wish to incorporate multiple

convolutional layers to process the data and view the features as different channels of the original information.

Lexicon Integrated CNN Models with Attention for Sentiment Analysis (Bonggun Shin, Timothy Lee, Jinho D. Choi, 2017) [15]: The paper proposes several approaches that effectively integrate lexicon embeddings and an attention mechanism to a Convolutional Neural Network for sentiment analysis. The experiments show that lexicon integration can improve accuracy, stability, and efficiency of the traditional CNN model. Multiple training results with different random seeds show the generalization of the effectiveness of using lexicon embeddings and embedding attention vectors. The training curve comparison further shows another benefit of this integration for more robust learning. The attention heatmap analysis confirms that embedding attention vectors endows CNN models with explanatory features, which gives a good understanding of how the CNN models work. The proposed attention models are applied to each single word. The authors hope that switching focus to multiple words could give more promising information. Application of the attention mechanism to multiple words at the same time is also a possible direction. Further, in order to maximize the score, an ensemble of multi layer CNN models could be applied.

Sentiment Analysis of Comment Texts Based on BiLSTM (Guixian Xu et al, 2019) [16]: The authors propose a BiLSTM based model (shown in Figure 3.3) for sentiment analysis of comments found on the web. They argue that their model captures not only the semantic information conveyed by a word but also the sentiment information. The bidirectional LSTM when used with TF-IDF representation extracts the context information of the word very effectively, while the feed forward component (latter half in Figure 3.3) extracts the sentiment information. The significance of their model's efficacy is apparent - it outperformed all prominent pre-existing architectures on the sentiment analysis task based on 15000 comments extracted from the hotel site Ctrip. RNN, CNN, Naive Bayes, and traditional LSTM could not match their model in terms of precision, recall, and F1-score (0.92, 0.93, and 0.92 respectively).

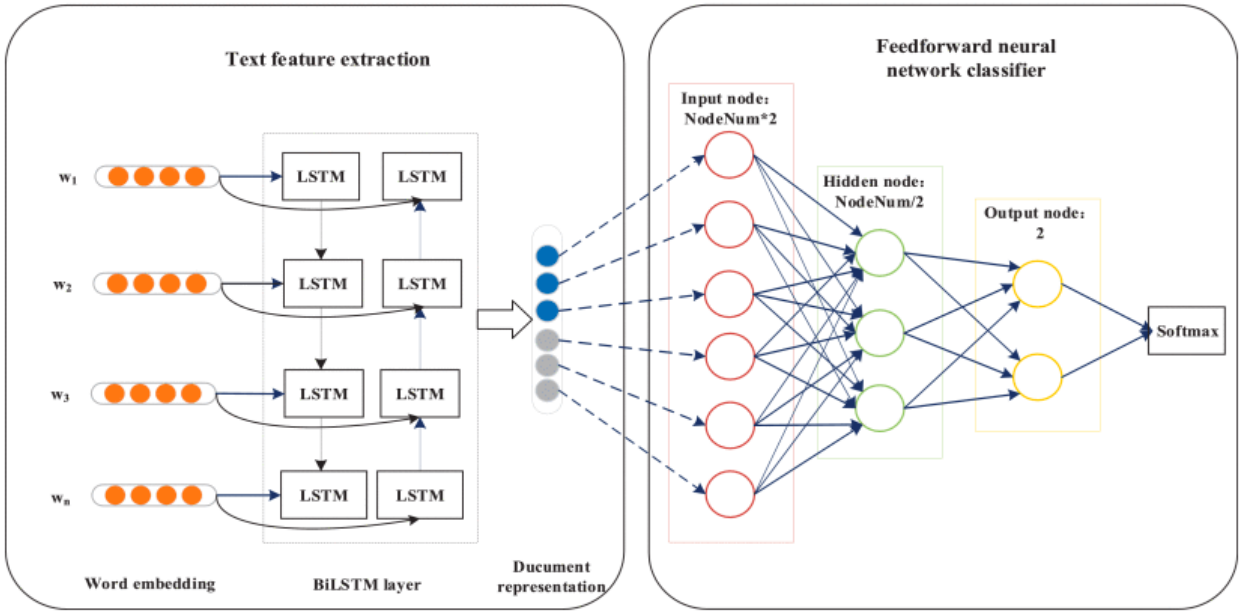


Figure 3.3: BiLSTM model for sentiment analysis of comments.
Image Credit: *Please see Reference 16.*

Hate Speech on Twitter: A Pragmatic Approach to Collect Hateful and Offensive Expressions and Perform Hate Speech Detection (Hajime Watanabe et al, 2018) [17]: Hate speech refers to the use of offensive language directed against a particular group - usually segregated on the basis of gender or race. With the ever-growing population of social media users, the filtering of hate speech from posting platforms - usually to make them safe for minors and more socially acceptable - can no longer be performed by manual human intervention. The authors propose an automated system to detect - and subsequently filter - hate speech based on collection of unigrams and patterns extracted automatically from representative texts. They achieve an accuracy of 0.87 in detecting whether a tweet is offensive or not using a collection of 2010 tweets. The words, extracted thus, may be used as features to further empower similar MLT models.

4. Research Gap

When we compare the domain of sentiment analysis to the relatively more popular domain of computer vision, we notice a relative death of ***ablation studies***. While there are plenty of models, as showcased in Section 3, that use deep learning based

techniques using RNNs (most commonly the LSTMs) and CNNs to perform sentiment classification and sarcasm detection, few perform the kind of analysis prevalent in computer vision and graphics to investigate the role of each layer.

Figure 4.1 shows a prime example of ablation studies common in computer vision and graphics for the NeRF model for recreation of a 3D scene from a sparse set of 2D images [11]. Similarly, through ablation studies of CNN for image classification, it is known that earlier CNN levels extract low-level features like presence or absence of horizontal and vertical lines, while later CNN layers can capture much more complex features including the face of a person.

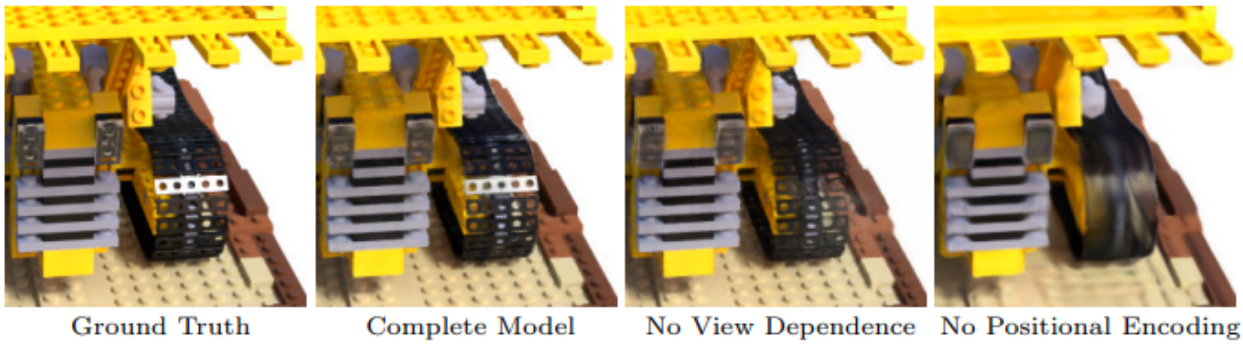


Figure 4.1: Ablation studies for NeRF - investigating the roles of positional encoding and view dependence.
Image Credit: *Please see Reference 11.*

Similar ablation analysis for sentiment analysis, and natural language processing in general, can reveal interesting facts about the common components of NLP models, thereby guiding researchers in the development of future models.

The present paper [3] does not address the problem of model ***overfitting***. While the results for the test partition are impressive for [3] when taken in isolation, a relatively big - generally more than 5% - gap in accuracy between training and testing sets can be worrisome when considering the deployment of models to real-world settings.

Speaking of generalization, the questions like *"Can a model trained on IMDb dataset be generalized to Netflix or Prime reviews?"* mostly remain unanswered

due to a lack of ***out-of-distribution generalization*** of model studies in sentiment analysis. At such a juncture, the questions like *"Can a model trained on IMDb dataset be generalized to Flipkart product reviews?"* remains distant.

The paper [3] evaluates the models based on performance metrics like accuracy, AUC, and F_1-score. It fails, however, to take into account the time factors - the ***model development time and training time***. Certain CNN based models can take significantly longer to get the hyperparameter optimization done than RNN based models, thereby separating the two in terms of developers' time. On the other hand, CNN based models - owing to a much lower number of parameters - train much faster than RNNs and fully-connected networks, thereby placing them ahead in terms of training time especially when deploying for large amounts of data. Thus, a performance comparison based solely on performance metrics seems superficial.

5. System Description

The present section covers the various structural components of our models and data procurement and preprocessing strategies. We also cover the unrelated (for the models used) yet important task of index construction based on Bag-of-word and TF-IDF models and subsequent calculation of similarities. We must emphasize though that our neural network models - following the methodology proposed in [3] - use Tensorflow's efficient `TextVectorization` and `Embedding` layers instead of the traditional indexes for vector representation of the reviews.

5.1 Overall System Design

Figure 5.1 shows the overall block diagram of the implemented system. The system has two chief independent components - *demonstration of index creation with tf-idf and BOW* and *sentiment analysis with LSTM, CNN, and LSTM-CNN models powered with Word Embedding.* Let us briefly discuss the plan depicted in Figure 5.1.

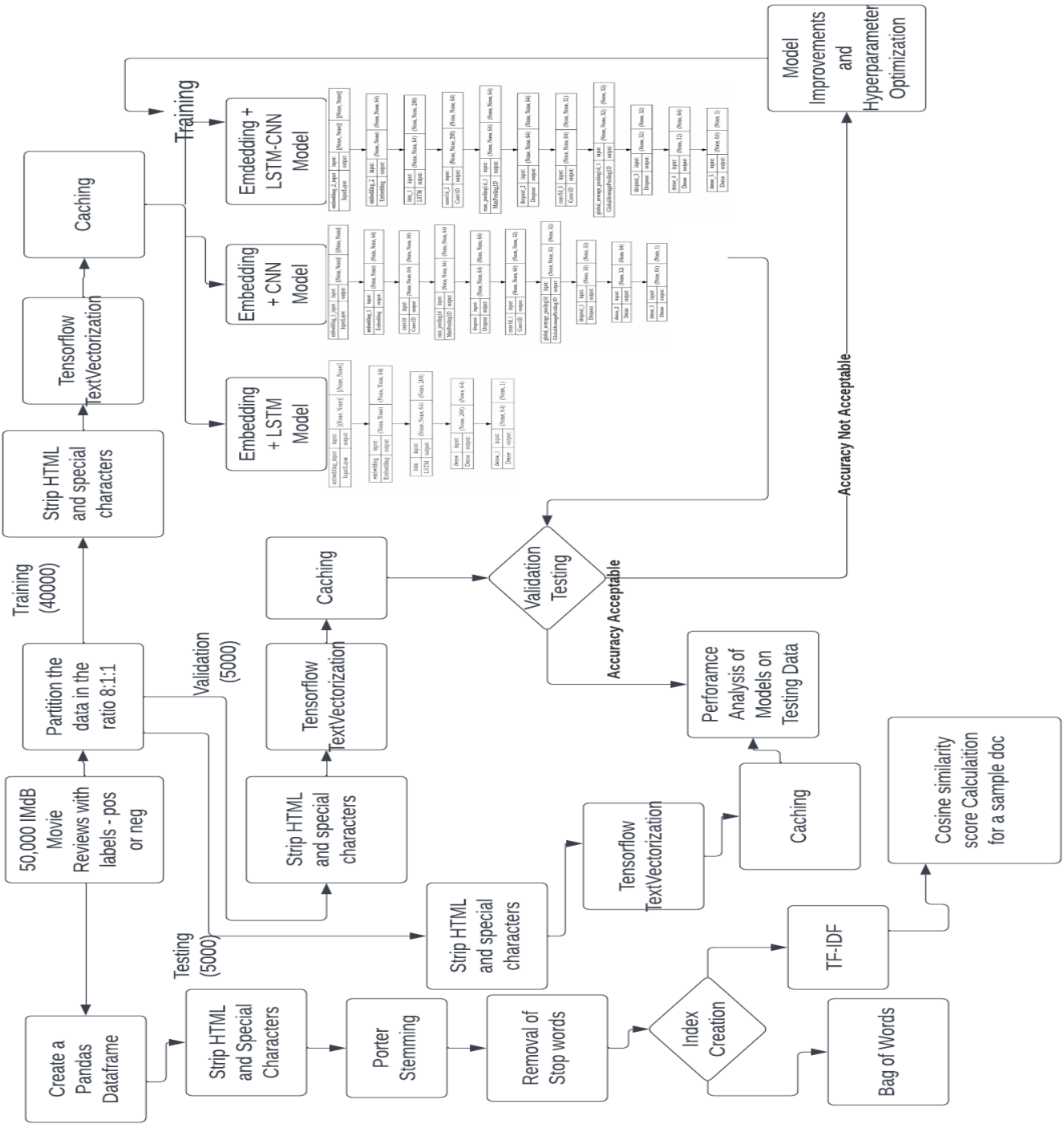


Figure 5.1: Block diagram of our implementation.

For the index creation and text representation task, the following steps have been undertaken: (1) create a `pandas` dataframe with all 50,000 reviews (2) ***strip the***

HTML and special characters (like punctuation) (3) perform normalization with ***Porter stemming*** (4) prepare a list of ***English stopwords*** using `nltk` and eliminate the stop words in the text.

Then either (A) prepare a ***Bag of words model*** using `CountVectorizer` which counts the occurrences of each term in the document. Or (B) prepare a ***tf-idf representation*** using `sklearn`'s `TfidfVectorizer`. If tf-idf is used, compute the ***pairwise cosine similarity*** of any randomly chosen document with all other documents ($cosSim(d_1, d_2) = \frac{d_1 \bullet d_2}{||d_1||_2 ||d_2||_2}$). List the first five documents when ***ranked in decreasing order of similarity*** as the closest five matches.

For the sentiment analysis task, the steps followed are (1) ***partition*** the dataset of 50,000 movie reviews with corresponding labels - *positive* or *negative* - into three parts - *training, validation,* and *testing* - in the ratio 80:10:10. (2) ***strip the HTML and the punctuation*** (3) perform ***text vectorization*** with *Keras* `TextVectorization` (4) ***cache the datasets for faster computation*** (avoids repeated disk seeks, which can be the bottleneck in the training and testing process) (5) ***train the 3 models*** - LSTM, CNN, and LSTM-CNN - for epochs specified in [3] (6) ***tune and improve based on accuracy of validation partition*** (7) ***evaluate the performances of the 3 models on testing partition*** based on the evaluation metrics listed in Section 6.

5.2 Dataset and Preprocessing

Following the methodology in [3], the dataset used in our implementation is the celebrated IMDb dataset with 50,000 polar movie reviews along with their corresponding sentiment labels - *positive* or *negative*. The dataset is publicly available on Stanford University's website (please refer to *Dataset Used* Section at the end of the report for more details). The reviews are divided into two folders 'train' and 'test', 25000 each, which in turn have two subfolders each 'pos' and 'neg' for storing the reviews of the corresponding categories.

For the present implementation, however, the dataset has been re-partitioned in the ratio 80:10:10 into training, validation or development, and testing sets respectively.

As a first data cleaning and preprocessing step, the HTML tags and punctuation have been removed from the reviews. The dataset has then been vectorized to vectors of length 250 with a vocabulary of 10,000 words using the Keras `TextVectorization` module (example shown in Figure 5.2).

```
Review: tf.Tensor(b"While exploring some caves with his wife, a doctor is bitten by a bat which causes some
Label: neg
Vectorized review: (<tf.Tensor: shape=(1, 250), dtype=int64, numpy=
array([[ 134, 5229,   46,    1,   15,   24,  324,    4,  886,    7, 7061,
          32,    4, 3267,   62, 2885,   46,    1,  502,  288, 1855,  972,
         836,    3,   46,  530,  152,  488,  326, 1311,  288,   89,  564,
          11,  947,  190,  501,   35,  108,    4, 5216,  793,    5,    1,
           2, 3267,   83, 2072,    9,  459,   32,  312, 2377,   35,   29,
        5054,   65,  176,    1,    1,    3, 4751,    8, 1710,    2,  112,
         109,  181,    6,  138, 1681,   72,    3,    2,   17,  109, 1525,
          33, 1670,   16,   48,  543,   39,   54,    4, 2337, 1207,   16,
           9,   31,    2,  175,    7,  961, 1539,    8,   63,  350,  127,
          10,  191,    2,   19,   46,  768,   16,   29, 2399,  746,  595,
           3,  323, 1326, 1766,    2, 1311,  160,    5,    2,  541, 3503,
        9271,    7,  179,   49,   99,   18,    9,  149,   75,   72,    5,
           4, 4487,  132,    4, 1025, 1417,   16,  245,  392,   28,    5,
           2, 3315,    1, 1413,   44,   47, 1091,   44,    5,    0,    0,
           0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
           0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
           0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
           0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
           0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
           0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
           0,    0,    0,    0,    0,    0,    0,    0]])>, <tf.Tensor: shape=(), dtype=int32, numpy=0>)
```

Figure 5.2: A text vector of length 250 prepared with a vocabulary size of 10,000 for a negative review.

5.3 Caching and Prefetching: Better Execution Efficiency

To ensure that the I/O operations (file reads in the present case) do not become blocking, caching and prefetching have been used. The `tf.data.Dataset.cache` transformation caches a dataset (keeps data in memory when it is loaded off the disk), either in memory or on local storage. This saves some operations like file opening and data reading from being executed during each epoch (Figure 5.3). The files are opened and read only during the first epoch; the subsequent epochs just reuse the cached data, saving disk seeks.

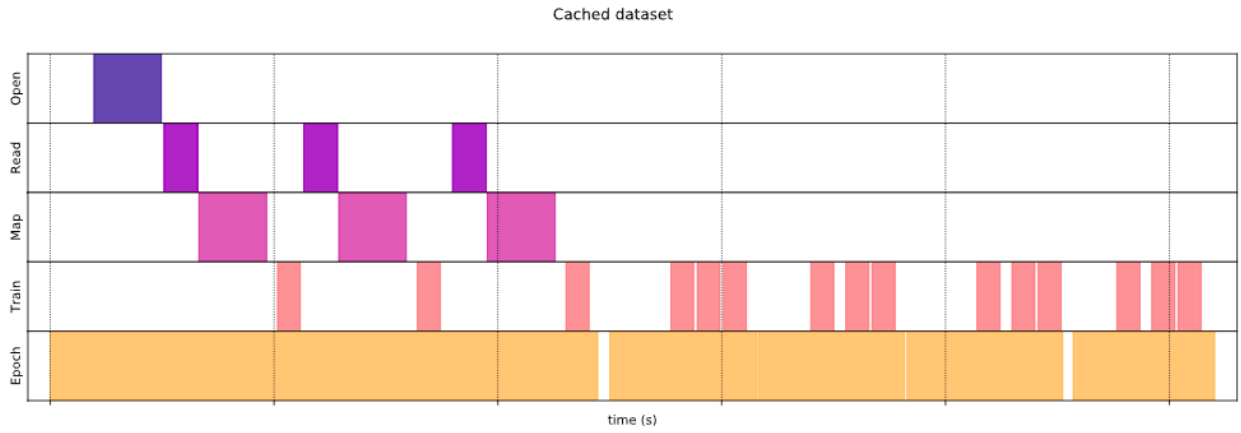


Figure 5.3: Caching saves file opening and reading operations during all but the first epoch as depicted in the ops versus time graph.
Image Credit: Tensorflow.org

Prefetching, as implemented through `.prefetch()`, overlaps data preprocessing and model execution while training. When the model is executed training step s, the input pipeline reads the data for the next step s+1 (Figure 5.4). Such an overlap reduces the overall sum of training and loading time.

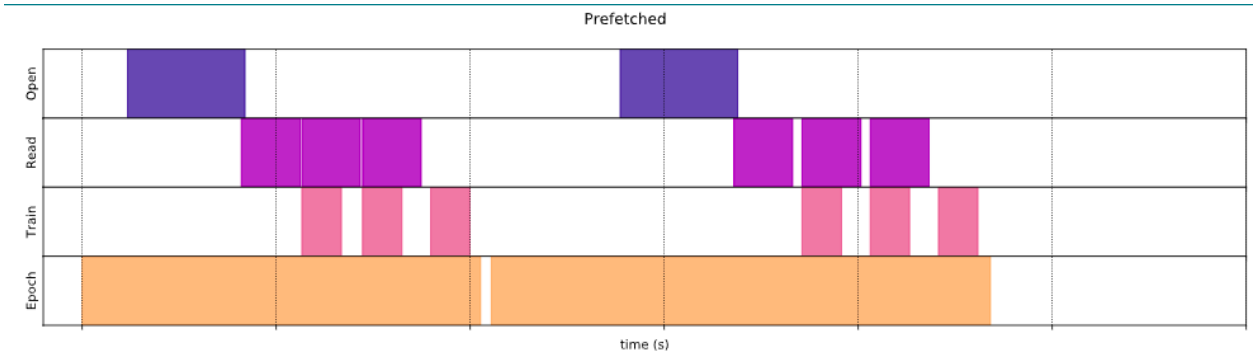


Figure 5.3: Prefetching overlaps operations for faster execution as depicted in the ops versus time graph.
Image Credit: Tensorflow.org

5.4 The Classification Models

Since the goal of [3] was a performance analysis of neural network based architecture for sentiment analysis of movie reviews, our present implementation takes up the general architecture proposed in the papers along with some overfitting and performance optimization mechanisms. Three models - LSTM,

CNN, and LSTM-CNN (depicted in Table 5.1) - have been used for the binary classification task. Each of three has a Word Embedding layer as the first layer.

Word Embeddings: They present a way to use an efficient, dense representation in which similar words have a similar encoding. An embedding is a dense vector of floating point values, which are trained in the embedding layer. The Keras `Embedding` layer can be understood as a lookup table that maps integer indices, standing for particular words, to dense floating-point embedding vectors (their embeddings) [18]. Similar words are gathered together in the resultant n-dimensional space. As per [3], this has been prepended to each of the three models as a first layer.

LSTM based model: LSTM is an improvised version of RNN which supports long term dependency. LSTMs possess a temporal loop, wherein the memory cell stores or updates the information based on the provided input sentences [3]. Due to its temporal support, LSTMs (much like several other sequence models) have been widely used in NLP (Section 3). Our implementation contains an LSTM layer with 200 units, followed by a relu-activated fully connected layer and an unactivated final layer of one neuron.

***CNN based model*:** [3] proposes the use of two convolutional layers for extracting features from the embedding vectors, followed by pooling layers to provide translation invariance. Our implementation incorporates two convolutional 1D layers, followed by `MaxPooling1D` and `GlobalAveragePooling1D` layers respectively. Dropout layers have been used (with dropout probability of 0.2) to prevent overfitting. Furthermore, the convolutional layers have been equipped with L2-regularization ($\lambda = 10^{-4}$) and He-initialization (conformant to relu-activation). The output of convolutional and pooling layers are fed into a fully connected layer, which in turns feed the extracted features into a hidden layer for classification [3]. The fully connected layer has also been powered with L2-regularization and He-initialization.

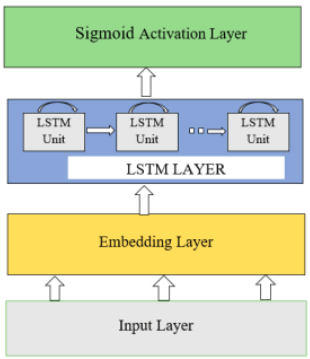
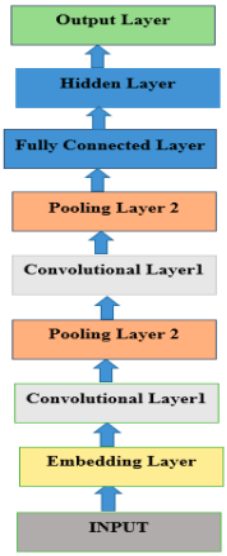
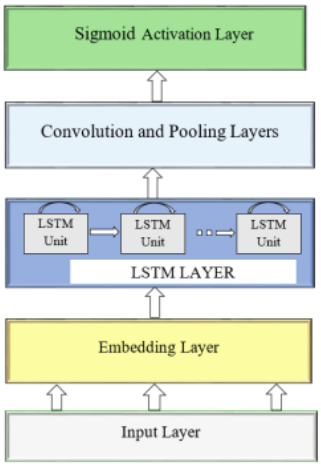
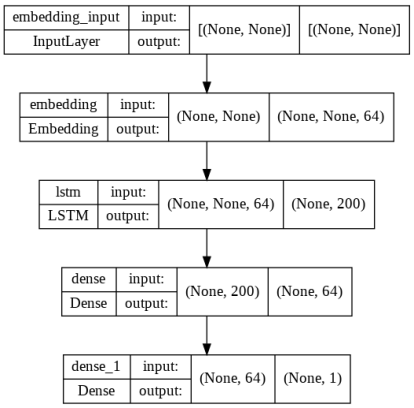
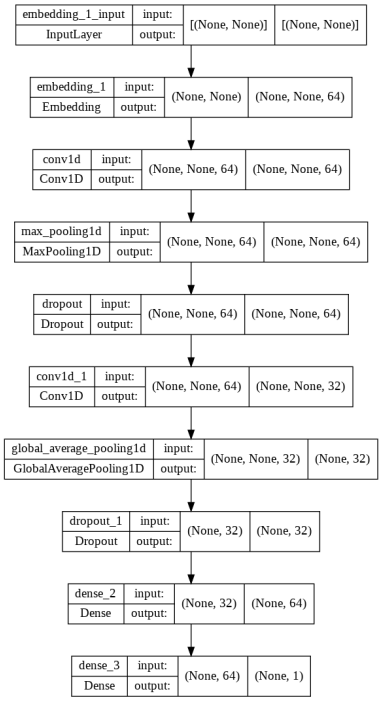
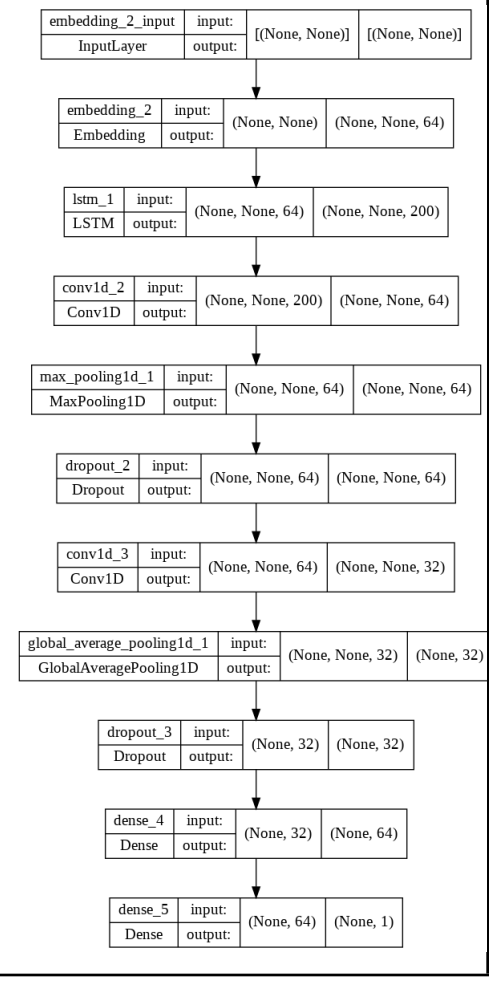
Model	LSTM	CNN	LSTM-CNN
Architecture Proposed in [3]			
Our Implement-ation			

Table 5.1: The proposed models in [3] and our corresponding implementations.

LSTM-CNN based model: As a final candidate model for sentiment classification, the LSTM-CNN model has been proposed in [3] by combining the aforementioned models - LSTM and CNN - in order. The LSTM unit is understood to analyze the syntax structure of a sentence, while the convolutional layer extracts features like the positive and negative catchphrases [3]. A 200-unit LSTM layer has been inserted between the embedding and the first convolutional layers in the CNN-based model to implement the LSTM-CNN model.

As for the optimizer, Adam optimizer has been used with the loss function - `BinaryCrossEntropy`, which is suitable for binary classification tasks where the labels are not one-hot encoded; the learning rate set to $\alpha = 10^{-4}$.

5.5 Bonus Task: Bag-of-Words and tf-idf

As a bonus task, we developed a pipeline to obtain the Bag-of-words and tf-idf representations for our collection of 50,000 IMDb movie reviews. The tf-idf representation, thus obtained, is used to demonstrate the search for closest matches for a sample document using the cosine similarity scores.

To ease the process of data cleaning and normalization, the reviews were used to fill a `pandas dataframe` as opposed to continuing with the individual files. Hence, `BeautifulSoup`'s HTML parser was used to strip the HTML tags -
, for instance - in the text. All special characters were eliminated using a regular expression for non-alphanumeric characters - `[^a-zA-Z0-9]`.

The corpus was hence put through stemming using the Porter's Stemming Algorithm, which is known to be empirically very effective [4]. Here, stemming refers to the heuristic process of chopping off the ends of the words in the hope of reducing the various inflectional forms of the words to a common base form [4]. The stopwords were then eliminated using the list of English stopwords, as available through `NLTK`.

To prepare the Bag-of-words model, `sklearn.feature_extraction.text.CountVectorizer` has been used, which counts the occurrences of each of the words in the vocabulary in each

document to create the corresponding cell entry in the BOW table. It outputs the representation in a special sparse vector format, owing to the very sparse nature of BOW representation for most NLP text data.
Similarly, the tf-idf representation has been prepared with `sklearn.feature_extraction.text.TfidfVectorizer`. The term frequency values have been calculated without logarithmic weighting - that is, n (natural), according to the SMART notation, has been used. The inverse document frequency, on the other hand, has been used with logarithmic weighting, that is, $idf = log \frac{N}{df_t}$ or t (idf) in the SMART notation. Again, the documents are represented as sparse vectors.

The `sklearn.metrics.pairwise.linear_kernel` module has been used for calculating pairwise cosine similarities of a chosen document with all the documents in the collection. The closest five matches, in terms of similarity, are then presented by ranking the documents in descending order of cosine similarity scores thus calculated.

6. Evaluation Metrics

Following the evaluation scheme proposed in [3], we use the following quantitative metrics to evaluate the performances of our three classifiers - LSTM, CNN, and LSTM-CNN.

Accuracy: It can be defined as the number of correct predictions as a fraction of total number of predictions, that is, $Accuracy = \frac{\#\ of\ correct\ predictions}{total\ \#\ of\ predictions}$ [12]. Here, a correct prediction would be a review assigned its correct sentiment, be it *positive* or *negative*, as the case might be.

Precision: It determines the fraction of instances that actually turn out to be *positive* in the group the classifier has declared as a *positive* class [12], that is, $Precision = \frac{TP}{TP+FP}$.

Recall: It measures the fraction of positive examples correctly predicted by the classifier [12], that is, $Recall = \frac{TP}{TP+FN}$.

Specificity: It measures the fraction of negative examples correctly predicted by the classifier, that is, $Recall = \frac{TN}{TN+FP}$.

F1-score: It has been designed to achieve a tradeoff between precision and recall, defined as $F_1 = \frac{2P \cdot R}{P+R}$.

***ROC Curve*:** The receiver operator characteristics curve plots the true positive rate against the false positive rate. A good classification model should be located as close to the upper left corner of the diagram as possible, while a model that makes random guesses should reside along the main diagonal (unit slope).

Area under curve (AUC): The area under the ROC curve can be a test for the quality of the classifier. In general, the classifiers can be graded according to Table 6.1.

AUC	Model Quality
Less than 0.5	No Discrimination
Between 0.5 and 0.7	Poor Discrimination
Between 0.7 and 0.8	Acceptable Discrimination
Between 0.8 and 0.9	Excellent Discrimination
More than 0.9	Outstanding discrimination

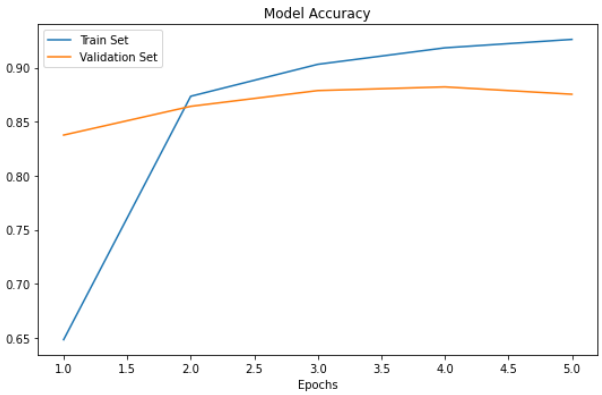
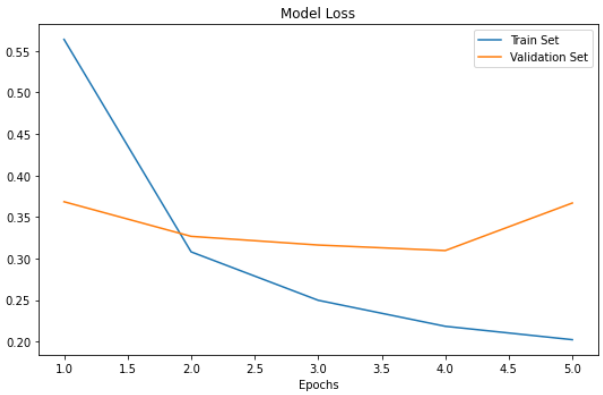
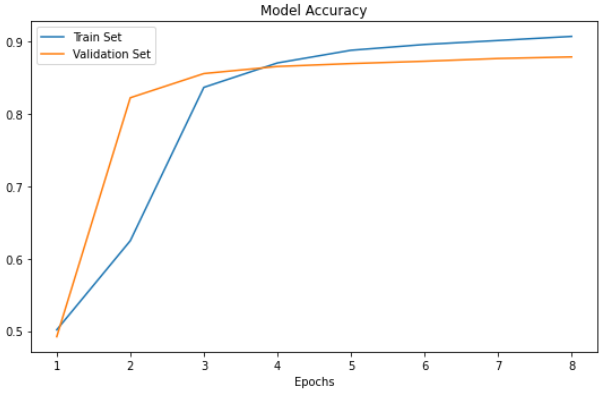
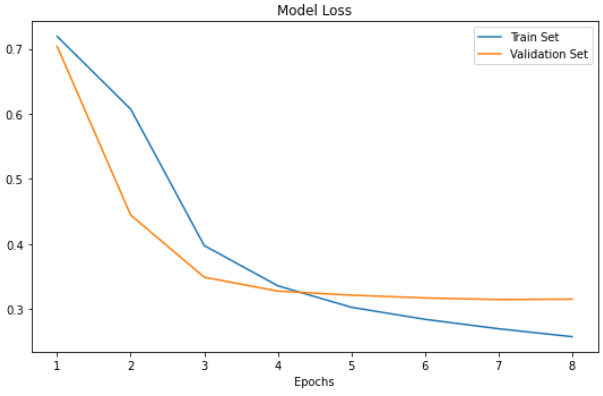
Table 6.1: The AUC Reference Chart.

7. Results

The present section discusses the results, as evaluated per the metrics presented in Section 6, for the three models in our implementation and also draws comparisons with the results obtained in [3].

7.1 Loss and Accuracy Plots

During the training of the three models - 5, 8, and 6 epochs respectively for LSTM, CNN, and LSTM-CNN models as per [3] - the training and validation accuracies and `BinaryCrossEntropy` losses have been plotted as given in Table 7.1. Note that following the guidelines in [4], we have not used the testing partition for this plotting process, and have restricted ourselves to the training and development partitions only. The testing partition is used in Section 7.2 once all the model improvements and hyperparameter values are finalized.

Model	Accuracy Plot (training and validation)	Loss Plot (training and validation)
LSTM		
CNN		

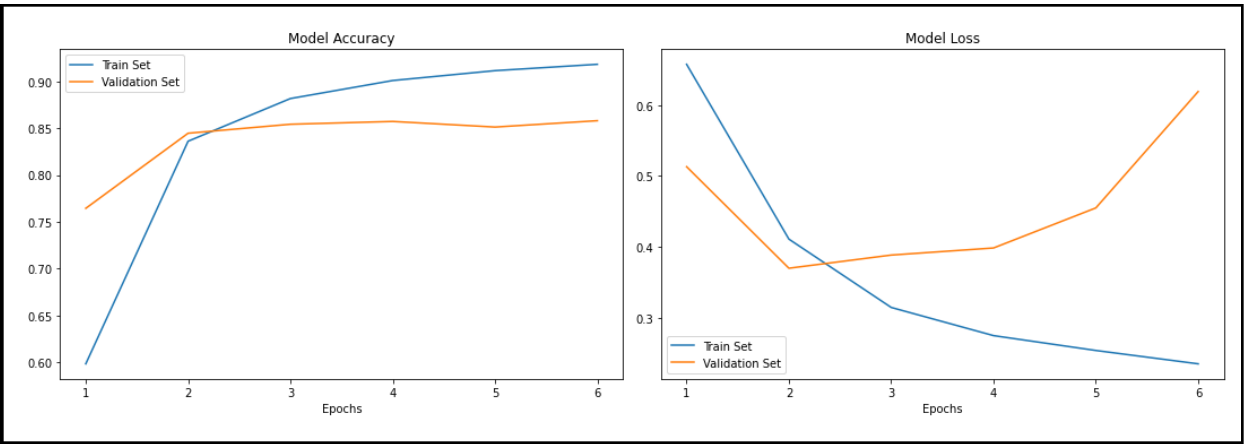
LSTM-CNN	

Table 7.1: The model accuracies and model loss for training (blue) and validation (red) sets with epochs for the three models.

It may be noted that LSTM based models show signs of slight overfit since the traditional mechanism to cope with overfitting - dropout layers and regularization, for instance - do not work as well for LSTMs as they do for CNNs. A better solution would be to reduce the number of LSTM units in the layer from 200 to 100 (say); such a change, however, would be a great alteration to the methodology proposed in [3] and hence has been avoided for a more fair comparison. The high model complexity of the LSTM-CNN might also be a contributing factor to the observed overfit in the same.

Further, it may be noted that the loss plots are monotonically decreasing for the training set, as is desirable. The loss plots are not optimized for the Validation set by Adam optimizer; hence, they are bound to show fluctuations. The validation accuracy plot, on the other hand, is overall increasing for validation sets, thereby greenlighting the models for the performance analysis stage, as discussed in the subsequent subsections.

7.2 Test Accuracy

Table 7.2 shows the training, validation and testing accuracies for the three models as per [3] and for our implementation. It is observed that the CNN model outperforms the other models in terms of testing accuracy. This may be attributed to the fact that syntax - usually captured well by RNN based models - is not as important as the catchphrase features - usually captured well for CNN based models - for sentiment analysis [3].

The LSTM-CNN model, on the other hand, shows high performance on the training partition due to its higher model complexity. The underfit, in effect, is minimum for it among the three models; such an optimization, however, comes at the cost of generalization error, as depicted through its relatively poor score on the testing partition.

Model	LSTM in [3]	CNN in [3]	LSTM-CNN [3]	Our LSTM	Our CNN	Our LSTM CNN
Training Accuracy	N/A*	N/A*	N/A*	0.93	0.91	0.92
Validation Accuracy	N/A*	N/A*	N/A*	0.88	0.88	0.86
Testing Accuracy	0.88	0.90	0.89	0.88	0.89	0.86

*Not reported in the paper [3]

Table 7.2: The training, validation, and testing accuracies for the three models.

It may be noted that all three of the models outperform the Random Forest, SVM, and Logistic Regression models for the same task (which are restricted to accuracy of upto 0.87) in related literature [3].

7.3 Other Evaluation Metrics

Other than accuracy, the precision, recall, specificity, and F1-score values have also been calculated for the three models for a more holistic comparison. All these metrics have been computed for the testing partition. Again, CNN outperforms the other models in F1-score (which combines both precision and recall), which may be explained as per Section 7.2.

Model	LSTM in [3]	CNN in [3]	LSTM-CNN [3]	Our LSTM	Our CNN	Our LSTM CNN
Precision	0.90	0.87	0.87	0.84	0.90	0.86

Recall	0.82	0.95	0.90	0.92	0.88	0.88
Specificity	0.84	0.90	0.87	0.82	0.90	0.87
F1-score	0.86	0.91	0.88	0.88	0.89	0.85

Table 7.3 Precision, recall, specificity, and F1-score for the 3 models on testing data.

To help better visualization of the above results, Table 7.4 shows the confusion matrices obtained on the testing data (5000 reviews) with three models.

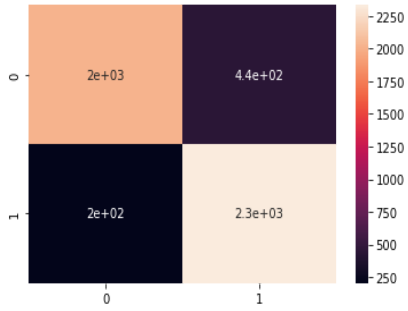
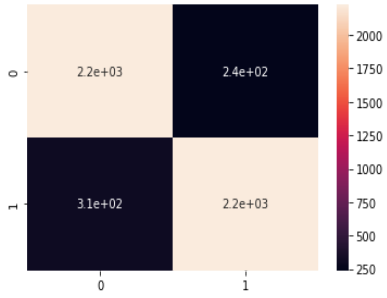
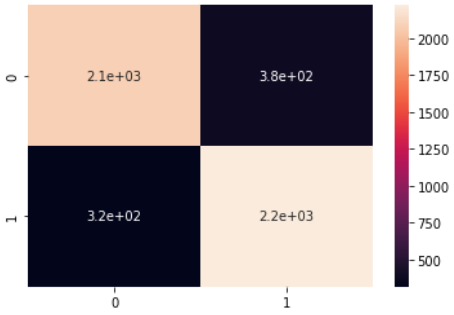
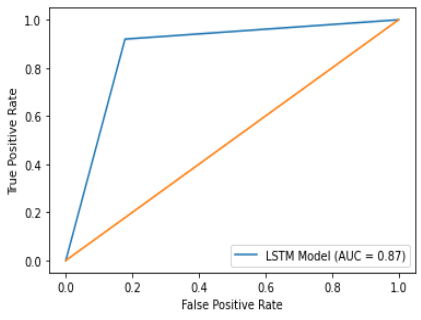
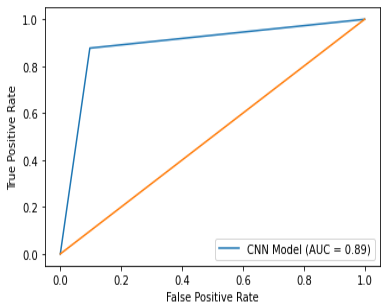
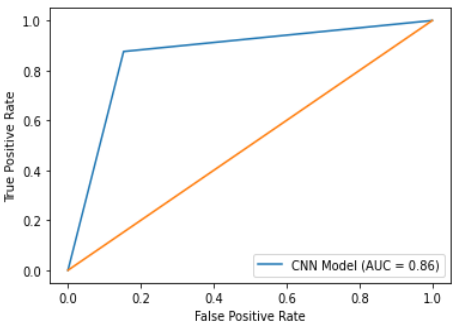
Model	LSTM	CNN	LSTM-CNN
Confusion Matrix			

Table 7.4: The confusion matrices for the three models for the 5000 testing cases.

7.4 ROC Curves and AUC

Table 7.5 shows the ROC curves obtained for the three models obtained in our implementation, along with our AUC scores and the AUC scores reported in [3].

Model	LSTM	CNN	LSTM-CNN
ROC Curves			

Our AUC	0.87	0.89	0.86
AUC in [3]	0.89	0.87	0.85

Table 7.5: ROC curves and the AUC scores for the three models on the testing data.

Using the criteria discussed in Table 6.1 for AUC scores, each of the models is judged to have an "*Excellent Discrimination"* power.

7.5 Deploying the Models with GUI

A StreamLit based GUI app has been created for deploying the three models to classify a user input review as *positive* or *negative.* The application uses the models saved after training on the IMDb movie reviews with the performance metrics mentioned in the preceding sections. Each of the three models has been used for classification separately.

The following are some screenshots demonstrating the deployment of the GUI for our models. Figure 7.6 shows the input interface for entering a textual movie review. Figure 7.7 shows the classification of a review with a *positive* sentiment; corresponding results were obtained for *negative* reviews.

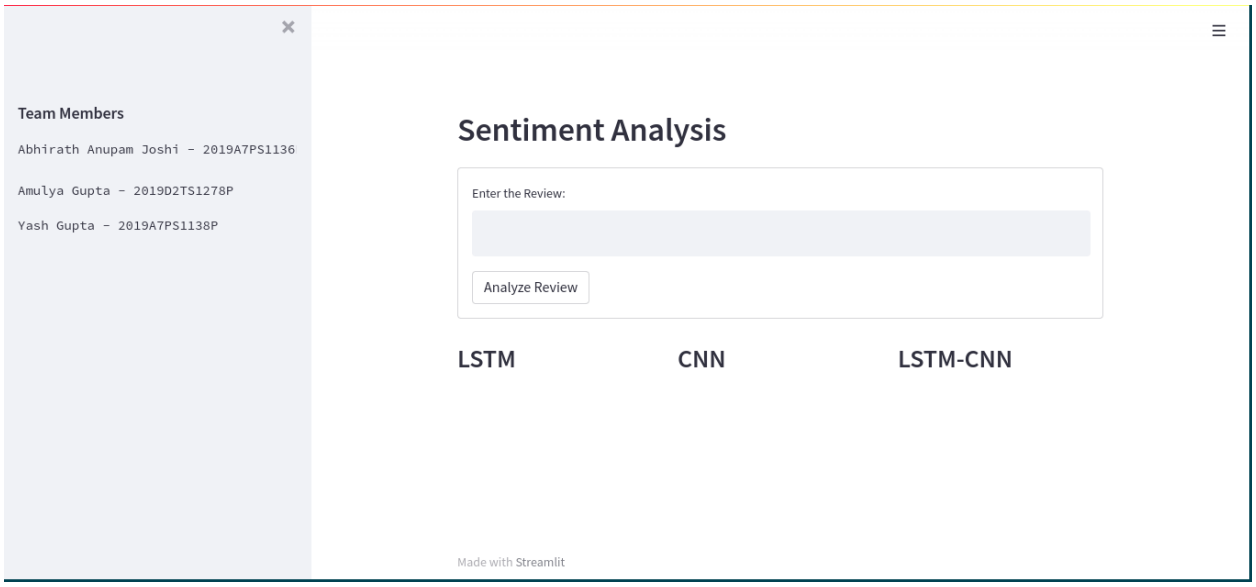


Figure 7.6: The GUI to enter a textual movie review. Type the review in the box and press "Analyze Review".

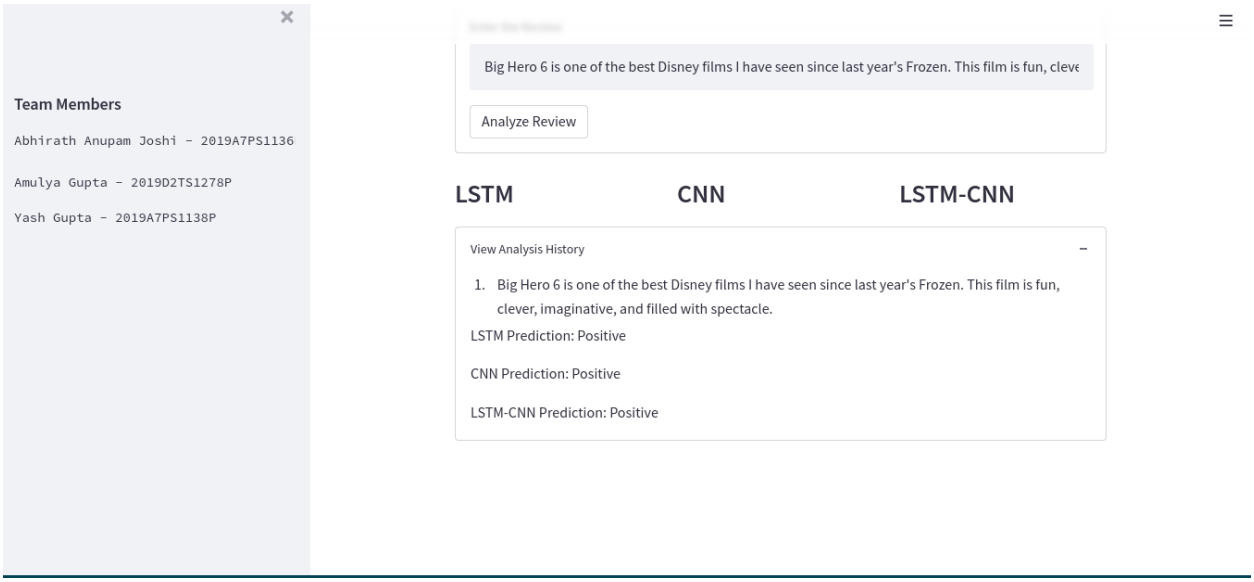


Figure 7.7: Press “View Analysis History” to view the results of classification of the review by each of the three models trained and deployed.

8. Conclusion and Future Work

As discussed in Section 7, the CNN based model outperforms the LSTM and LSTM-CNN on majority of the evaluation metrics. [3] attributes it to the relatively higher importance of features like words and phrases conveying strong or clear-cut sentiments as opposed to the syntactic structure of the text. Regardless, all three models provide significant improvement over the more traditional MLTs for the same task [3], justifying the growing adoption of deep learning based techniques in the domains of data science and machine learning.

The authors of [3], however, neglect the problem of overfitting. The present report incorporates certain mechanisms to cope with overfitting - including regularization and dropout - especially for the convolutional layers. Further, it has been felt that the LSTM models would perform better on the validation and testing partitions with early stopping and lesser number of memory cells.

The authors of [3] also did not include any mechanisms to reduce the training time. With caching and prefetching mechanism proposed in the present report, the

training times have been brought to an average of $600\ \frac{s}{epoch}$, $45\ \frac{s}{epoch}$, and $700\ \frac{s}{epoch}$ for the LSTM, CNN, and LSTM-CNN models respectively.

Till now, the focus has been on performance on previously unseen testing data which belongs to the same distribution as the training data. However, it would be desirable to be able to deploy models trained on movie reviews with little change on a website dealing with reviews on amusement parks or restaurants, say. Such data would obviously belong to a similar yet not the same distribution. Thus, a future line of work would be to take up this problem of out-of-distribution generalization, already prevalent in computer vision [19], and develop robust models for the same.

The RNN based model in [3] uses a LSTM layer where the flow of information is in one direction. A different strategy to try would be to use a bidirectional RNN which exploits the flow of information in both directions across the layer. Similarly, the use of Word2Vec instead of Word Embedding layer in the network would be an interesting experiment to take up.

Finally, a comprehensive ablation study of the three models discussed and similar models would be highly beneficial to the NLP research community to understand the exact role of each layer in the network and subsequently use them as building blocks for better networks.

References

[1] M. Somers, "Emotion AI, explained," *MIT Sloan*, 08-Mar-2019. [Online]. Available: https://mitsloan.mit.edu/ideas-made-to-matter/emotion-ai-explained. [Accessed: 21-Apr-2022].

[2] "Sentiment analysis," *Wikipedia*, 15-Apr-2022. [Online]. Available: https://en.wikipedia.org/wiki/Sentiment_analysis. [Accessed: 21-Apr-2022].

[3] M. R. Haque, S. Akter Lima and S. Z. Mishu, "Performance Analysis of Different Neural Networks for Sentiment Analysis on IMDb Movie Reviews," 2019 3rd International Conference on Electrical, Computer & Telecommunication Engineering (ICECTE), 2019, pp. 161-164, doi: 10.1109/ICECTE48615.2019.9303573.

[4] Manning, Christopher D., Raghavan, P., and Schutze, H. (2008). Introduction to Information Retrieval. Cambridge Press. ISBN: 978-1-107-66639-9.

[5] "Imdb_reviews : tensorflow datasets," *TensorFlow*. [Online]. Available: https://www.tensorflow.org/datasets/catalog/imdb_reviews. [Accessed: 21-Apr-2022].

[6] E. Cambria, S. Poria, A. Gelbukh and M. Thelwall, "Sentiment Analysis Is a Big Suitcase," in IEEE Intelligent Systems, vol. 32, no. 6, pp. 74-80, November/December 2017, doi: 10.1109/MIS.2017.4531228.

[7] Z. Jianqiang, G. Xiaolin and Z. Xuejun, "Deep Convolution Neural Networks for Twitter Sentiment Analysis," in IEEE Access, vol. 6, pp. 23253-23260, 2018, doi: 10.1109/ACCESS.2017.2776930.

[8] N. Majumder, S. Poria, H. Peng, N. Chhaya, E. Cambria and A. Gelbukh, "Sentiment and Sarcasm Classification With Multitask Learning," in IEEE Intelligent Systems, vol. 34, no. 3, pp. 38-43, 1 May-June 2019, doi: 10.1109/MIS.2019.2904691.

[9] Duyu Tang, Bing Qin, Xiaocheng Feng, and Ting Liu. 2016. Effective LSTMs for Target-Dependent Sentiment Classification. In Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 3298–3307, Osaka, Japan. The COLING 2016 Organizing Committee.

[10] S. M. Qaisar, "Sentiment Analysis of IMDb Movie Reviews Using Long Short-Term Memory," 2020 2nd International Conference on Computer and Information Sciences (ICCIS), 2020, pp. 1-4, doi: 10.1109/ICCIS49240.2020.9257657.

[11] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. NeRF: Representing scenes as neural radiance fields for view synthesis. In ECCV, 2020.

[12] Tan, P.-N., Steinbach, M.,, Kumar, V. (2005). Introduction to Data Mining. Addison Wesley. ISBN: 0321321367.

[13] X. Ouyang, P. Zhou, C. H. Li and L. Liu, "Sentiment Analysis Using Convolutional Neural Network," *2015 IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing*, 2015,

[14] Yazhi Gao, W. Rong, Y. Shen and Z. Xiong, "Convolutional Neural Network based sentiment analysis using Adaboost combination," *2016 International Joint Conference on Neural Networks (IJCNN)*, 2016

[15] Shin, Bonggun, Timothy Lee, and Jinho D. Choi. "Lexicon integrated CNN models with attention for sentiment analysis." *arXiv preprint arXiv:1610.06272* (2016).

[16] G. Xu, Y. Meng, X. Qiu, Z. Yu and X. Wu, "Sentiment Analysis of Comment Texts Based on BiLSTM," in IEEE Access, vol. 7, pp. 51522-51532, 2019, doi: 10.1109/ACCESS.2019.2909919.

[17] H. Watanabe, M. Bouazizi and T. Ohtsuki, "Hate Speech on Twitter: A Pragmatic Approach to Collect Hateful and Offensive Expressions and Perform Hate Speech Detection," in IEEE Access, vol. 6, pp. 13825-13835, 2018, doi: 10.1109/ACCESS.2018.2806394.

[18] "Word embeddings : text : tensorflow," *TensorFlow*. [Online]. Available: https://www.tensorflow.org/text/guide/word_embeddings. [Accessed: 26-Apr-2022].

[19] Spandan Madan, Timothy Henry, Jamell Dozier, Helen Ho, Nishchal Bhandari, Tomotake Sasaki, Frédo Durand, Hanspeter Pfister, Xavier Boix. "When and how CNNs generalize to out-of-distribution category-viewpoint combinations." *arXiv 2007.08302 (2020).*

Dataset Used

IMDb Movie Reviews: The dataset containing 50,000 movie reviews, along with corresponding labels (positive or negative), has been linked to the colab environment directly from https://ai.stanford.edu/~amaas/data/sentiment/aclImdb_v1.tar.gz.