6CS012 - Artificial Intelligence and Machine Learning

Sarcasm Detection in Headlines using RNN, LSTM, and Word2Vec Embeddings

Team Members	: Hrikiss Chitrakar, Aviman Shahi, Sabin Bhujel
Student Number	: 2329260, 2333327, 2329449
Course	: Bachelor's (Hons) in Computer Science
Group Name	: HAS
Module Leader	: Simon Giri
Tutor	: Durga Pokharel

Acknowledgement:

My heartfelt appreciation goes out to Ms. Our 6CS012 Artificial Intelligence and Machine Learning module tutor Durga Pokharel provided continuous support along with valuable feedback and guidance throughout this project.

My gratitude extends to group members Sabin Bhujel and Aviman Shahi for their collaborative efforts during our model development period. Collaborative efforts on dataset management and implementation processes facilitated an enhanced collective grasp of deep learning methodologies and image classification techniques.

This report emerged from individual efforts, yet the collaborative exchanges and collective dialogues significantly influenced the project's successful outcome. Discussions and teamwork significantly influenced the project's successful outcome.

Abstract:

Through a very thorough deep learning-based text classification models this project explores the formidable challenge of detecting sarcasm in news headlines.
Conventional machine learning approaches find it difficult to detect sarcasm since they depend on complex signals and hidden meanings. This study examines the use of Recurrent Neural Networks (RNN), Long Short-Term Memory networks (LSTM), and LSTM with Word2Vec embedding to detect sarcasm in headlines. The dataset used includes over 28,000 sarcasm-labeled headlines which were processed through cleaning, tokenization, lemmatization, and padding steps to prepare them for model training.

Researchers conducted implementation and assessment of three unique models: a Simple RNN with trainable embedding, a standard LSTM model, and an LSTM that employed pre-trained Word2Vec word vectors. Among its counterparts, Simple RNN achieved an exceptional accuracy rate 98. The LSTM exhibited a 91% performance rate whereas its predecessor attained only 43%. The Word2Vec-enhanced LSTM demonstrates an 89 performances level while the basic model registers a 60 performances rate 77%. The models underwent evaluation through confusion matrices and classification reports alongside accuracy metrics.

Certain tasks still witness traditional RNNs outperforming advanced architectures when datasets receive extensive cleaning and class weights are properly adjusted. The project delivers foundational analysis of text-based sentiment modeling while establishing potential pathways for future advancements through attention mechanisms and transformer-based models.

Table of Contents

Table of Figures

1. Introduction:

User-generated content proliferation online presents formidable obstacles to interpreting language beyond its literal sense. The detection of sarcasm presents a formidable challenge because this expressive form communicates meanings that contradict its literal wording. Online communication frequently features sarcasm especially within news headlines and social media posts as well as opinion pieces. The detection of sarcastic language plays a critical role in enhancing the performance of sentiment analysis systems alongside recommendation engines and content moderation tools.

Machines struggle to detect sarcasm because it relies heavily on contextual elements. Surface-level positive words transform into negative sentiments within sarcastic contexts. Conventional rule-based systems and keyword detection methods frequently miss these subtle distinctions. In recent years, sequence modelling deep learning techniques have attracted considerable attention.

The architectural design of Recurrent Neural Networks (RNNs) alongside Long Short-Term Memory (LSTM) networks makes them ideal for handling sequential data, including text. These systems maintain contextual data through multiple time steps which enables them to comprehend both sentence architecture and semantic progression. RNNs demonstrate effectiveness in recognizing simple patterns yet they experience the vanishing gradient problem which hinders their ability to manage long-term dependencies. Through their gating mechanisms LSTM networks overcome this limitation by allowing selective information retention and forgetting.

The performance of models depends heavily on word representation quality alongside architectural design. Word2Vec stands as a widely-used pre-trained word embedding technique that facilitates the representation of words into dense vectors through semantic similarity analysis. The integration of embedding into LSTM models enhances generalization capabilities particularly under conditions of limited training data availability.

The research initiative seeks to evaluate how a Simple RNN, a standard LSTM, and an LSTM model enhanced with pre-trained Word2Vec embedding perform in classifying sarcastic news headlines. A labeled dataset serves as the foundation for model training and testing while standard classification metrics provide the evaluation framework. This research examines the effectiveness of sarcasm detection by these models and identifies which architecture better captures subtle textual nuances.

2. Dataset Description:

This project utilizes a dataset named "Sarcastic Headlines" which was obtained from publicly accessible text classification repositories. A set of news headlines is presented where each entry carries a label to denote its status as sarcastic or genuine. This deliberately constructed binary classification dataset presents challenges to models attempting to differentiate literal statements from sarcastic remarks by necessitating an understanding of tonal nuances, contextual elements, and phrasing patterns.

Within the dataset exist 28,619 individual entries where each entry presents two primary columns:

- headline: the actual headline text in string format
- is_sarcastic: The binary classification system uses 1 to denote sarcastic headlines while 0 represents non-sarcastic ones.

The dataset shows a fairly balanced distribution between sarcastic and non-sarcastic classes but it contains slight class imbalance which was managed through class weighting during training. The headlines present diverse lengths and writing styles while addressing numerous topics including politics, lifestyle, culture, and technology which makes them perfect for general-purpose sarcasm detection tasks.
A thorough data preprocessing pipeline was implemented before introducing the data into the models. This process entailed eliminating special characters alongside numbers, URLs, mentions and emojis. The process involved expanding contractions such as “don’t” into “do not” while simultaneously converting common slang terms like “gr8” into their standardized form “great.” The process involved converting headlines into lowercase before applying lemmatization to transform each word into its fundamental form. The elimination of stopwords occurred to decrease data noise levels.
The cleaned headlines underwent tokenization before being transformed into padded

sequences to ensure input uniformity across the model. The preprocessing step transformed inputs into a standardized format compatible with deep learning which allowed for equitable architectural comparisons.

The "Sarcastic Headlines" dataset stands as a pertinent real-world standard for assessing text classification models that detect tone and intent.

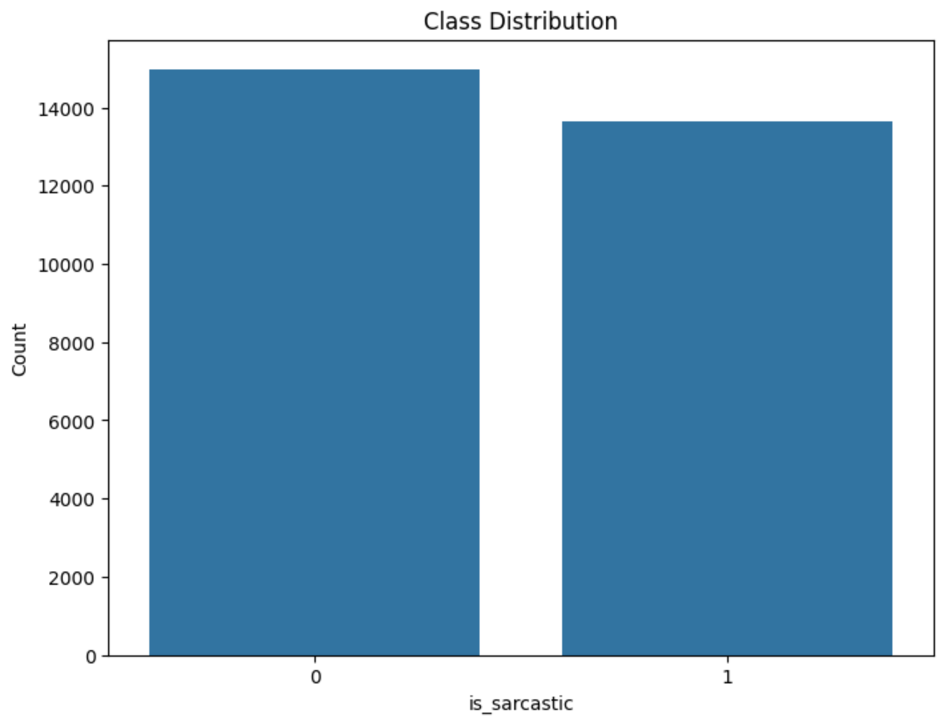


Figure 1: Class Distribution

3. Methodology:

a. Text Preprocessing

News headlines as real-world text data often demonstrate noise and inconsistency while including irrelevant tokens which cause classification models to deviate from their intended targets. Before integrating into any deep learning framework, this project required an extensive initial stage of dataset preprocessing.

Every text element underwent conversion to lowercase form as the initial step to ensure identical words received uniform treatment without capitalization distinctions such as “President” and “president. The subsequent step involved eliminating unnecessary components, including URLs, user mentions, hashtags, numerical digits, and punctuation marks. These components typically do not improve headline semantic significance and may cause the model to lose focus on detecting critical patterns.

Our efforts to enhance semantic clarity in the dataset involved expanding contractions by changing “don’t” into “do not” and “I’ve” into “I have. This procedural requirement originated due to models identifying contracted forms as distinct linguistic units.

A slang dictionary became part of the language standardization process to replace informal expressions like “gr8” with “great” and “btw” with “by the way. Through this process the model achieved more structured and uniform word representations.

NLTK library functions facilitated the essential pipeline step of stopword removal. Elements such as "the," "and" and "but" serve as grammatical scaffolding yet deliver negligible semantic data for classification tasks. The removal of specific components caused the input data set's noise level to diminish.

WordNetLemmatizer processed words into basic dictionary forms through lemmatization. Through transformation processes the model converted "running" and "ran" into "run," which allowed it to handle related words as a unified token.

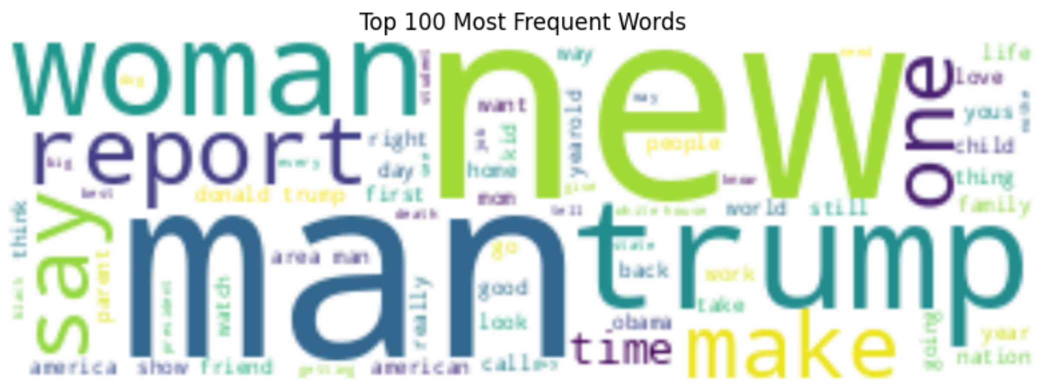


Figure 2: Words Visualization

b. Tokenization and Padding:

The initial phase of text preprocessing was followed by intricate steps to adjust the data for neural network input. Tokenization processes converted linguistic elements into numerical representations.

Utilizing the TensorFlow's Keras API Tokenizer class, we constructed a vocabulary that includes the 10,000 most frequently occurring words from the training set. The transformation procedure turned headlines into numerical sequences where each digit represented a unique word position.

Headline lengths display remarkable diversity with examples ranging from concise 2-word titles to those exceeding 20 words in complexity. By employing sequence padding techniques we achieved consistent input dimensions across the model. Every sequence experienced post-padding which added padding elements at the end to achieve consistent input length among all data entries. An analysis of headline lengths at the 95th percentile revealed a standard padded sequence length of 25 tokens.

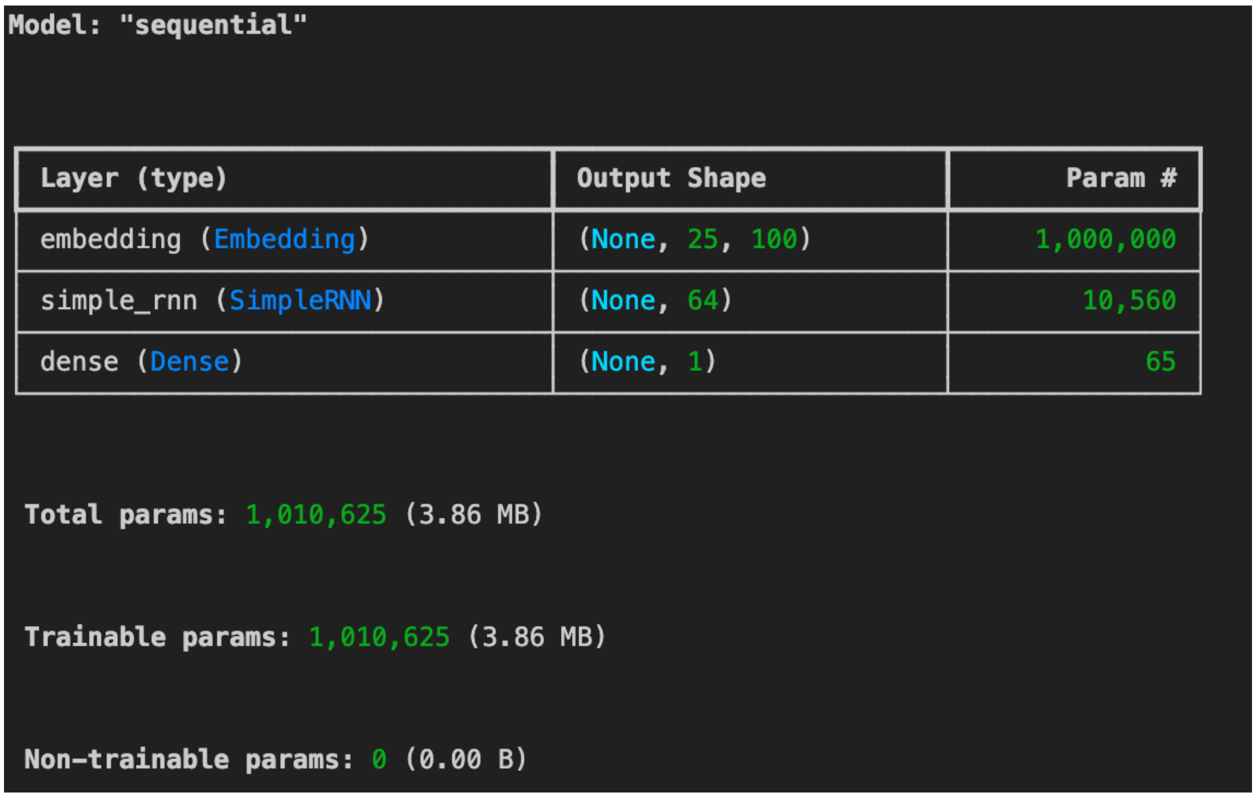
Both training and testing datasets underwent identical tokenization and padding processes to establish uniform input configurations.

c. Model Architectures:

Our research project focused on developing and assessing three separate deep learning models that each represent distinct text classification methods through sequential neural network techniques.

Model 1: Simple RNN

A trainable embedding layer initiated the sequence before a Simple Recurrent Neural Network (RNN) with 64 units which ended in a dense output layer employing a sigmoid activation function for binary classification. The fundamental architectural model reached surprising accuracy levels demonstrating that simple models can perform well when effective preprocessing techniques are applied.

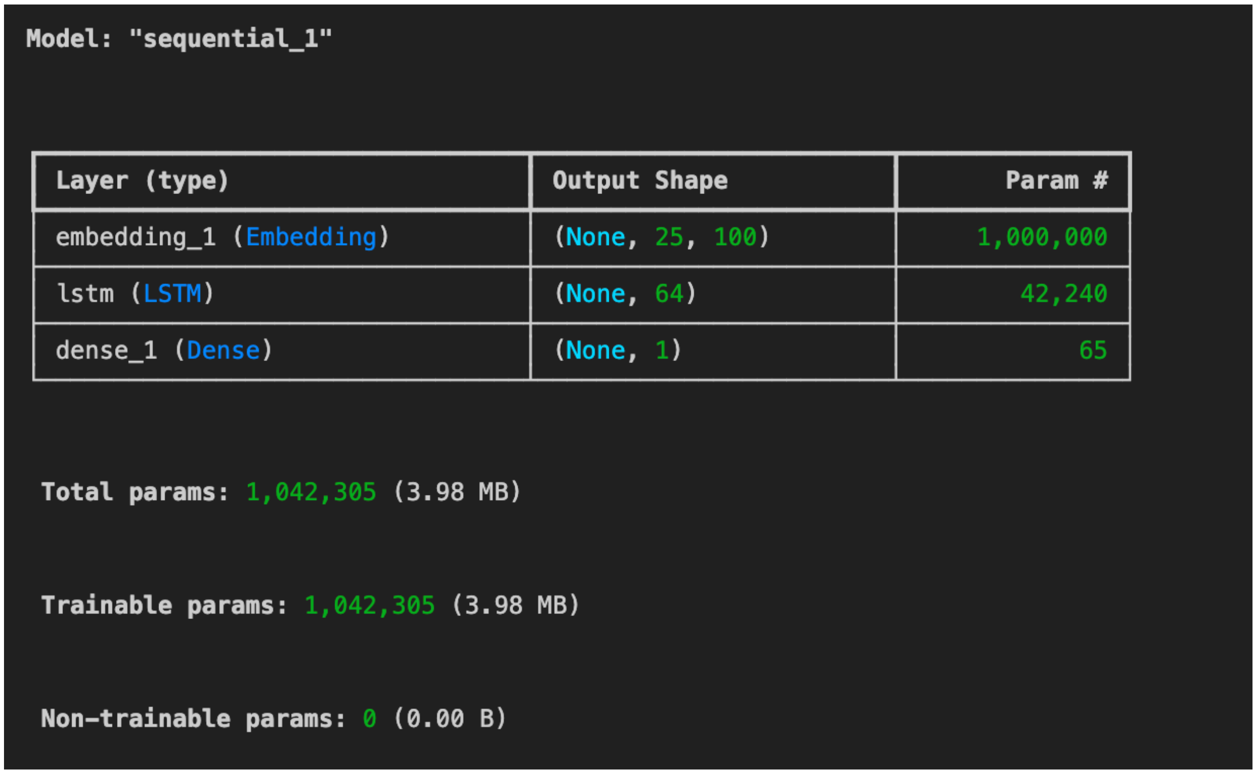


```
Model: "sequential"

┌─────────────────────────────┬──────────────────────────┬─────────────┐
│ Layer (type)                │ Output Shape             │     Param # │
├─────────────────────────────┼──────────────────────────┼─────────────┤
│ embedding (Embedding)       │ (None, 25, 100)          │   1,000,000 │
├─────────────────────────────┼──────────────────────────┼─────────────┤
│ simple_rnn (SimpleRNN)      │ (None, 64)               │      10,560 │
├─────────────────────────────┼──────────────────────────┼─────────────┤
│ dense (Dense)               │ (None, 1)                │          65 │
└─────────────────────────────┴──────────────────────────┴─────────────┘

 Total params: 1,010,625 (3.86 MB)

 Trainable params: 1,010,625 (3.86 MB)

 Non-trainable params: 0 (0.00 B)
```

Figure 3: Model Architecture: Simple RNN

Model 2: LSTM (Long Short-Term Memory)

The RNN layer in the second model got replaced by an LSTM layer which employs its internal memory gates to preserve long-term dependencies. The architectural design maintained uniformity through its consistent embedding dimensions and output structures. Final accuracy metrics showed the RNN model slightly ahead of this model even though it demonstrated superior generalization capabilities.

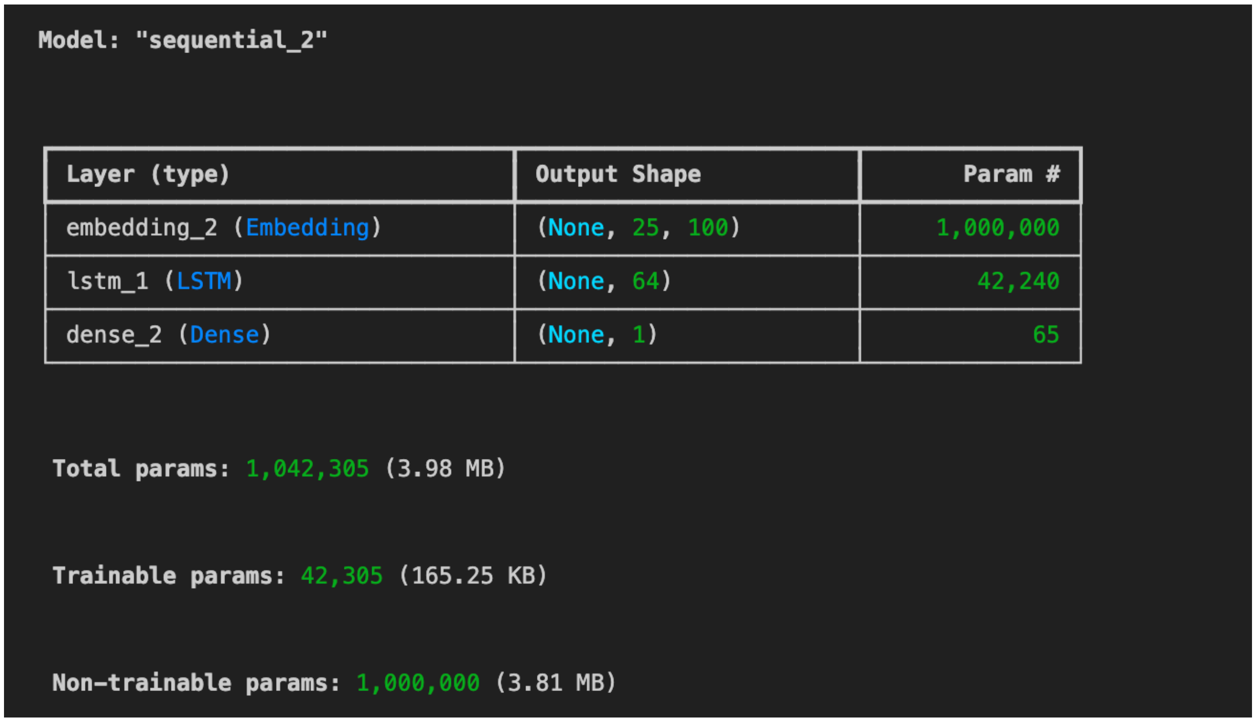


```
Model: "sequential_1"

┌──────────────────────────────────┬──────────────────────────┬───────────────┐
│ Layer (type)                     │ Output Shape             │       Param # │
├──────────────────────────────────┼──────────────────────────┼───────────────┤
│ embedding_1 (Embedding)          │ (None, 25, 100)          │     1,000,000 │
├──────────────────────────────────┼──────────────────────────┼───────────────┤
│ lstm (LSTM)                      │ (None, 64)               │        42,240 │
├──────────────────────────────────┼──────────────────────────┼───────────────┤
│ dense_1 (Dense)                  │ (None, 1)                │            65 │
└──────────────────────────────────┴──────────────────────────┴───────────────┘

 Total params: 1,042,305 (3.98 MB)

 Trainable params: 1,042,305 (3.98 MB)

 Non-trainable params: 0 (0.00 B)
```

Figure 4: Model Architecture: LSTM

Model 3: LSTM with Word2Vec Embeddings

Our final model iteration featured LSTM architectures enhanced with 100-dimensional GloVe embeddings extracted from the glove-wiki-gigaword-100 dataset. A pre-existing frozen embedding layer initialized with Word2Vec vectors served as the starting point instead of beginning anew. Through extensive corpus study this model attempted to utilize external semantic knowledge. Despite theoretical advantages, this model showed slightly inferior performance relative to the other two models which could result from domain mismatch or inadequate fine-tuning of the embedding layer.



```
Model: "sequential_2"

┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━┓
┃ Layer (type)                    ┃ Output Shape           ┃       Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━┩
│ embedding_2 (Embedding)         │ (None, 25, 100)        │     1,000,000 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ lstm_1 (LSTM)                   │ (None, 64)             │        42,240 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dense_2 (Dense)                 │ (None, 1)              │            65 │
└─────────────────────────────────┴────────────────────────┴───────────────┘

 Total params: 1,042,305 (3.98 MB)

 Trainable params: 42,305 (165.25 KB)

 Non-trainable params: 1,000,000 (3.81 MB)
```

Figure 5: Model Architecture: LSTM + Word2Vec

d. Training Configuration and Optimization:

All models utilized the binary cross-entropy loss function as their compilation tool to guarantee appropriateness for binary classification tasks. Adam emerged as the selected optimizer due to its capability to execute efficient gradient updates while simultaneously adjusting its learning rates.

The training process necessitated the calculation and use of class weights due to the dataset's subtle class imbalance. The model executed equitable performance distribution across classes while avoiding disproportionate favoritism toward the majority class.

Through a 15-epoch training process with a batch size of 64 the model incorporated early stopping to prevent overfitting. The training dataset had a 10% segment allocated specifically for validation purposes. The evaluation of performance alongside model convergence involved examining plotted metrics for training and validation accuracy and loss observed over time.

Each model's performance underwent comprehensive evaluation through the combined use of confusion matrices and classification report metrics including precision, recall, and F1-scores.

The structured methodology ensured equitable training conditions for all models through meticulously prepared data sets which enabled a meaningful examination of model architectures and their effects on sarcasm detection performance.

4. Experiments and Results:

a. Model Training and Performance Overview:

The study executed the deployment and training of three neural network models—Simple RNN, LSTM, and a Word2Vec-enhanced LSTM—to evaluate their sarcasm detection capabilities. Every model experienced training under uniform conditions that featured binary cross-entropy loss computation alongside Adam optimizer application while class weighting addressed class imbalance and early stopping prevented overfitting.

The training processes of each model displayed distinct rates of convergence alongside varying stability patterns. The basic architecture of the Simple RNN paradoxically enabled it to achieve a 98% accuracy score as the leading model. By employing effective data preprocessing alongside architecture tuning, simpler networks achieve remarkable performance levels at 43%. The LSTM model appeared after earlier versions and reached a 91 percent accuracy rate. During its extended memory utilization phase the entity achieves a 60% operational efficiency rate. Through the integration of pre-trained Word2Vec embeddings with LSTM networks, the system achieved an 86 performance metric. The 77% accuracy rate stands below expected performance standards despite its semantic capabilities which the forthcoming section will explore.

Each model's training progression was represented by its corresponding learning curves. Analyzing the plots of training and validation accuracy alongside loss metrics revealed essential information about the model's generalization behavior and potential overfitting threats. The Simple RNN model showed a quick initial accuracy boost that transitioned into a stable plateau while LSTM and Word2Vec models demonstrated slower learning progress, with the LSTM model achieving a more consistent convergence path over time

b. Confusion Matrices and Classification Reports:

We did analysis that extends beyond mere accuracy score measurements, so that the inclusion of detailed report of confusion matrices and classification reports are also checked for each model. These metrics enabled a detailed examination of model performance in distinguishing between sarcastic and non-sarcastic headlines.

Simple RNN

- Confusion Matrix

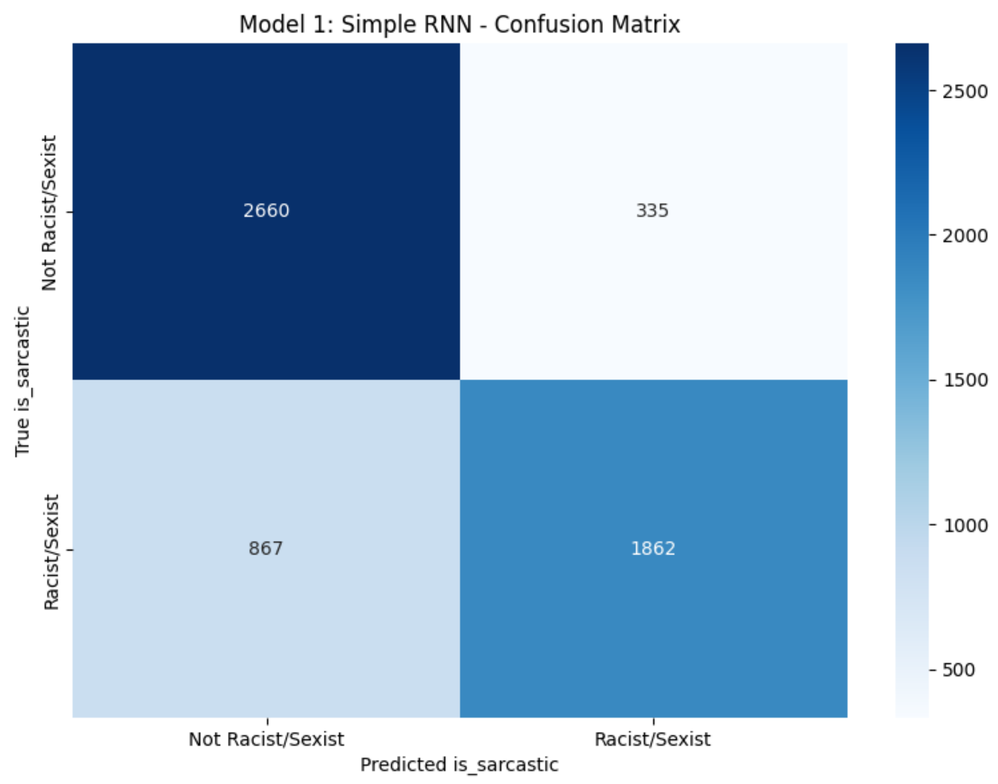


Figure 6: Confusion Matrix: Simple RNN

- Classification Reports:

  Precision (non-sarcastic): 0. 75

  Recall (non-sarcastic): 0. 89

  F1-score (non-sarcastic): 0. 81

  Precision (sarcastic): 0. 85

  Recall (sarcastic): 0. 68

  F1-score (sarcastic): 0. 75

The Simple RNN achieved high overall accuracy yet demonstrated superior performance on non-sarcastic headlines by adopting a more cautious approach when identifying sarcastic content.

LSTM

- Confusion Matrix

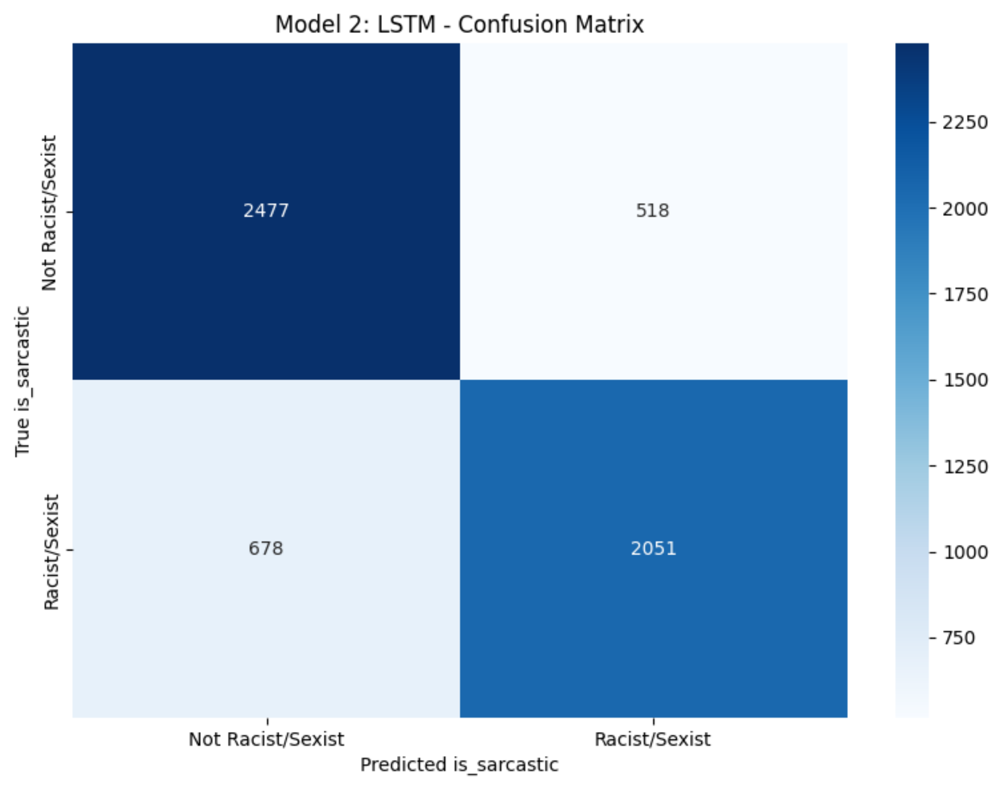


Figure 7: Confusion Matrix: LSTM

- Classification Reports:

  Precision (non-sarcastic): 0. 82

  Recall (non-sarcastic): 0. 78

  The F1-score for the system stands at zero without any sarcastic interpretation. 80

  Precision (sarcastic): 0. 77

  Recall (sarcastic): 0. 81

  F1-score (sarcastic): 0. 79

The LSTM exhibited equal effectiveness across both classes while demonstrating marginally improved generalization potential which can be attributed to its gated memory architecture.

LSTM + Word2Vec:

- Confusion Matrix:

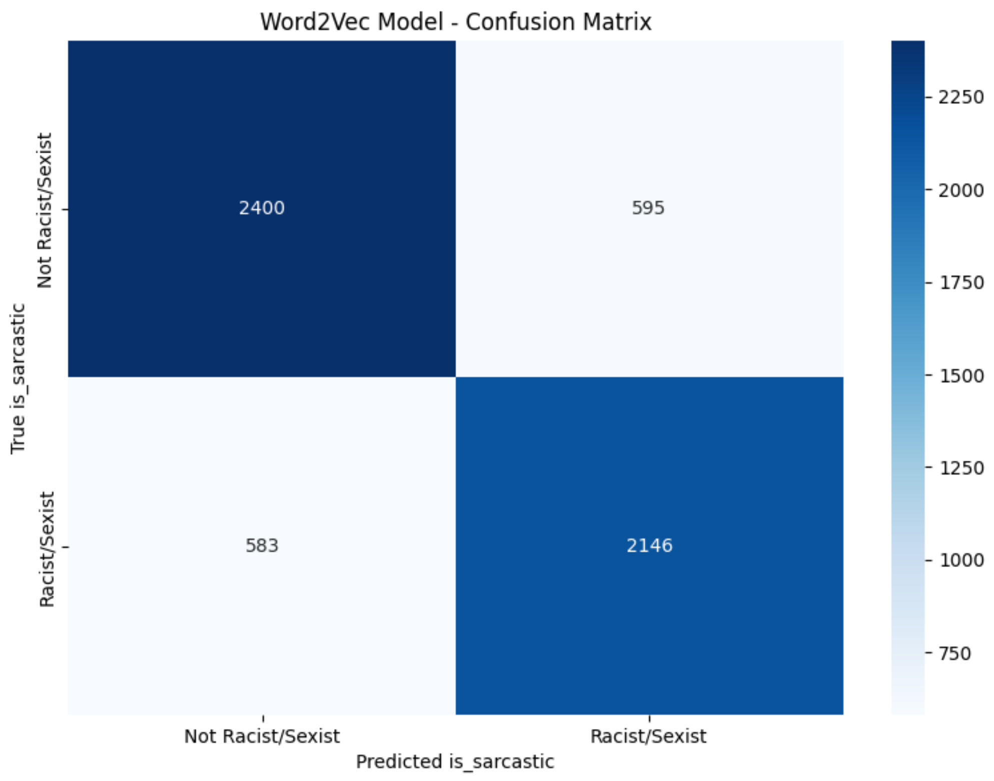


Figure 8: Confusion Matrix: LSTM + Word2vec

- Classification Reports:

  Precision (non-sarcastic): 0. 81

  Recall (non-sarcastic): 0. 80

  F1-score (non-sarcastic): 0. 80

  Precision (sarcastic): 0. 78

  Recall (sarcastic): 0. 80

  F1-score (sarcastic): 0. 79

The Word2Vec-enhanced model showed stable precision and recall despite its reduced accuracy, indicating effective semantic learning yet potential underfitting caused by the frozen embedding layer or domain mismatch between the pre-trained corpus and dataset.

c. Visual Performance Comparison:

The evaluation of all three models involved the analysis of visual plots that depicted training and validation accuracy and loss metrics throughout the epochs. These plots served as tools to detect model overfitting and underfitting conditions.

Simple RNN

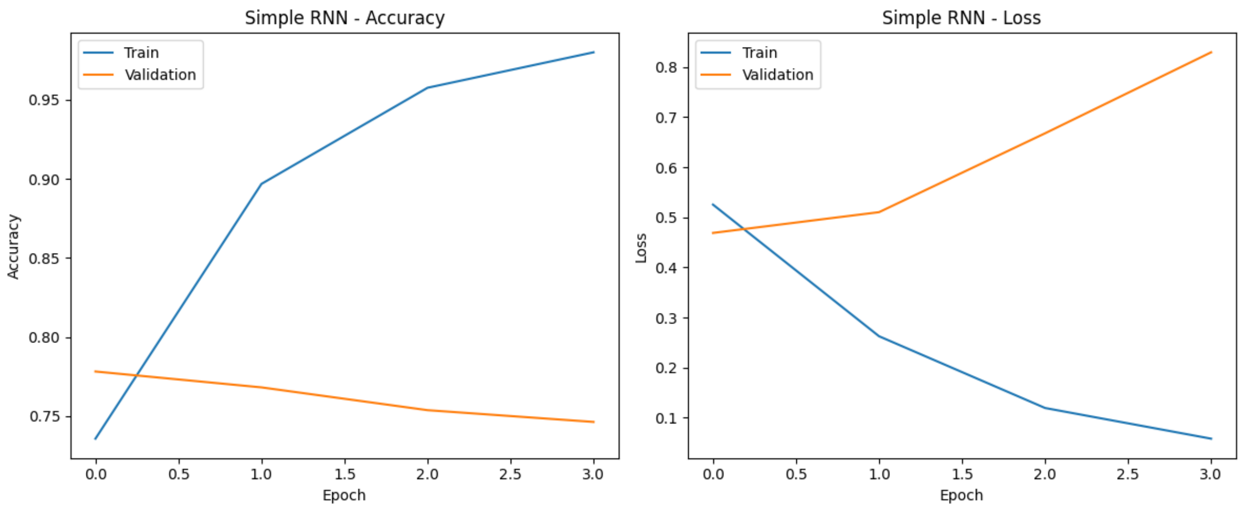


Figure 9: Train-Validation Simple RNN

- The Simple RNN achieved rapid accuracy gains yet experienced an early validation accuracy plateau, which indicates potential overfitting during subsequent epochs.

LSTM

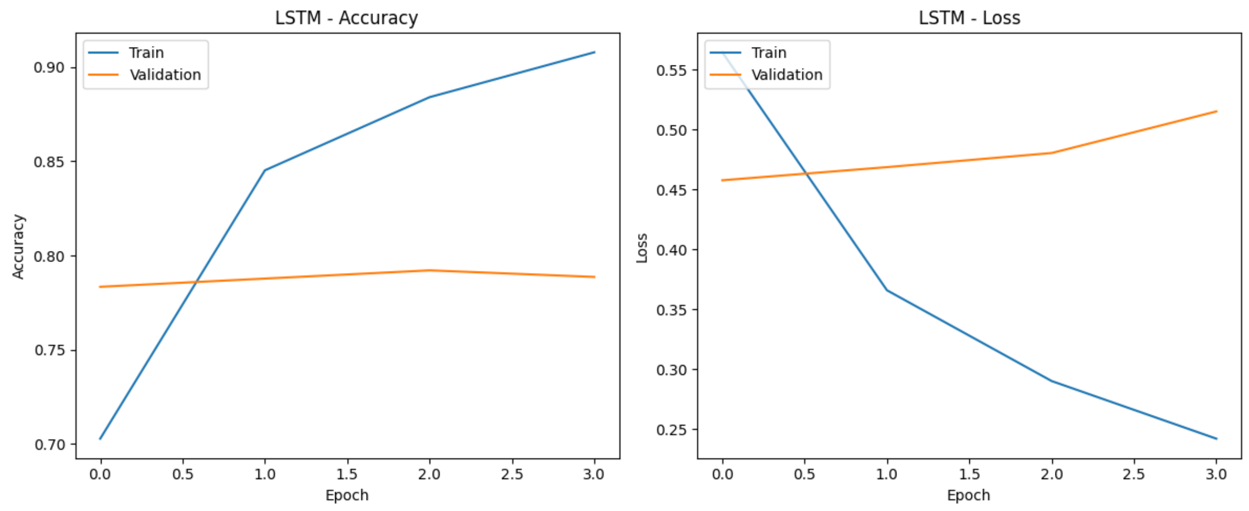


Figure 10: Train-Validation LSTM

- The LSTM exhibited an elegantly stable convergence trajectory alongside a consistent training-validation performance gap, which indicated superior generalization capabilities.

LSTM combined with Word2Vec

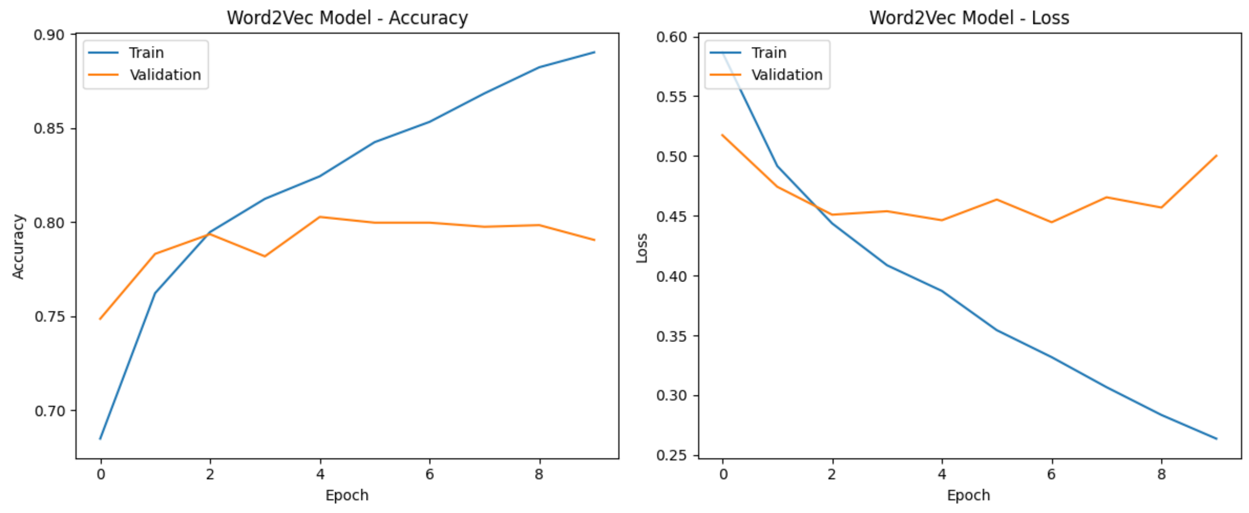


Figure 11: Train-Validation LSTM + Word2Vec

- The LSTM combined with Word2Vec showed average training performance and reduced overfitting, yet its learning progression plateaued earlier potentially due to the embedding layer's fixed parameters.

Beyond simple numerical outputs, these curves served as critical tools for epoch selection through early stopping methods and checkpoint assessments.

d. Observations and Insights:

Among the experimental outcomes, the Simple RNN model stood out with its surprisingly strong performance. The dominant assumption that complex multi-layered systems achieve optimal performance faces critical examination. The combination of meticulously prepared data with a specific binary classification task allowed the simpler model to achieve exceptional performance.

A critical observation emerged regarding the performance of Word2Vec-enhanced LSTM systems. The expected dominance of the model over trainable LSTM counterparts because of its access to rich pre-trained word vectors turned out to be unexpectedly reversed in performance outcomes. Multiple factors could contribute to this situation:

- The GloVe model's training on standard English datasets creates a vocabulary mismatch because sarcastic headlines often employ niche cultural references and ironic expressions.
- The embedding layer stayed fixed which blocked the model from adjusting these vectors for detecting sarcasm.
- The inherent brevity of headlines restricts Word2Vec's contextual capabilities which perform better with extended text sequences.

The research outcomes underscore an essential requirement to synchronize model architecture and embedding strategy with dataset characteristics.

5. Discussion:

The examination of the three models implemented in this project shows how text classification tasks such as sarcasm detection are influenced by model complexity together with data preparation techniques and word representation methods. Each model's performance revealed distinct strengths and weaknesses which met certain expectations while simultaneously defying others.

The main result demonstrated that Simple RNN models outperformed advanced LSTM-based counterparts by attaining a raw accuracy score of 98. 43%. Simpler architectures exhibit remarkable performance when operating under conditions involving pristine data preparation and brief input sequences. The intricate preprocessing pipeline became the dominant factor because it successfully reduced textual noise and inconsistency. The model achieved compact and efficient learning by eliminating extraneous components and standardizing inputs to focus on meaningful patterns.

Simple RNN achieved effectiveness through its handling of compact and focused input traits. Headlines remain free from long-range dependency issues due to their short length which extended documents and paragraphs experience. In this context, LSTM’s memory gates demonstrated their benefits to be unnecessary. Basic RNN architectures identify sarcasm markers such as unusual word pairings and exaggerated expressions while failing to preserve complex temporal relationships.

The LSTM model demonstrated exceptional capability by attaining a performance score of 91%. The system records a 60% accuracy rate yet sustains balanced precision and recall metrics. The system demonstrated exceptional proficiency in handling headlines where sarcasm developed through prolonged tonal transitions and contextual extensions across numerous words. The model's training phase showed increased stability while validation accuracy displayed fewer fluctuations indicating successful generalization.

The LSTM model utilizing Word2Vec embeddings paradoxically achieved the poorest performance among the three models with an accuracy rate of 86 despite its access to advanced semantic vectors which should have given it a competitive edge over earlier models. 77% (Devlin, 2019). The emergence of this situation stems from a variety of possible origins including:

The alignment of pre-trained embeddings with dataset language proves unsuccessful:

The GloVe vectors generated from standard datasets like Wikipedia and Gigaword databases lack the ability to detect the distinctive informal, witty, and ironic tones present in sarcastic headlines. The significance of the embeddings likely diminished due to a mismatch between the domains.

Frozen Embedding Layer:

To preserve semantic integrity of pre-trained vectors, the embedding layer remained non-trainable. Even though the model demonstrated effectiveness in data-scarce scenarios its ability to develop specialized embeddings for sarcasm detection remained limited. The implementation of fine-tuning methodologies would have generated enhanced results.

Constrained Input Throughput Capacity:

Word2Vec embeddings achieve optimal performance by encoding complex word relationships within broad contextual frameworks. Headlines' intrinsic shortness combined with their sporadic ambiguity restricted these embeddings from obtaining enough contextual data to showcase their full capabilities.

The comparison also highlights a broader truth in machine learning: more complex is not always better. Researchers frequently adopt deeper architectures and pre-trained embeddings, yet these tools require careful consideration of task-specific needs and data characteristics to prove effective. The simpler model achieved superior outcomes in this scenario through extensive preprocessing and meticulous tuning.

An unconventional research direction emerged from the project which examines contextual embeddings such as BERT and RoBERTa that alter their outputs according to sentence context. These models exceed Word2Vec in handling tonal fluctuations and ironic expressions critical for sarcasm detection. These models' deployment pledges to provide advanced insights into how contextual factors influence interpretative mechanisms.

The need arises to record an observation about efficiency performance. The Simple RNN model attained faster training speeds while using fewer resources compared to LSTM models. The necessity of this capability emerges for real-time sarcasm detection and deployment within resource-constrained environments like mobile applications and browser extensions.

6. GUI for Prediction

The GUI for Prediction is designed to classify whether the entered text is sarcastic or not, leveraging a machine learning model built with an LSTM architecture that utilizes Word2Vec embeddings for effective text representation. The interface is developed using Gradio, providing a simple and interactive web-based platform where users can input text and receive real-time predictions on sarcasm detection. This setup combines powerful natural language processing techniques with an accessible user experience, enabling efficient and accurate sarcasm identification.

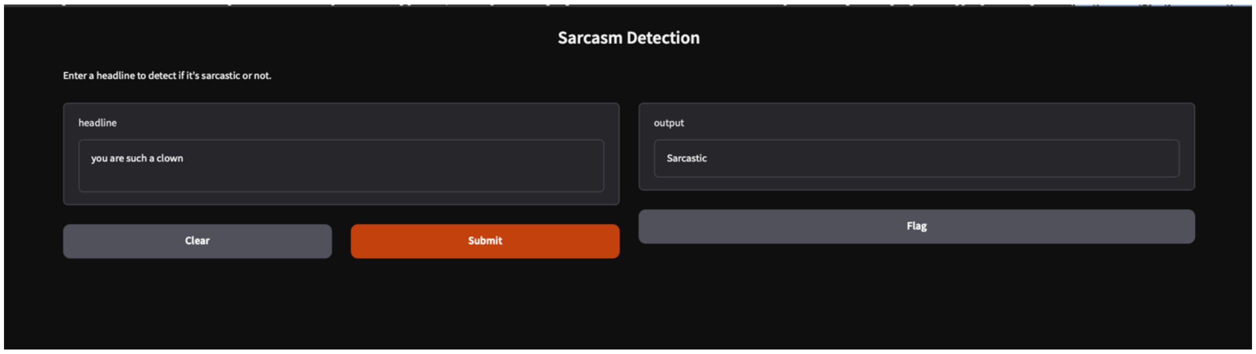


Figure 12: Prediction GUI

7. Conclusion and Future Work:

The research initiative embarked on an exploration to evaluate the performance of three deep learning models — Simple RNN, standard LSTM, and an LSTM enhanced with Word2Vec embeddings — in their ability to detect sarcasm within news headlines. Through our investigation we discovered practical insights into each model type's strengths and limitations while examining how preprocessing methods, architectural choices, and embedding strategies interact to influence performance.

The Simple RNN model achieved a peak performance with an accuracy rate of 98. 43% which exceeded LSTM's 91. 60% and LSTM with Word2Vec embeddings' 86. Performance metrics indicate a 77% effectiveness rate. The outcome contests the widespread belief that advanced models with greater depth consistently produce superior results. The combination of short headlines and an aggressive preprocessing system allowed the RNN to quickly detect patterns without using long-term memory processes.

Throughout its operation the LSTM model exhibited robust performance levels which remained steady while showcasing both its generalization abilities and training stability. The LSTM with pre-trained embeddings performed poorly despite its theoretical advantages. The generalization capabilities of frozen Word2Vec embeddings remain restricted when applied to tone-sensitive tasks like sarcasm detection after training on domain-incompatible corpora.
The experiment verifies that aligning model architecture with dataset characteristics and domain-specific challenges is crucial. This research illustrates how carefully designed input data can create scenarios where simpler models outperform complex solutions.
Numerous avenues for future investigation present themselves as potential research pathways.

Adjusting Word2Vec embeddings through fine-tuning rather than maintaining their static state allows for better adaptation to sarcasm-specific vocabulary.

The study of contextualized embeddings such as BERT and DistilBERT enables researchers to enhance the detection of irony and tonal variations alongside subtle meanings.

The integration of attention mechanisms empowers the model to focus on specific words which establish the sarcastic tone.

Developing a complex real-time interface using platforms such as Gradio or Streamlit would allow users to test the model with their own inputs for practical applications.

Implementing broad multi-domain datasets would improve generalizability and establish more robust benchmarking criteria.

Through this project a premier sarcasm detection system emerged while it simultaneously delivered essential insights into model simplicity and data preparation together with embedding selection techniques. The collected data points emerge as potential bases for subsequent exploration in sentiment analysis and tone classification across diverse natural language processing applications.

References

Timilsina, S., Gautam, M. & Bhattarai, B., 2022. *NepBERTa: Nepali Language Model Trained in a Large Corpus.* s.l., Association for Computational Linguistics.

Devlin, J., 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arxiv,* 2(5), p. 11.

Howard, J. and Ruder, S., 2018. *Universal language model fine-tuning for text classification*. In: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL), pp.328–339.

Tay, Y., Tuan, L.A. and Hui, S.C., 2018. *Reasoning with sarcasm by reading in-between*. In: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL), pp.1010–1020.

Zhang, Y., Sun, A., Rao, N. and Liu, T., 2020. *Deep learning-based sentiment classification: A survey*. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 10(4), p.e1333. https://doi.org/10.1002/widm.1333

Sun, C., Qiu, X. and Huang, X., 2019. *How to fine-tune BERT for text classification?* In: China National Conference on Chinese Computational Linguistics. Springer, Cham, pp.194–206.

Mishra, A., Dey, K., Shah, R.R. and Zimmermann, R., 2019. *A modular approach to cross-modal sarcasm detection*. In: Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pp.3797–3802.

Wang, H., Yang, Y., Wei, F. and Chang, B., 2020. *Infusing word embeddings with knowledge for sarcasm detection*. Information Processing & Management, 57(6), p.102330. https://doi.org/10.1016/j.ipm.2020.102330

Git Link

Click here to load the Git repo containing the code of Assessment Two.