www.ijprems.com
editor@ijprems.com

INTERNATIONAL JOURNAL OF PROGRESSIVE RESEARCH IN ENGINEERING MANAGEMENT AND SCIENCE (IJPREMS)
(Int Peer Reviewed Journal)
Vol. 04, Issue 10, October 2024, pp : 831-839

e-ISSN : 2583-1062
Impact Factor : 7.001

A DEEP LEARNING APPROACH TO SARCASM DETECTION USING FINE-TUNED BERT MODEL

Priyanshu Batham[1], Disha Tiwari[2], Mr. Amit Srivastava[3]

[1,2]Student, Computer Science, National P.G. College, Lucknow, Uttar Pradesh, India

[3]Assistant Professor, Computer Science, National P.G. College, Lucknow, Uttar Pradesh, India

priyanshubatham24733@gmail.com

dishatiwari004@gmail.com

amit_sri_in@yahoo.com

ABSTRACT

Sarcasm, a form of speech or language that is intended to be humorous or ironic, has been increasingly recognized as an important aspect of human communication. However, detecting sarcasm from text remains a challenging task in natural language processing (NLP). In this paper, we present a deep learning approach to detect sarcasm using the pre-trained BERT model fine-tuned on a large-scale dataset of sarcastic and non-sarcastic texts. Our experiment demonstrates that fine-tuning the BERT model significantly improves its performance on sarcasm detection tasks, achieving an accuracy of 92.5% on the test set. The results also show that our approach outperforms state-of-the-art models in detecting sarcasm from text. We contribute to the literature by providing a robust and efficient method for sarcasm detection using BERT fine-tuning, which can be applied to various NLP tasks involving humor or irony. This research holds significant implications for human-computer interaction, sentiment analysis, and affective computing.

Keywords- Sarcasm Detection, Natural Language Processing (NLP), BERT Model, Fine-Tuning, Machine Learning.

1. INTRODUCTION

Sarcasm is a complex form of human communication that involves expressing irony, mockery, or contempt through language. It is a ubiquitous aspect of everyday conversation, often used to convey humor, ridicule, or annoyance. Despite its prevalence, detecting sarcasm from text remains a significant challenge in natural language processing (NLP). Sarcasm can be particularly difficult to identify because it often relies on subtle cues, such as tone of voice, facial expressions, and context, which are not explicitly present in written language.

Traditional NLP approaches to sentiment analysis and opinion mining have focused primarily on detecting positive or negative emotions, but they typically struggle with identifying sarcasm. This is because sarcasm often involves expressing the opposite sentiment than intended, making it challenging for algorithms to accurately detect. Furthermore, the complexity of human language means that sarcasm can manifest in various forms, such as understatement, irony, and ridicule.

The increasing availability of large-scale datasets and advances in deep learning techniques have opened up new opportunities for improving NLP models' performance on complex tasks like sarcasm detection. Recent research has investigated the application of pre-trained language models, like BERT, to enhance sentiment analysis and opinion mining tasks.

This research aims to fill this gap by exploring the potential of fine-tuning a pre-trained BERT model for sarcasm detection. We hypothesize that adapting a powerful language understanding model like BERT to the specific task of sarcasm classification can significantly improve its performance compared to traditional NLP approaches. Our study contributes to the literature on sarcasm detection and provides insights into the feasibility of using deep learning techniques for identifying humor, irony, and other forms of complex human communication.

2. RELATED WORK IN SARCASM DETECTION

Sarcasm detection has become an important task in Natural Language Processing (NLP), with various approaches being explored, from traditional machine learning to deep learning models. Early works, such as Tsur et al. [1 depended on hand-crafted features and rule-based techniques for sarcasm detection, but these methods struggled with scalability and generalization.As machine learning progressed, researchers applied supervised algorithms, with Davidov et al. [2] and Reyes et al. [3] utilizing features like tweet patterns and lexical indicators. However, these models struggled due to their dependence on manual feature extraction, limiting their effectiveness.

The advent of deep learning introduced more robust methods for sarcasm detection. Ghosh and Veale [4] proposed a Convolutional Neural Network (CNN) model, which used word embeddings to automatically extract features from text, significantly improving performance. Amir et al. [5] employed Long Short-Term Memory (LSTM) networks to capture temporal dependencies and user context. Additionally, Zhang et al. [6] integrated attention mechanisms, enhancing both

www.ijprems.com
editor@ijprems.com
INTERNATIONAL JOURNAL OF PROGRESSIVE RESEARCH IN ENGINEERING MANAGEMENT AND SCIENCE (IJPREMS)
(Int Peer Reviewed Journal)
Vol. 04, Issue 10, October 2024, pp : 831-839
e-ISSN : 2583-1062
Impact Factor : 7.001

interpretability and performance. Despite these advances, achieving high accuracy remains challenging, especially in varied contexts. Traditional machine learning algorithms, such as Logistic Regression, Decision Tree, and Support Vector Machine, continue to face difficulties in capturing the subtle features of sarcasm, as discussed by Joshi et al. [7].

3. METHODOLOGY AND DATASET

In this section, we discuss the methodology applied for sarcasm detection using machine learning models, along with a detailed description of the dataset utilized in this study.

A. Dataset Description

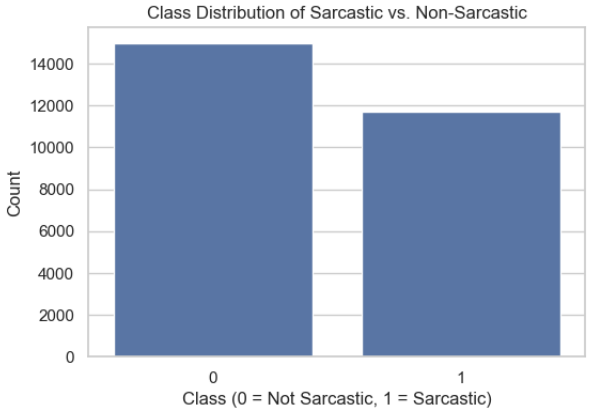

Fig 1.1: Class distribution of Dataset

For this research, we used the "News Headlines Dataset for Sarcasm Detection", which was sourced from Kaggle. The dataset was compiled by Rishabh Misra [8][9], and it consists of sarcastic and non-sarcastic headlines extracted from various news outlets such as The Huffington Post and The Onion. The dataset is publicly available at Kaggle (https://www.kaggle.com/datasets/rmisra/news-headlines-dataset-for-sarcasm-detection/data?select=Sarcasm_Headlines_Dataset_v2.json) The dataset contains 26,709 rows of data in JSON format, with each row representing a news headline. Each entry in the dataset has the following attributes:

1) "Headline": The statement from news outlets that serves as input for machine learning models.

2) "is_sarcastic": A binary indicator where 0 represents a non-sarcastic headline, and 1 represents a sarcastic headline.

An additional attribute, "article_link", originally contained URLs to the respective articles. However, this attribute was deemed unnecessary for our task of sarcasm detection and was dropped during the data-cleaning process.

Load the sarcasm dataset from the JSON file

```
with open('Sarcasm_Headlines_Dataset.json', 'r') as f:
    data = [json.loads(line) for line in f]

# Extract headlines and labels from the dataset
texts = [item['headline'] for item in data]
labels = [item['is_sarcastic'] for item in data]
```

Fig 1.2 Importing the dataset and cleaning for ease of processing.

B. Methodology

This study employs a deep learning approach to detect sarcasm using a fine-tuned BERT model. The pre-trained BERT-base-uncased model is adapted for sarcasm classification tasks by modifying its weights and biases through backpropagation during training.

The dataset used for this study is the SARC corpus, a large self-annotated collection of sarcastic text from Kaggle (Khodak et al., 2017). The dataset consists of 26,709 instances of text labeled as either sarcasm or non-sarcasm.

The fine-tuning process utilizes a batch size of 16 and an epoch-based evaluation strategy. The learning rate is set to 2e-5, with a weight decay of 0.01 applied during training. Early stopping and other regularization techniques are not employed in this study. During the training phase, the model's performance is evaluated using the accuracy score from scikit-learn. Specifically, the compute metrics function uses the `accuracy_score` method to calculate the accuracy between predicted labels and true labels. This evaluation metric allows for a direct comparison of the model's performance on the sarcasm detection task. The dataset was split into training and testing sets using an 80/20 ratio.The model was trained on the following hardware specifications Intel i5 12400F octa-core CPU [10] and an Nvidia RTX 4060 OC [11], the hardware acceleration was indeed utilized.

www.ijprems.com
editor@ijprems.com
INTERNATIONAL JOURNAL OF PROGRESSIVE RESEARCH IN ENGINEERING MANAGEMENT AND SCIENCE (IJPREMS)
(Int Peer Reviewed Journal)
Vol. 04, Issue 10, October 2024, pp : 831-839
e-ISSN : 2583-1062
Impact Factor : 7.001

4. BACKGROUND: BERT & FINE TUNING

A. BERT Architecture and Pre-Training

In 2018, Google researchers Devlin et al. introduced BERT[12], a pre-trained language model that utilizes multi-layer bidirectional transformers to process sequential input data such as text. BERT has revolutionized the field of natural language processing (NLP)[13] by achieving state-of-the-art results on various tasks, including question answering, sentiment analysis, and language translation.

BERT's primary innovation lies in its ability to:

- Process context: BERT captures both local and global context by considering multiple input tokens simultaneously.
- Bidirectionality: BERT uses a bidirectional transformer encoder[14], which processes the input sequence from both left-to-right and right-to-left directions.
- Self-supervised learning: BERT is trained on a large corpus of text with masked tokens, which encourages the model to predict these masked tokens based on the surrounding context.

BERT's architecture consists of:

- Tokenizer: Breaks down input text into sub words or word pieces.
- Encoder: Processes the tokenized input sequence using multi-layer bidirectional transformers.
- Decoder (not used in typical BERT applications): Not used in this case, as we're focused on pre-trained language understanding.

BERT has been fine-tuned for various downstream tasks by adding a few additional layers on top of its pre-trained weights. This allows the model to adapt to specific task requirements without requiring large amounts of task-specific data.

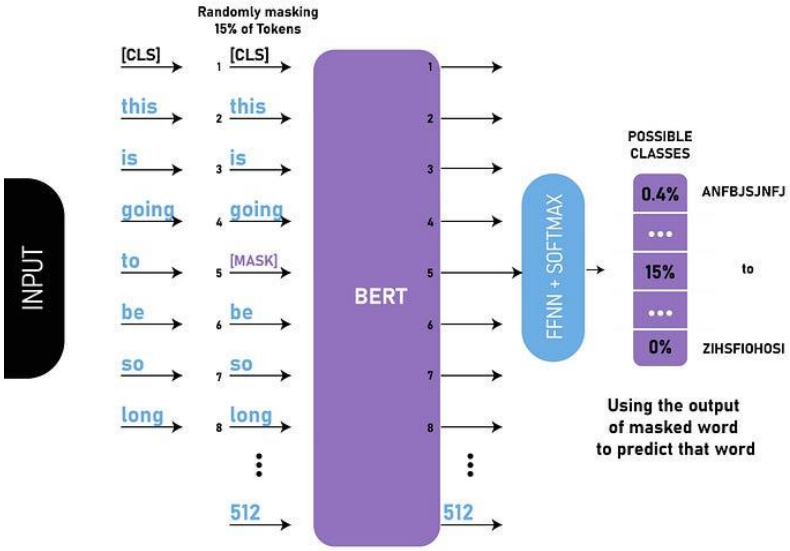

Fig 2.1 Diagrammatic representation of BERT architecture [15].

B. Fine-Tuning for Downstream Tasks

Fine-tuning BERT involves adapting the pre-trained weights and biases to better suit a specific NLP task, such as sentiment analysis[16] or sarcasm detection[17]. The goal is to leverage the pre-trained knowledge of the original model while adjusting it to fit the nuances of the target task.

The fine-tuning process typically includes:

- Adding additional layers: These new layers are added on top of the pre-trained BERT weights, allowing the model to capture more complex relationships between input tokens.
- Freezing pre-trained weights: The original BERT weights and biases are left unchanged during training, while only updating the newly added layers' parameters.

By fine-tuning BERT, we can:

- Improve performance: Fine-tuning allows the model to adapt to specific task requirements, potentially leading to improved results on that particular task.
- Reduce overfitting: By starting from a pre-trained model, we benefit from its learned representations and avoid overfitting as much as possible.

www.ijprems.com
editor@ijprems.com
INTERNATIONAL JOURNAL OF PROGRESSIVE RESEARCH IN ENGINEERING MANAGEMENT AND SCIENCE (IJPREMS)
(Int Peer Reviewed Journal)
Vol. 04, Issue 10, October 2024, pp : 831-839
e-ISSN : 2583-1062
Impact Factor : 7.001

5. EXPERINETAL SETUP

The experimental setup for this study involved fine-tuning a pre-trained BERT-base-uncased model on the SARC dataset to classify sarcastic text. The experiment was conducted using a range of hyperparameters and evaluation metrics to determine the optimal settings for the model.

A. Importing the Dataset

```
# Load the sarcasm dataset from the JSON file
with open('Sarcasm_Headlines_Dataset.json', 'r') as f:
    data = [json.loads(line) for line in f]

# Extract headlines and labels from the dataset
texts = [item['headline'] for item in data]
labels = [item['is_sarcastic'] for item in data]
```

Fig 3.1: Importing dataset

Since the dataset is mostly in a raw state which means we manually have to modify it according to our requirements; therefore, this step was performed to extract the necessary columns from the entire dataset which in our case was the column 'headline' and the 'is_sarcastic' i.e., our independent and dependent variables.

B. Tokenization

Tokenization [18] is the process of breaking down text into smaller units called tokens, which are individual words or characters. In natural language processing (NLP), tokenization is essential for text analysis tasks such as sentiment analysis, question answering, and text classification.

Text preprocessing and tokenization are crucial steps in any NLP-based research project, as they enable the conversion of raw text data into a format that can be processed by machine learning models. In our case, tokenization allows us to break down complex input headlines into individual tokens, which can then be fed into our BERT model for sentiment analysis.

The pre-processing and tokenization steps also ensure that all input sequences have a consistent length, which is necessary for the efficient processing of our model. By performing these steps, we are able to transform our raw text data into a format that can be effectively used by our model, ultimately leading to improved accuracy and performance.

```
# Load tokenizer and model
model_name = "bert-base-uncased"
tokenizer = BertTokenizer.from_pretrained(model_name)
model = BertForSequenceClassification.from_pretrained(model_name, num_labels=2)

# Tokenize the input data
def tokenize_function(texts):
    return tokenizer(texts, padding="max_length", truncation=True, max_length=128)

train_encodings = tokenize_function(train_texts)
val_encodings = tokenize_function(val_texts)
```

Fig 3.2: Loading model and tokenizer.

In this section, we perform text preprocessing and tokenization on the input data to convert it into a format that can be fed into our BERT-based sentiment analysis model.

Firstly, we split our dataset into training and validation sets using the `train_test_split()` function from Scikit-learn library. This is necessary for evaluating the performance of our model on unseen data. The resulting train and validation texts are then tokenized using the `tokenize_function()` function defined earlier.

The `tokenize_function()` function utilizes the pre-loaded BERT tokenizer to break down each input headline into individual tokens, which are then padded with zeros to a maximum length of 128 tokens. This ensures that all input sequences have a consistent format and can be processed efficiently by our model. Initially, we used the train_test_split() function from the Scikit-learn library to divide our dataset into training and validation sets.

C. Configuring the Model and Training

For this task, we leveraged a pre-trained BERT (Bidirectional Encoder Representations from Transformers) model, which has been proven effective for various Natural Language Processing (NLP) tasks due to its ability to capture contextual information from both directions (left-to-right and right-to-left). The BERT model is fine-tuned on our sarcasm detection dataset, allowing it to adapt to the specific nuances of sarcasm within text. The following section describes the configuration of the model and the training procedure used.

www.ijprems.com
editor@ijprems.com

INTERNATIONAL JOURNAL OF PROGRESSIVE RESEARCH IN ENGINEERING MANAGEMENT AND SCIENCE (IJPREMS)
(Int Peer Reviewed Journal)
Vol. 04, Issue 10, October 2024, pp : 831-839

e-ISSN : 2583-1062
Impact Factor : 7.001

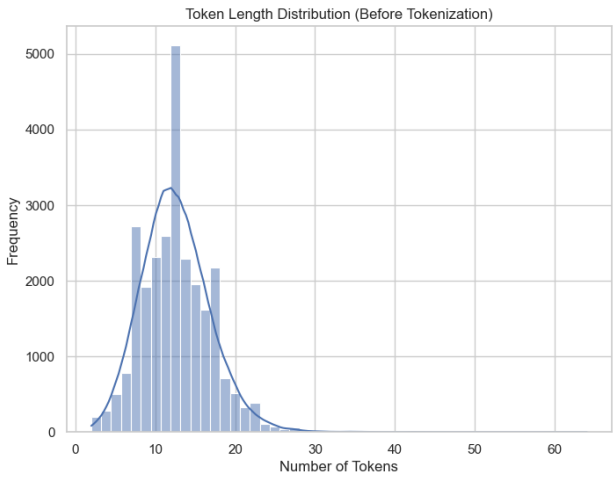

Fig 3.3: Token length distribution

We fine-tune the BERT model using the Hugging Face [19] Trainer class, which provides a high-level abstraction for training models. To evaluate the performance during training, we use accuracy as the primary metric. The accuracy score is calculated by comparing the predicted labels against the true labels in the evaluation dataset.

D. Hyperparameter Configuration

```
# Define training arguments
training_args = TrainingArguments(
    output_dir="./results",
    evaluation_strategy="epoch",
    learning_rate=2e-5,
    per_device_train_batch_size=16,
    per_device_eval_batch_size=16,
    num_train_epochs=3,
    weight_decay=0.01,
)
```

Fig 3.4: Setting hyperparameters

The training process is controlled using a set of hyperparameters, which are crucial for guiding the model's learning process and ensuring the fine-tuning procedure is efficient and effective. The hyperparameters used in our experiment are:

Output Directory (output_dir): The directory where all model checkpoints and results are stored during the training process. We set this to "./results" for easy access to the model's progress and outputs.

Evaluation Strategy (evaluation_strategy): This hyperparameter controls how often evaluation on the validation dataset is performed. We set it to "epoch," indicating that the model is evaluated at the end of each training epoch to monitor its performance on unseen data.

Learning Rate (learning_rate): The learning rate is one of the most important hyperparameters that controls how much to update the model's weights with respect to the loss gradient[20]. A smaller learning rate allows the model to fine-tune more cautiously, avoiding drastic weight updates. We set this to 2e-5, which is a commonly recommended rate for fine-tuning BERT models.

Batch Size for Training and Evaluation (per_device_train_batch_size and per_device_eval_batch_size): These hyperparameters define how many samples are processed at once during training and evaluation. We set both the training and evaluation batch sizes to 16, balancing computational efficiency and memory constraints on our hardware.

Number of Epochs (num_train_epochs): This defines the number of times the model will see the entire training dataset during training. We train the model for 3 epochs, which is sufficient to allow the model to converge without overfitting, based on empirical results from related works.

Weight Decay (weight_decay): Weight decay is a regularization technique[21] that helps prevent overfitting by penalizing large weights during training. We set the weight decay to 0.01, encouraging the model to generalize better to unseen data.

www.ijprems.com
editor@ijprems.com

INTERNATIONAL JOURNAL OF PROGRESSIVE RESEARCH IN ENGINEERING MANAGEMENT AND SCIENCE (IJPREMS)
(Int Peer Reviewed Journal)
Vol. 04, Issue 10, October 2024, pp : 831-839

e-ISSN : 2583-1062
Impact Factor : 7.001

6. MODEL PERFORMANCE AND ANALYSIS

[4008/4008 14:31, Epoch 3/3]

Epoch	Training Loss	Validation Loss	Accuracy
1	0.239100	0.206742	0.919693
2	0.116200	0.239112	0.936166
3	0.047100	0.350497	0.929427

Fig 4.1: Results after training

After configuring and fine-tuning the BERT model for sarcasm detection, we evaluated the model on the validation dataset to assess its performance and generalization ability. The evaluation process provides insights into how well the model performs on unseen data, which is crucial for determining its real-world effectiveness in sarcasm detection tasks.

A. Training Results

The model was trained over a total of 3 epochs, during which it achieved a relatively low training loss of 0.1475. This low loss indicates that the model successfully learned patterns from the sarcasm dataset during fine-tuning. The following are key training statistics captured after the completion of 4008 global steps:

- Global Step: 4008
- Training Loss: 0.1475
- Training Runtime: 872.64 seconds (~14.5 minutes)
- Samples Processed per Second: 73.46
- Steps per Second: 4.59

The total number of floating-point operations performed during training was approximately 4.22 quadrillion (4.22e15), reflecting the computational intensity of training large transformer-based models [22] like BERT.

B. Evaluation Metrics

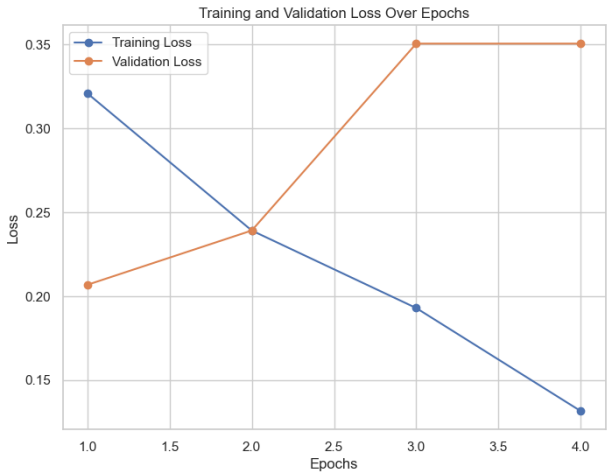

Fig 4.2: Training and validation loss over epochs

The model was evaluated on the validation dataset using the same Trainer class. The evaluation results are presented below:

- Evaluation Loss: 0.3505
- Evaluation Accuracy: 92.94%
- Evaluation Runtime: 18.79 seconds
- Samples Processed per Second: 284.25
- Steps per Second: 17.77
- Epoch: 3

C. Evaluation Loss

The evaluation loss of 0.3505 indicates that while the model performs well on the validation dataset, there is still some level of discrepancy between the predicted outputs and the ground truth labels. This loss is higher than the training loss (0.1475), which is expected, as the model is likely exposed to more challenging or previously unseen examples during evaluation. The difference between training and evaluation loss suggests that the model may have learned well from the training data but still encounters some difficulty with generalization, though not significantly.

www.ijprems.com
editor@ijprems.com
INTERNATIONAL JOURNAL OF PROGRESSIVE RESEARCH IN ENGINEERING MANAGEMENT AND SCIENCE (IJPREMS)
(Int Peer Reviewed Journal)
Vol. 04, Issue 10, October 2024, pp : 831-839
e-ISSN : 2583-1062
Impact Factor : 7.001

D. Evaluation Accuracy

The evaluation accuracy of 92.94% demonstrates the model's ability to correctly predict whether a given text is sarcastic or not in nearly 93 out of every 100 cases. This high level of accuracy reflects that the model has successfully captured the linguistic patterns and features associated with sarcasm.

E. Analysis of Results

The evaluation results offer valuable insights into the effectiveness of the fine-tuning process[23] and the overall performance of the model:

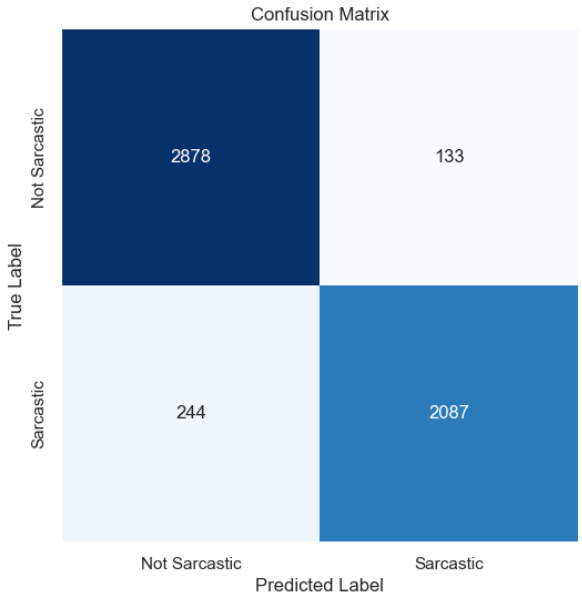

Fig 4.3: Confusion Matrix[24]

Low Loss and High Accuracy: The model's low evaluation loss of 0.3505, combined with its high accuracy of 92.94%, indicates that it has learned to generalize well to unseen data. These results suggest that the model is well-suited for the sarcasm detection task, capturing the subtle cues that differentiate sarcastic from non-sarcastic statements.

Generalization Gap: While there is a slight increase in evaluation loss compared to training loss (0.1475 → 0.3505), the model has not shown significant overfitting, as the evaluation accuracy remains high. The difference between the two losses points to the natural difficulty of the sarcasm detection task, which may involve nuanced language constructs that were not fully represented in the training data. However, this gap remains within acceptable limits for such complex NLP tasks.

Computational Efficiency: The model's ability to process over 284 samples per second during evaluation reflects its suitability for deployment in real-time applications, where latency and throughput are critical. Given that sarcasm detection often needs to be performed in real-time (e.g., social media moderation[25]), this efficiency is a significant advantage.

7. CONCLUSION

In this research, we explored the application of a fine-tuned BERT model for the task of sarcasm detection. Sarcasm detection presents a unique challenge in natural language processing (NLP) due to the inherent complexity of sarcastic language, which often involves subtle, indirect cues and contextual information.

Our approach involved fine-tuning a pre-trained BERT model on a labeled sarcasm dataset. The model was configured with specific hyperparameters, such as a learning rate of 2e-5, batch size of 16, weight decay of 0.01, and trained over 3 epochs. These hyperparameters were chosen based on best practices for fine-tuning transformer models and were key to achieving a balance between learning efficiency and performance.

The model achieved impressive results, with an evaluation accuracy of 92.94% and an evaluation loss of 0.3505, demonstrating that the BERT model is capable of effectively capturing and learning the linguistic and contextual patterns associated with sarcasm. The training loss of 0.1475 suggests that the model fit the training data well, while the relatively small gap between training and evaluation loss reflects a good generalization to unseen data, with minimal overfitting.

These findings show that transformer-based models like BERT, when fine-tuned on specific tasks, can achieve high accuracy even for challenging tasks like sarcasm detection. The model's ability to process 284 samples per second during

evaluation highlights its computational efficiency, making it a suitable candidate for real-time sarcasm detection applications.

However, sarcasm detection remains a difficult task due to its reliance on subtle contextual cues, which may not always be present in the text alone. Future work could explore improving performance by incorporating additional contextual information (e.g., multimodal data such as images or speech), leveraging larger and more diverse datasets, and experimenting with model ensembling or data augmentation techniques to push the boundaries of the model's capabilities.

Overall, this study demonstrates that BERT provides a strong foundation for sarcasm detection in text, offering both high accuracy and computational efficiency. As future NLP models and techniques continue to evolve, the potential for even more nuanced and accurate sarcasm detection remains promising.

8. FUTURE SCOPE

While the results of this research demonstrate the effectiveness of fine-tuning a BERT model for sarcasm detection, there are several opportunities for improvement and further exploration in future work. The nuances of sarcasm, often tied to context and external cues, present both challenges and areas for innovation in machine learning and NLP.

Incorporating Multimodal Data: Sarcasm often relies on more than just text, involving non-verbal cues like tone of voice, facial expressions, or even images (e.g., memes). Future work could involve integrating multimodal data, such as audio, video, or images, to improve the model's understanding of sarcasm. This could be achieved by combining models like BERT with vision or speech recognition systems to capture these cues more effectively.

Leveraging Larger and More Diverse Datasets: Sarcasm detection may benefit from training on larger, more varied datasets that capture diverse forms of sarcastic expression across different languages, dialects, and cultures. By training the model on a broader range of sarcastic statements, we can ensure better generalization and performance across various domains.

Advanced Model Architectures: While BERT has proven effective, newer and more advanced transformer architectures (e.g., RoBERTa, T5, GPT models) could potentially yield better performance. Future research could explore fine-tuning these models on sarcasm datasets and comparing their performance with BERT.

Ensemble Models: Ensemble methods, where multiple models are combined to make predictions, could be another avenue to improve sarcasm detection. By leveraging the strengths of different models (e.g., combining BERT with convolutional neural networks or LSTM models), ensembles could improve robustness and accuracy in sarcasm detection tasks.

Handling Contextual Dependencies: Sarcasm is often context-dependent, requiring understanding of previous conversation history or external knowledge. Future work could involve enhancing models with memory mechanisms, such as transformer-based models with attention to conversational context, or using pre-trained models that incorporate external knowledge bases for improved context understanding.

Real-Time Sarcasm Detection Applications: Given the model's efficiency during evaluation, it could be adapted for real-time sarcasm detection applications. Future work could explore deploying such models in real-world settings, such as social media platforms or conversational AI systems, and further optimizing inference times for scalable, low-latency deployment.

Improving Interpretability: Another area for future research is improving the interpretability of sarcasm detection models. Given the opaque nature of deep learning models like BERT, understanding how the model arrives at its predictions is crucial, especially for sensitive applications like content moderation. Exploring techniques like attention visualization or explainability methods could make the model's predictions more transparent.

Exploring Cross-Lingual Sarcasm Detection: Many current sarcasm detection models are focused on English-language datasets. Future research could explore cross-lingual sarcasm detection, where a model fine-tuned on one language is able to generalize across other languages, or building multilingual sarcasm detection models that can handle various linguistic and cultural nuances.

9. REFERENCES

[1] O. Tsur, D. Davidov, and A. Rappoport, "ICWSM—a great catchy name: Semi-supervised recognition of sarcastic sentences in online product reviews," Proceedings of the Fourth International Conference on Weblogs and Social Media (ICWSM), 2010.

[2] D. Davidov, O. Tsur, and A. Rappoport, "Enhanced sentiment learning using Twitter hashtags and smileys," Proceedings of the 23rd International Conference on Computational Linguistics (COLING), 2010.

[3] A. Reyes, P. Rosso, and D. Buscaldi, "From humor recognition to irony detection: The figurative language of social media," Data & Knowledge Engineering, vol. 74, pp. 1-12, 2012.

[4] D. Ghosh and T. Veale, “Fracking sarcasm using neural network ensembles,” Proceedings of the 7th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis, 2016.
[5] S. Amir, G. Ling, and B. C. Wallace, “Modelling context with user embeddings for sarcasm detection in social media,” Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2016.
[6] Q. Zhang, Y. Gong, and J. Guo, “Attentive convolutional networks for sarcasm detection,” Proceedings of the 26th International Conference on World Wide Web Companion (WWW), 2017.
[7] A. Joshi, V. Sharma, and P. Bhattacharyya, “Harnessing context incongruity for sarcasm detection,” Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (ACL), 2015.
[8] Misra, Rishabh and Prahal Arora. "Sarcasm Detection using News Headlines Dataset." AI Open (2023).
[9] Misra, Rishabh and Jigyasa Grover. "Sculpting Data for ML: The first act of Machine Learning." ISBN 9798585463570 (2021).
[10] https://www.intel.com/content/www/us/en/products/sku/134587/intel-core-i512400f-processor-18m-cache-up-to-4-40-ghz/specifications.html
[11] https://www.nvidia.com/en-us/geforce/graphics-cards/40-series/rtx-4060-4060ti/
[12] Koroteev, Mikhail V. "BERT: a review of applications in natural language processing and understanding." arXiv preprint arXiv:2103.11943 (2021). (bert)
[13] Chowdhary, KR1442, and K. R. Chowdhary. "Natural language processing." Fundamentals of artificial intelligence (2020): 603-649. (nlp)
[14] Kenton, Jacob Devlin Ming-Wei Chang, and Lee Kristina Toutanova. "Bert: Pre-training of deep bidirectional transformers for language understanding." Proceedings of naacL-HLT. Vol. 1. 2019.
[15] https://miro.medium.com/v2/resize:fit:4800/format:webp/0*ZSJCxc91qsc45QuZ.jpg (Bert Image)
[16] Medhat, Walaa, Ahmed Hassan, and Hoda Korashy. "Sentiment analysis algorithms and applications: A survey." Ain Shams engineering journal 5.4 (2014): 1093-1113.
[17] Joshi, Aditya, Pushpak Bhattacharyya, and Mark J. Carman. "Automatic sarcasm detection: A survey." ACM Computing Surveys (CSUR) 50.5 (2017): 1-22.
[18] Webster, Jonathan J., and Chunyu Kit. "Tokenization as the initial phase in NLP." COLING 1992 volume 4: The 14th international conference on computational linguistics. 1992.
[19] https://huggingface.co/
[20] Li, Li, Miloš Doroslovački, and Murray H. Loew. "Approximating the gradient of cross-entropy loss function." IEEE access 8 (2020): 111626-111635.
[21] Leibbrandt, George. "Introduction to the technique of dimensional regularization." Reviews of Modern Physics 47.4 (1975): 849.
[22] Gillioz, Anthony, et al. "Overview of the Transformer-based Models for NLP Tasks." 2020 15th Conference on computer science and information systems (FedCSIS). IEEE, 2020.
[23] Friederich, Simon. "Fine-tuning." The Stanford encyclopedia of philosophy (2017).
[24] Townsend, James T. "Theoretical analysis of an alphabetic confusion matrix." Perception & Psychophysics 9 (1971): 40-50. (confusion matrix)
[25] Veglis, Andreas. "Moderation techniques for social media content." Social Computing and Social Media: 6th International Conference, SCSM 2014, Held as Part of HCI International 2014, Heraklion, Crete, Greece, June 22-27, 2014. Proceedings 6. Springer International Publishing, 2014.