Exploring Machine Learning and Few Shot Techniques for Text Classification

Aman Parikh amparikh@ucsd.edu
Sakshi Subedi sasubedi@ucsd.edu
Kunj Champaneri kchampaneri@ucsd.edu

1 Introduction

Text classification has surged due to the demand for digital data and extracting valuable information from text. It has applications in spam detection, sentiment analysis, and topic classification. Our focus is exploring text classification by categorizing documents into classes. This problem is important and we aim to identify the most effective approach, including the few-shot technique, to enhance accuracy. Our initiative focuses on few-shot learning methods for text classification, which allow accurate predictions with limited labeled data. This is valuable when collecting data for individual classes is challenging. Our goal is to enhance the precision and efficiency of text categorization, enabling scalability and adaptability to new categories with minimal training data. Ultimately, we aim to advance text classification and develop more powerful models for practical applications.

2 What you proposed vs. what you accomplished

Our project aimed to investigate various text classification techniques, including machine learning, deep learning algorithms, and few-shot techniques. Our primary objective was to understand the impact of different factors such as the amount of data and the nature of labels (binary or multi-class) on choosing the most suitable technique for text classification. We collected datasets, performed data preprocessing, and built models to explore the differences and effectiveness of these techniques. We have accomplished what we proposed.

3 Related work

1. In [3] Emphasizing text preprocessing's significance in text classification efficiency, they highlight key steps like tokenization. The preprocessing stage is critical and complex, converting raw textual data into a structured representation to identify distinguishing text features.

2. In [5] *SetFit* is a few-shot fine-tuning framework for Sentence Transformers that achieves remarkable accuracy with fewer parameters. It involves a two-step training process using a contrastive, Siamese approach. It is a simple and efficient method for few-shot learning.

3. In [2] presented how we can incorporate machine learning techniques for text classification. The paper describes the process of text classification and various machine learning algorithms that have been proposed for text classification, including support vector machine (SVM).

4. In [4] the paper introduces how to fine-tune BERT models for classification in order to understand its performance on a variety of datasets. Thus this paper will help us understand and implement the different steps involved in fine-tuning.

5. [1] This paper introduces a CNN architecture for text classification, demonstrating its effectiveness. We compare CNN's performance to LSTM on different datasets to assess their comparative capabilities.

4 Your Dataset

4.1 Data Summary and Statistics

4.1.1 IMDB Dataset

We have taken the IMDB dataset from Kaggle. This dataset has 50,000 rows and has 2 columns:

1. Review: Text column which contains the reviews of movies.

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

Figure 1: IMDB dataset

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50000 entries, 0 to 49999
Data columns (total 2 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   review     50000 non-null  object
 1   sentiment  50000 non-null  int64
dtypes: int64(1), object(1)
```

Figure 2: Overview of IMDB data

2. Sentiment: Contains ”positive” or ”negative” sentiments towards the movie. (Binary labels).

The following are the statistics of the data:

1. Figure 1 and 2 provide an overview of the *IMDB* dataset, indicating that there are no missing or null rows present in the dataset.

2. As depicted in Figure 3, both the positive and negative classes exhibit a well-balanced distribution of data, with 25,000 data each. This balanced distribution contributes to accurate predictions.

Example The following is an example of both classes.

1. **Positive**: If you like original gut-wrenching laughter you will like this movie. If you are young or old then you will love this movie, hell even my mom liked it.$< br/ >< br/ > GreatCamp$!!!

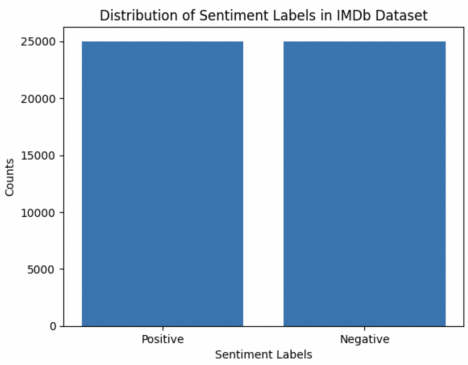

Figure 3: Bar plot of IMDB Sentiment Labels

	Class Index	Title	Description
0	3	Wall St. Bears Claw Back Into the Black (Reuters)	Reuters - Short-sellers, Wall Street's dwindli...
1	3	Carlyle Looks Toward Commercial Aerospace (Reu...	Reuters - Private investment firm Carlyle Grou...
2	3	Oil and Economy Cloud Stocks' Outlook (Reuters)	Reuters - Soaring crude prices plus worries\ab...
3	3	Iraq Halts Oil Exports from Main Southern Pipe...	Reuters - Authorities have halted oil export\f...
4	3	Oil prices soar to all-time record, posing new...	AFP - Tearaway world oil prices, toppling reco...

Figure 4: Head of AG News train dataset

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 120000 entries, 0 to 119999
Data columns (total 3 columns):
 #   Column       Non-Null Count   Dtype
---  ------       --------------   -----
 0   Class Index  120000 non-null  int64
 1   Title        120000 non-null  object
 2   Description  120000 non-null  object
dtypes: int64(1), object(2)
memory usage: 2.7+ MB
```

Figure 5: Overview of AG News train dataset

2. **Negative**: The screen-play is very bad, but there are some action sequences that i really liked. I think the image is good, better than other romanian movies. I liked also how the actors did their jobs.

Task
The task is to predict the sentiment of the user based on the review given by them.

4.1.2 AG News Dataset

We have also utilized the AG News Classification Dataset obtained from Kaggle, which comprises multi-class labels. The dataset includes a train file with $120,000$ rows and a test file with $7,600$ rows. Each row in the dataset has three columns. They are:

1. Title: Contains the title of the news.

2. Description: Contains new's description.

3. Class Index: The dataset consists of news articles classified into four categories: *World*, *Sports*, *Business*, and *Science/Technology*. Each category contains $30,000$ articles, making it a balanced dataset.

The following are the statistics of the data:

1. Figure 4 and 5 provide an overview of the *AG News* train dataset, indicating that there are no missing or null rows present in the training and testing dataset.

2. As depicted in figure 6, all the classes (World, Sports, Business, and Science/Technology) exhibit a well-balanced distribution of data with $30,000$ data each. This balanced distribution contributes to accurate predictions.

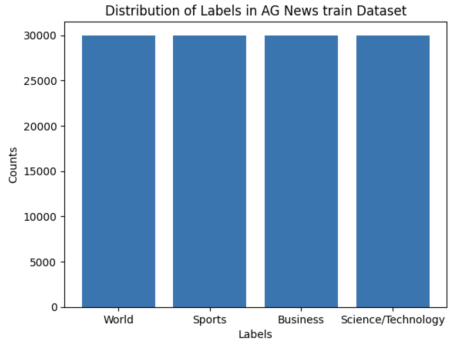

Figure 6: Bar plot of AG News Labels

Example The following is an example of all four classes.

1. **World**:Walking link to low dementia riskWalking may protect the elderly from developing dementia, research suggests.

2. **Sports**:Wisdom was main courseWALTHAM – He is an 87-year-old man with a cane and a cigar, and the clout of a king.

3. **Business**:Nepal blockade b́low to tourismŃepal tour operators say tourists cancelled millions of dollars of bookings due to the rebel blockade of Kathmandu.

4. **Science/Technology**:This week in merger newsThis week saw three merger deals worth about $60 billion–including one that ranks as the largest software merger in history.

Task
The task of this dataset is to classify news articles into their respective categories.

4.2 Data preprocessing

We have performed the following data preprocessing steps:

1. **Label Mapping**: Sentiment labels in the IMDB dataset have been transformed into numeric values (0 and 1) for easier processing and analysis.

2. **Concatenating text :** In the AG News dataset, the Title and Description columns have been concatenated into a single text column to enable analysis.

3. **Elimination of stopwords :** Stopwords, common insignificant words in English, have been removed from the review columns in the IMDB dataset and the text column in the AG News dataset using the NLTK library.

4. **Using regular expressions :** Additional transformations using regular expressions have been applied to the data, including word replacements and expansion of abbreviations. Only alphanumeric characters have been retained, creating a standardized dataset.

5. **Tokenization :** The "Review" column in IMDB and the merged text column in AG News has been tokenized, breaking the text into individual words or tokens. A vocabulary of unique words has been created based on a predetermined threshold, improving tracking and comprehension of specific words in the dataset

4.3 Challenges with Dataset

1. Both datasets have a large corpus of words, which can be resource-intensive and time-consuming to process.

2. Our analysis revealed that certain data points in both datasets exhibited complex semantic patterns that could lead to inaccurate predictions. Specific examples are provided in the error analysis.

5 Baselines

5.1 IMDB Dataset

The *K-nearest neighbors (KNN)* algorithm serves as a baseline model for the *IMDB* dataset. No hyperparameter tuning was performed as default values were used. The sentiment labels were transformed into binary values: 1 for positive labels and 0 for negative labels. The dataset was loaded from local storage in CSV format and randomly divided into a training set, comprising 80% of the data, and a test set, containing the remaining 20% of the data. The train set is used to fit a TF-IDF vectorizer, which is then used to transform both the train and test set into TF-IDF feature vectors. The baseline model achieved an accuracy rate of 76.9%. It took approximately $15ms$ to train the model. The implementation code can be found here

5.2 AG News Dataset

The *K-nearest neighbors (KNN)* algorithm is used as a baseline model for the AG News dataset as well. No hyperparameter tuning was conducted,

and default parameter values were employed. The train and test CSV files were merged and randomly divided into a training set, which accounts for 80% of the data, and a test set consisting of the remaining 20%. The train set is used to fit a TF-IDF vectorizer, which is then used to transform both the train and test set into TF-IDF feature vectors. The baseline model achieved an accuracy rate of 89.36%. It took approximately $18ms$ to train the model. The implementation code can be found here

6 Models

6.1 Machine Learning Models

6.1.1 IMDB Dataset

We applied several machine learning algorithms, such as *Logistic Regression*, *Naive Bayes*, *Support Vector Machine* to the IMDB dataset.

IMPLEMENTATION: We performed similar implementation as we performed in our baseline *KNN* model. The code are Logistic Regression, Naive Bayes, and SVM.

RUNTIME: Below are the approximate runtime achieved by each model.

Logistic Regression	Naive Bayes	SVM
9ms	20ms	12ms

RESULTS: Below are the accuracy scores achieved by each model.

Logistic Regression	Naive Bayes	SVM
90.02%	86.35%	90.95%

All of these models performed better than the baseline *KNN* model.

6.1.2 AG News Dataset

Similarly, we applied several machine learning algorithms, such as *Logistic Regression*, *Naive Bayes*, *Support Vector Machine* to the AG News dataset.

IMPLEMENTATION: We performed similar implementation as we performed in our baseline *KNN* model. The code are Logistic Regression, Naive Bayes, and Support Vector Machine.

RUNTIME: Below are the approximate runtime achieved by each model.

Logistic Regression	Naive Bayes	SVM
37ms	16ms	16ms

RESULTS: Below are the accuracy scores achieved by each model.

Logistic Regression	Naive Bayes	SVM
91.37%	90.21%	92.22%

All of these models performed better than the baseline *KNN* model.

6.2 Deep Learning Architecture

6.2.1 IMDB Dataset

1. **LSTM Model**

   MODEL ARCHITECTURE: The model starts with an embedding layer that converts input word indices into dense word embeddings. This layer maps each word index to a continuous vector representation of size $embed\_size$. These embeddings are learned from pre-trained *GLOVE* embeddings. Next, there is an *LSTM* layer that processes the word embeddings. It takes the embedding vectors of size $embed\_size$ as input and produces output tensors of size $num\_hiddens$. The *LSTM* layer is bidirectional, meaning it processes the input sequence in both forward and backward directions. Following the *LSTM* layer, there is a linear layer that maps the concatenated output of the *LSTM* layer to a single output value. The input size of this linear layer is $4 * num\_hiddens$ because the *LSTM* layer is bidirectional, and the outputs from both directions are concatenated. Finally, the output of the linear layer is passed through a sigmoid activation function to squash the values between 0 and 1.

   HYPERPARAMETER TUNING: We conducted experiments by varying the learning rate, number of epochs, and batch size, and selected the configuration that yielded the best outcomes. Additionally, we explored different dropout values to assess any potential overfitting or underfitting issues. However, in our case, dropout did not prove to be beneficial, so we proceeded without using it.

   RUNTIME: Each epoch of the model's execution required approximately 30 seconds.

   RESULTS:The model achieves an accuracy of **87.73%**, which represents a substantial enhancement compared to our *KNN* baseline model. The loss vs accuracy curve can be viewed in 7

   Learning Rate : **0.01**, N_epochs = **7**

2. **TextCNN Model**

   MODEL ARCHITECTURE: The architecture consists of two embedding layers where the first layer $self.embedding$ is trainable and the second embedding

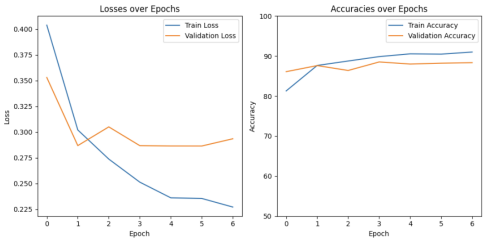

Figure 7: Loss Vs Accuracy Curve of LSTM Model on IMDB dataset

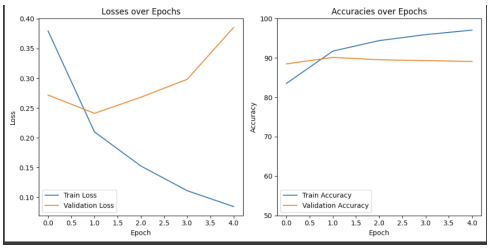

Figure 8: Loss Vs Accuracy Curve of TextCNN Model on IMDB dataset

layer $self.constant\_embedding$ is non-trainable. Similar to the *LSTM* model, the *GloVe* embeddings are employed. Multiple one-dimensional convolutional layers are defined using $self.convs$, which is an $nn.ModuleList()$. Each convolutional layer has a specific kernel size determined by $kernel\_sizes$ and a specific number of output channels specified by $num\_channels$. Adaptive average pooling $self.pool$ is applied to reduce each channel to a single value. The pooled features are concatenated and passed through a linear layer $self.decoder$, which maps the concatenated features to a single output value. The output of the linear layer is then passed through a sigmoid activation function to constrain the value between 0 and 1, representing the probability of the input belonging to a particular label. To prevent overfitting, dropout with a rate of 0.5 is applied to the concatenated encoding.

HYPERPARAMETER TUNING: We conducted experiments with various values for the learning rate, number of epochs, batch size, kernel sizes, and the number of channels. After evaluating their performance, we selected the configuration that yielded the most favorable outcomes. Additionally, we experimented with different dropout values to address potential overfitting, and as a mitigation measure, we applied a dropout value of 0.5.

RUNTIME Each epoch of the model's execution required approximately 34 seconds.

RESULTS: The model achieves an accuracy of **89.54%**, which represents a substantial improvement compared to our *KNN* baseline model. Furthermore, while the improvement over the *LSTM* model is not significant, it is noteworthy that a *CNN*, typically associated with image applications, has the potential to outperform *RNN/LSTM* architectures in natural language tasks. The loss vs accuracy curve can be viewed in 8.

LR = 0.001, **Num Epochs = 5**
Embed Size = 100, **Kernel Sizes = [3, 4, 5]**, **Nums Channels = [100, 100, 100]**

The code implementation of *LSTM* and *TextCNN* can be found here

6.2.2 AG News Dataset

1. **LSTM Model**:
   MODEL ARCHITECTURE:
   The modified architecture is designed for multi-class classification. It includes an embedding layer initialized with a pre-trained GloVe embedding matrix. The LSTM layer takes the embedded input tokens and produces outputs for each token. The last output state of the LSTM sequence is extracted. A fully connected layer is used to obtain logits, representing the predicted scores for each class. This architecture accommodates multi-class classification by utilizing pre-trained embeddings, LSTM layers, and fully connected layers. It enables the model to process input sequences, capture temporal information, and generate logits for multi-class classification tasks.
   HYPERPARAMETER TUNING: We conducted experiments by varying the learning rate, number of epochs, and batch size, number of channels and selected the configuration that yielded the best outcomes. Additionally, we explored different dropout values to assess any potential overfitting or underfitting issues. However, in our case, dropout did not prove to be beneficial, so we proceeded without using it.
   RUNTIME: Each epoch of the model's execution required approximately 26 seconds.
   RESULTS: The model achieves an accuracy of **91.47%**, which represents a substantial en-

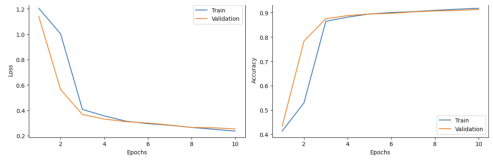

Figure 9: Loss Vs Accuracy Curve of LSTM Model on AG News dataset

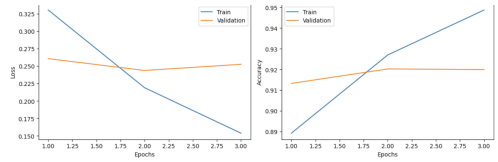

Figure 10: Loss Vs Accuracy Curve of TextCNN Model on AG News dataset

hancement compared to our *KNN* baseline model. The loss vs accuracy curve can be viewed in 9.
LR = 0.001, **Num Epochs = 5**

2. **TextCNN Model**:
MODEL ARCHITECTURE:
The model architecture consists of an embedding layer that maps input tokens to dense embeddings. These embeddings are then processed through convolutional blocks, each consisting of a 2D convolutional layer with specified kernel sizes and output channels. The resulting convolutional outputs are pooled by selecting maximum values along the last dimension. The pooled outputs are concatenated, and the concatenated tensor is passed through a fully connected layer. Dropout regularization is applied with a rate of 0.5, and a ReLU activation function is used. In the forward pass, the input tokens are embedded, convolutions are performed, and the resulting pooled outputs are concatenated. Dropout is applied before obtaining the final output. This model architecture enables the model to learn representations from the input tokens and capture relevant features for the desired output dimension.
HYPERPARAMETER TUNING: We conducted experiments with various values for the learning rate, number of epochs, batch size, kernel sizes, and number of channels. After evaluating their performance, we selected the configuration that yielded the most favorable outcomes. Additionally, we experimented with different dropout values to address potential overfitting and ultimately chose a dropout rate of 0.5 as it provided the best results.
RUNTIME Each epoch of the model's execution required approximately 99 seconds.
RESULTS: The model achieves an accuracy of **91.83%**, which represents a substantial enhancement compared to our *KNN* baseline model. It's worth noting that this accuracy surpasses the performance of our implemented *LSTM* model as well. The loss vs accuracy curve can be viewed in 10.
LR = 0.001, **Num Epochs = 3**

The code implementation of *LSTM* and *TextCNN* can be found here

6.3 Few Shot Technique

SetFit is a prompt-free framework [6] for efficient few-shot fine-tuning of Sentence Transformers (ST). Its goal is high accuracy with minimal labeled data. SetFit learns to represent sentences as dense vectors through training on a large text corpus. It fine-tunes a Sentence Transformer on a small labeled dataset and trains a classifier on the embeddings using contrastive loss. Positive pairs share the same class, while negative pairs differ. SetFit is effective for text classification, excelling in few-shot learning and accurate models on small datasets. Its advantages include faster training and inference without the need for large-scale models.

SetFit outperforms BERT in few-shot text classification due to its smaller size and faster training speed. SetFit, as a sentence transformer model, excels in adapting to new tasks with limited data, while BERT's large size and training on a large dataset make it slower and less suitable for few-shot learning. SetFit is also more robust to noise, thanks to being trained on a large corpus of text, while BERT is more sensitive to noise due to training on a smaller corpus. Therefore, SetFit is the preferred choice for few-shot text classification tasks compared to BERT. Also, as mentioned in our proposal, we performed a few-shot technique using *SetFit* and *BERT* models only on *IMDB* dataset.

6.3.1 Few-Shot Technique using BERT Model

MODEL ARCHITECTURE: The code utilizes a small sample size of only 100 examples to train and test the *BERT* model. The pre-trained BERT model and tokenizer are imported from the transformers library. The reviews are processed and

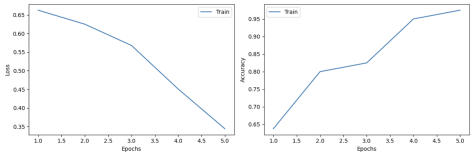

Figure 11: Loss Vs Accuracy Curve of Few-Shot BERT Model on IMDB dataset

converted into input IDs and attention masks using the tokenizer's *batch_encode_plus* method. These tokenized data and corresponding labels are then transformed into PyTorch tensors and organized into TensorDataset objects. The logistic regression model is trained using a training loop that iterates over batches from the train dataloader. During training, backpropagation is applied to update the model's parameters using the *AdamW* optimizer with *optimizer.step()*. Once training is completed, the model is evaluated on the test set using a testing loop. The implementation code can be found here

HYPERPARAMETER TUNING: We conducted experiments with various values for the number of epochs and batch size. After evaluating their performance, we selected the configuration that yielded the most favorable outcomes.

RUNTIME: How long did it take to train your model? Each epoch of the model's execution required approximately 9 seconds.

RESULTS: The model achieves an accuracy of **80%**, which represents a substantial enhancement compared to our *KNN* baseline model. The loss vs accuracy curve can be viewed in 11.

batch size = 16, **num epochs = 5**

6.3.2 Few-Shot Technique using SetFit Model

MODEL ARCHITECTURE: The code snippet trains and tests the *SetFit* model using a small sample size of 100 examples. The model used is *paraphrase-mpnet-base-v2* from the Sentence Transformers library. The trainer, called *SetFitTrainer* is instantiated with specific parameters and begins the fine-tuning process. Once training is complete, the model's performance is evaluated on the testing dataset. The implementation code can be found here

HYPERPARAMETER TUNING: We conducted experiments with various values for the number of epochs, batch size, learning rate, and number of iterations. After evaluating their performance, we selected the configuration that yielded the most favorable outcomes.

RUNTIME: How long did it take to train your model? One epoch required approximately 723 seconds of the model's execution.

RESULTS: The model achieves an accuracy of **98%**, which represents a substantial enhancement compared to our *KNN* baseline model. It's worth noting that this accuracy surpasses the performance of our implemented *few-shot* technique using *BERT* model as well.

batch size = 8, **mum epochs = 1**
learning rate = 0.00002
mum iterations = 40

The following are common for all models:

WORKING IMPLEMENTATION: We have successfully achieved a functioning implementation.

OTHER PEOPLE'S CODE: We referred to code snippets from the research papers we read as well as ChatGPT for minor assistance.

COMPUTE: We utilized our personal machines as well as Google Collab for computing our tasks.

REFERENCE: Although we considered various pre-processing steps to improve the model's efficiency, we did not adhere strictly to a single notebook for this purpose. Additionally, for the model architecture, we drew inspiration from different sources and experimented with multiple models, ultimately choosing the one that yielded the highest accuracy.

7 Error analysis

We encountered two examples in which our baseline models and best-performing SVM model failed:

For *IMDB* dataset

- The screen-play is very bad, but there are some action sequences that i really liked. I think the image is good, better than other romanian movies. I liked also how the actors did their jobs.

- Rip off of "Scream" or especially "I know what you did last summer", there's some entertainment here, and a little scary, but they needed some originality.¡br /¿¡br /¿An entertainment score? 6.5/10 Overall? 5.5/10

For *AG News* dataset

- A Distraction as a Deadline ApproachesAs the holidays approach, Christmas spirit is a bargain-hunting essential.

- FCC Mulls Airborne Mobile Phone Use Although there may have been technical limitations at the time the cell phone ban was established, according to IDC 39;s Shiv Bakhshi, it is unclear why the ban has remained in place, given that

The examples from the IMDB dataset and AG News dataset share commonalities. In terms of semantics, both involve expressing opinions or discussing specific topics. For IMDB, it's about movies and aspects like screenplay, action sequences, image quality, actors' performances, originality, and rating scores. In the AG News dataset, the topics are the significance of Christmas spirit during the holidays and the FCC considering mobile phone use on airplanes. Syntactically, both examples utilize various linguistic elements. They include introductory phrases for context, complex sentence structures with multiple clauses, and the use of conjunctions to connect ideas or introduce contrasting viewpoints. The IMDB examples also use a combination of positive and negative language to express opinions, while the AG News examples focus on providing informative content.

Some of the annotated examples we took for the *IMDB* dataset which our *Logistic Regression* model failed to predict are:

- "Excellent writing and wild cast. The tech is poor but it's obviously very low budget. Looks like they didn't cut the negative but had to release on a video output. In any case one of the most inventive comedies I've seen lately. The screenwriter in particular is fine." with *positive* sentiment

- "Trekkies is not truly a film that explores the world of Star Trek fandom. Instead, it presents a spectacle focusing on those Star Trek fans who lack a connection to reality. As a spectacle, it may be entertaining. However, it would be misguided to believe that this movie provides genuine insight into the lives of Star Trek fans. In fact, many Star Trek enthusiasts find themselves cringing at the portrayal depicted in this film." with *negative* sentiment

The model assigned a predicted sentiment label as *negative* and *positive* for the above examples.

Some of the annotated examples we took for the *AG News* dataset which our *Logistic Regression* model failed to predict are:

- "Mounting evidence suggests that gene therapy has the potential to cure children who are born with a condition that impairs their innate immune system, effectively restoring their ability to defend against infections." with *Class Index* as 1 denoting as *World* article.

- "The charges for broadband internet access are expected to decrease as a result of Ofcom's mandate for British Telecom to reduce the fees it levies on internet service providers.", with *Class Index* as 3 denoting as *Business* article.

The model assigned a predicted label of 4, indicating that the article falls under the category of "Science".

8 Contributions of group members

List what each member of the group contributed to this project here. For example:

- Aman: Data pre-processing, built and trained deep learning models.

- Sakshi: Error Analysis, Built and trained machine learning models

- Kunj: Error Analysis, Built and trained few shot models.

9 Conclusion

Surprisingly, simpler machine learning models like logistic regression showed comparable or even superior accuracy to complex deep learning models, challenging the assumption of their superiority. Contrary to initial assumptions, Convolutional Neural Network (CNN) models are equally effective for natural language tasks as RNN models like LSTM, not just limited to computer vision. Few Shot learning demonstrated the potential of accurate predictions with limited training samples. Sentence Transformer, with fewer parameters, outperformed BERT, showcasing its efficiency in natural language processing.

10 Future Work

We plan to create a CLDNN model by combining CNN and LSTM, leveraging their respective strengths in temporal and spatial understanding for

improved performance in natural language tasks. Additionally, we will delve deep into few-shot learning for text classification, exploring concepts like induction networks and XLNet to understand their mechanisms and potential applications in this domain.

References

[1] Alexis Conneau, Holger Schwenk, Loïc Barrault, and Yann Lecun. Very deep convolutional networks for text classification. *arXiv preprint arXiv:1606.01781*, 2016.

[2] M Ikonomakis, Sotiris Kotsiantis, and Vasilis Tampakas. Text classification using machine learning techniques. *WSEAS transactions on computers*, 4(8):966–974, 2005.

[3] Ammar Ismael Kadhim. An evaluation of preprocessing techniques for text classification. *International Journal of Computer Science and Information Security (IJCSIS)*, 16(6):22–32, 2018.

[4] Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. How to fine-tune bert for text classification? In *Chinese Computational Linguistics: 18th China National Conference, CCL 2019, Kunming, China, October 18–20, 2019, Proceedings 18*, pages 194–206. Springer, 2019.

[5] Lewis Tunstall, Nils Reimers, Unso Eun Seo Jo, Luke Bates, Daniel Korat, Moshe Wasserblat, and Oren Pereg. Efficient few-shot learning without prompts. *arXiv preprint arXiv:2209.11055*, 2022.

[6] Moshe Wasserblat. Sentence transformer fine-tuning: Setfit outperforms gpt-3 on few-shot text classification. *Towards Data Science*, 2021.