Clickbait Spoiler Type Classification and Spoiler Generation

Sushilkumar Yadav
University of Waterloo
s3yadav@uwaterloo.ca

Abstract

This paper presents our approach to address the SemEval 2023 clickbait spoiling task, which comprises two distinct sub-tasks: spoiler classification and spoiler generation. For the classification task, involving the identification of different spoiler types, such as phrase, multi-sentence, and passage spoilers, we explore a diverse array of machine learning, deep learning, and transformer models. To assess the performance of our approach, we adopt the F1 score, a widely used metric for classification tasks.

Regarding the spoiler generation task, our primary objective is to generate spoilers from article texts using clickbait titles. To accomplish this, we leverage pretrained models, including BART(Lewis et al., 2019), T5(Raffel et al., 2020), and Pegasus(Zhang et al., 2020), fine-tuned specifically for the spoiler generation task. It is noteworthy that these pretrained models were primarily trained for summarization tasks(HuggingFace). The efficacy of our approach is evaluated using rogue and meteor evaluation metrics, commonly used to assess the quality of generated text.

1 Introduction

Clickbait articles have become ubiquitous in online media, characterized by sensational headlines designed to attract readers' attention and entice them into clicking on the article. While these clickbait titles may increase traffic to websites, they often result in frustrating user experiences when readers accidentally encounter plot-revealing spoilers before they can watch or read the content themselves. Spoilers can significantly diminish the enjoyment of movies, TV shows, books, and other media, making their identification and mitigation crucial for a positive user experience.

The SemEval (Semantic Evaluation) workshop has been a driving force in fostering research on various natural language processing (NLP) tasks through shared tasks and challenges. One of the latest SemEval tasks, the SemEval 2023 clickbait spoiling task, focuses on addressing the clickbait spoiler issue. This task encompasses two key challenges: spoiler classification and spoiler generation(SemEval).

1.1 Task Overview

The SemEval 2023(SemEval) clickbait spoiling task requires participants to develop innovative approaches to tackle two primary sub-tasks: spoiler classification and spoiler generation. The objective of the classification task is to identify different types of spoilers, including phrase spoilers, multi-sentence spoilers, and passage spoilers, in order to understand the nature and extent of the plot revelations within an article. On the other hand, the generation task involves producing spoilers from the article content based on the given clickbait titles.

For spoiler classification, we explored a diverse range of machine learning, deep learning, and transformer models for spoiler classification. Leveraging the latest advancements in NLP and model architectures, we aim to develop an efficient and accurate system capable of discerning different spoiler types within articles.

To achieve spoiler generation, we employ various pretrained models, including BART(Lewis et al., 2019), T5(Raffel et al., 2020), and Pegasus(Zhang et al., 2020), that have been fine-tuned specifically for the spoiler generation task. These models, originally designed for summarization tasks, have demonstrated promising capabilities in generating coherent and informative text.

In the subsequent sections of this paper, we provide detailed descriptions of our methodology for each sub-task, including data preprocessing, model architectures, fine-tuning procedures, and hyperparameter optimization. Furthermore, we present the experimental results, performance analysis, and insights gained from our approach.

The remainder of this paper is organized as follows: Section 2 discusses related works and the existing literature on clickbait and spoiler detection. Section 3 discusses about the dataset used and task descriptions from the SemEval 2023 competitions. Section 4 outlines the methodology adopted for the spoiler classification task, while Section 5 presents the approach for spoiler generation. Section 6 details the experimental setup. Section 7 presents the results and analysis of our approach, followed by a discussion. Finally, Section 8 concludes the paper with a summary of our contributions and potential directions for future research.

2 Related Work

2.1 Clickbait and Spoiler Detection

Early research focused on traditional text-based features and rule-based methods to identify clickbait headlines based on linguistic patterns and characteristics. These approaches often relied on the presence of sensational words, exaggerated language, and question-style titles commonly found in clickbait articles.

As NLP and machine learning advanced, researchers turned to supervised learning techniques using labeled datasets to build classifiers for clickbait detection. Features such as n-grams, part-of-speech tags, and sentiment analysis were employed to distinguish clickbait from non-clickbait headlines. Moreover, deep learning models like recurrent neural networks (RNNs) and convolutional neural networks (CNNs) have been applied to capture more intricate linguistic patterns and semantic representations for improved clickbait detection.

Similar efforts were made to address the issue of spoiler detection. Existing works explored various methods, including keyword matching, sentence similarity, and named entity recognition, to identify potential spoilers within articles and reviews. However, these approaches often struggled with nuanced spoilers that were not explicitly stated but implied within the text.

2.2 SemEval 2023 official competition Literature Review(Fröbe et al., 2023)

The paper compares various teams' submissions for the clickbait spoiling task at SemEval 2023.

For Task 1, which involves spoiler type prediction, the authors employ balanced accuracy as the main measure of effectiveness. The baseline model performs well, and only one team, Billy Batson, slightly outperforms the baseline in balanced accuracy. However, the difference is not significant. The overall best accuracy achieved is 0.74, indicating room for further improvement. The precision and recall scores for different spoiler types suggest that different approaches may complement each other well, offering potential directions for future work.

In Task 2, which focuses on spoiler generation, the authors evaluate submissions using BLEU-4, BERTScore, and METEOR metrics. John King achieves the highest overall effectiveness (BLEU-4 of 0.48) using a generative sequence-to-sequence approach.

The summary table for the teams that participated in the 2023 SemEval challenge for Task 1 is presented in Table 11, and for Task 2, it is provided in Table 10. Refer Appendices to see the summary table.

3 Dataset and Task Description

The Clickbait Challenge 2023 focuses on the task of clickbait spoiling, where the goal is to generate short texts that satisfy the curiosity induced by a clickbait post. The challenge features two subtasks: Task 1 involves classifying the spoiler type needed, and Task 2 involves generating the spoiler.

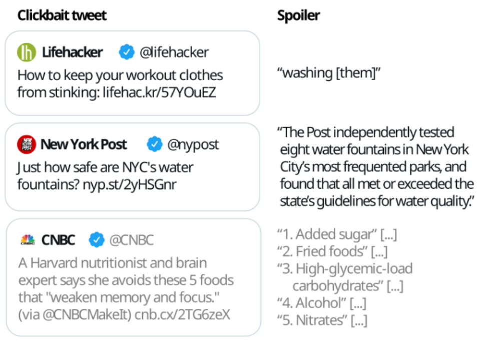
3.1 Dataset desciption

The dataset provided for the challenge contains clickbait posts and manually cleaned versions of the linked documents, along with extracted spoilers for each clickbait post. The spoilers are categorized into three types: short phrase spoilers, longer passage spoilers, and multiple non-consecutive pieces of text.

The training dataset (`training.jsonl`) consists of 3,200 clickbait posts, and the validation dataset (`val.jsonl`) contains 400 clickbait posts. After training and validation, systems are evaluated on 400 test posts.

The data is provided in JSON Lines format (`.jsonl`), where each line represents an entry with the following fields:

The dataset contains the following fields: **uuid**, a unique identifier for each dataset entry; **postText**: the clickbait post that requires "spoiling"; **targetParagraphs**: manually extracted main content paragraphs from the linked web page for both Task 1 and Task 2; **targetTitle**: the title of the linked



Clickbait tweet | Spoiler

Lifehacker @lifehacker
How to keep your workout clothes from stinking: lifehac.kr/57YOuEZ
"washing [them]"

New York Post @nypost
Just how safe are NYC's water fountains? nyp.st/2yHSGnr
"The Post independently tested eight water fountains in New York City's most frequented parks, and found that all met or exceeded the state's guidelines for water quality."

CNBC @CNBC
A Harvard nutritionist and brain expert says she avoids these 5 foods that "weaken memory and focus." (via @CNBCMakeIt) cnb.cx/2TG6zeX
"1. Added sugar" [...]
"2. Fried foods" [...]
"3. High-glycemic-load carbohydrates" [...]
"4. Alcohol" [...]
"5. Nitrates" [...]

Figure 1: Clickbait tweets and spoilers from the linked web pages (phrase, passage, and multipart spoiler)

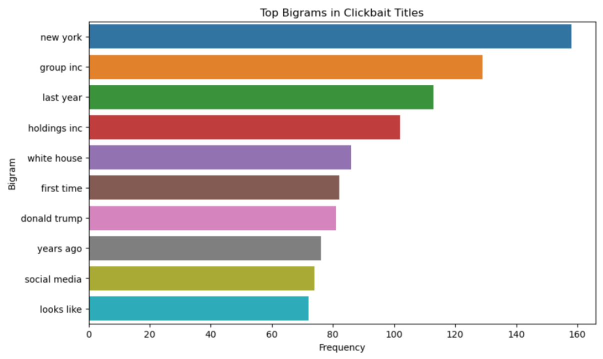


Figure 2: Clickbait tweets and spoilers from the linked web pages (phrase, passage, and multipart spoiler)

web page used in Task 1 and Task 2; **targetUrl**: the URL of the linked web page as a reference; **humanSpoiler** and `spoiler`: which are human-generated and extracted spoilers for the clickbait post, respectively, available in the training and validation datasets; **spoilerPositions**: providing position information of extracted spoilers, available in the training and validation datasets; and **tags**: indicating the spoiler type (e.g., "phrase," "passage," or "multi") for Task 1, available in the training and validation datasets.

The sample dataset entry can be found in Appendices Table 12

3.2 Task 1 on Spoiler Type Classification

The input is the clickbait post and the linked document. The task is to classify the spoiler type that the clickbait post warrants (either "phrase", "passage", "multi"). For each input, an output like {"id": "<ID>", "spoilerType": "<SPOILER-TYPE>"} has to be generated where <SPOILER-TYPE> is either phrase, passage, or multi.

3.3 Task 2 on Spoiler Generation

The input is the clickbait post and the linked document (and, optional, the spoiler type if your approach uses this field). The task is to generate the spoiler for the clickbait post. For each input, an output like {"uuid": "<UUID>", "spoiler": "<SPOILER>"} has to be generated where <SPOILER> is the spoiler for the clickbait post.

3.4 Data Analysis

We performed data analysis on the clickbait data for Task 1, which involved Spoiler Type Classification. We first checked the dataset's balance for all the classification classes and obtained the following output: `['passage', 'phrase', 'multi']`.

The count of occurrences for each unique label in the 'labels' column are as follows: 'phrase': **1367**, 'passage': **1274**, 'multi': **559**.

This analysis indicates that the dataset is not perfectly balanced for all classification classes. The 'phrase' label has the highest count, followed by 'passage' and 'multi'.

To capture more context in the language patterns, we conducted a bi-gram analysis. Bigram analysis can be found in fig 2. The bigram analysis reveals common word pairs frequently used in clickbait titles, including references to locations ("new york"), famous people names, etc. These bigrams are likely employed to increase the appeal and curiosity of the headlines to potential readers.

We utilized named entity recognition techniques to identify and extract named entities from the clickbait posts and linked documents. NER analysis can be found in fig 3. In the NER analysis, we observed that the majority of the content in the clickbait titles revolves around named entities related to people, organizations, and dates. By focusing on prominent figures, well-known organizations, and significant dates, clickbait creators aim to increase the click-through rates and engagement with their content.

3.5 Data Preprocessing for Task 1

For task 1, we performed some preprocessing steps that would in standardizing the data and improving the quality of the input before feeding it to the machine learning models. we also converted the labels into categorical format. or the data preprocessing, we applied tokenization, lowercase conversion, punctuation removal, number handling, stopword removal, space removal, and lemmatization.

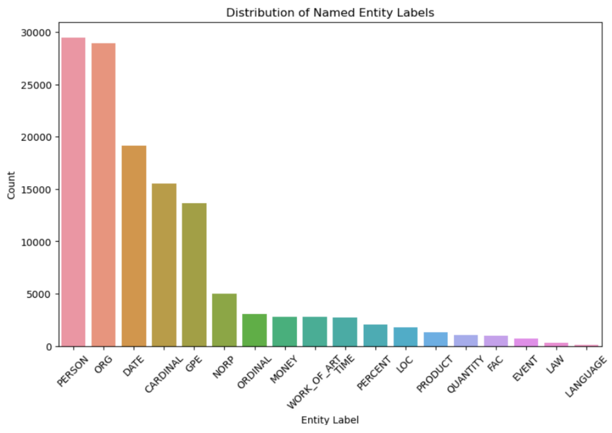


Figure 3: Clickbait tweets and spoilers from the linked web pages (phrase, passage, and multipart spoiler)

3.6 Data Preprocessing for Task 2

For task 2, since we are using the pretrained transformer models from the hugging face library. we performed basic preprocessing such as character removal, etc. we then converted the dataset into the required hugging face dataset form i.e. datasetdict using Datasets library of the hugging face.

4 Methodology

4.1 Task 1: Spoiler type detection

For classification task, we aimed to explore the effectiveness of classical machine learning algorithms, deep learning and pretrained LLMs such as BERT. we used TFIDF vectorizer to transform the text into a meaningful representation of numbers before fitting the data into machine learning models. TFIDF technique assigns weights to individual words based on their frequency in a document and inverse frequency across the entire dataset.

LLM Approach using BERT: BERT (Bidirectional Encoder Representations from Transformers) (Devlin et al., 2019) is a state-of-the-art pretrained language model based on the Transformer architecture. The Transformer model is a deep learning model introduced by Vaswani et al. in 2017(Vaswani et al., 2023). It leverages self-attention mechanisms to effectively capture long-range dependencies in sequential data, making it highly suitable for natural language processing tasks.

As depicted in Figure 4, the Transformer model consists of an encoder-decoder architecture. However, for our Spoiler Type Classification task, we focus on the encoder part only. The encoder is composed of multiple layers of self-attention and feed-forward neural networks.

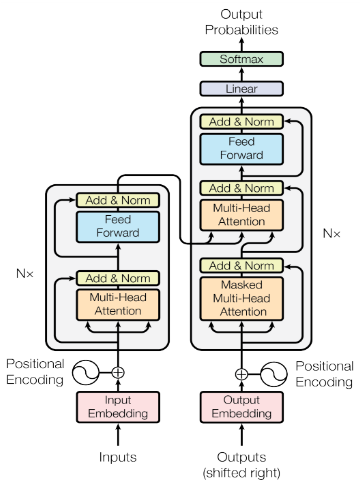


Figure 4: Transformer Model Architecture

BERT, being a pre-trained model, is first initialized with weights learned from a large corpus, such as Wikipedia text. It is then fine-tuned on our specific clickbait dataset to adapt to our classification task. Fine-tuning involves training BERT on our labeled data while updating its weights to make it perform well on the Spoiler Type Classification task.

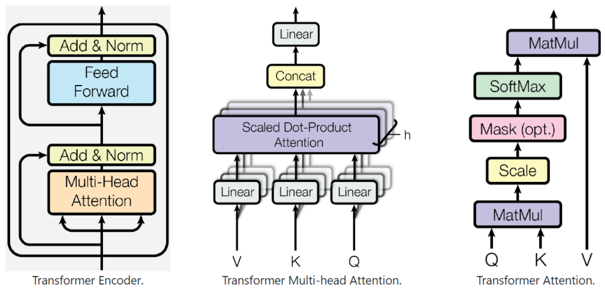


Figure 5: BERT Model Architecture

Figure 5 illustrates the architecture of BERT. It uses a bidirectional approach to understand the context of each word by considering both left and right contexts. The input text is tokenized into WordPiece tokens, and BERT processes these tokens through several layers of self-attention and feed-forward networks. The model outputs contextual embeddings for each token, which are further used for classification.

Evaluation Metric (Task 1):

To evaluate the performance of each algorithm for Spoiler Type Classification, we employed the F1 score as the primary evaluation metric. The F1 score is a commonly used metric for binary and multi-class classification tasks, combining precision and recall into a single value.

The precision and recall for each class (i.e., 'phrase', 'passage', 'multi') are calculated as follows:

$$\text{Precision} = \frac{\text{TruePositives}}{\text{TruePositives} + \text{FalsePositives}} \tag{1}$$

$$\text{Recall} = \frac{\text{TruePositives}}{\text{TruePositives} + \text{FalseNegatives}} \tag{2}$$

The F1 score is then given by the harmonic mean of precision and recall:

$$\text{F1Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{3}$$

The F1 score provides a balanced measure of the classifier's performance, taking into account both false positives and false negatives. It is especially useful when dealing with imbalanced classes, as it offers a fair assessment of the model's effectiveness across different categories.

4.2 Task 2: Spoiler generation

For generating spoilers in Task 2, we utilized pretrained transformer models from Hugging Face and fine-tuned them on the clickbait dataset. Given the size of humungous models, we cannot train such large LLM models from scratch considering the amount of computation and data required. for that reason, we implemented the Transfer learning approach. Specifically, we employed the following transformer models for spoiler generation: **T5-small**, **T5-base**, **BART_base**, **PEGASUS**

T5 (Text-to-Text Transfer Transformer):(Raffel et al., 2020) T5 is a state-of-the-art transformer model that is trained using a text-to-text transfer learning framework. It can handle a wide range of natural language processing tasks by casting them into a unified text-to-text format. The original T5 model is trained on C4 (Colossal Clean Crawled Corpus) dataset achieves state-of-the-art results on many NLP benchmarks while being flexible enough to be fine-tuned to a variety of important downstream tasks. The list of different variants of T5 models can be found in the table 1. T5 model structure consists of just a standard sort of vanilla encoder-decoder transformer 6. The details about *dmodel* (i.e., the dimension of the model), *dkv* (dimension of key-value pair), and *dff* (dimension of feed-forward layer) can be found in the "Attention is All You Need" paper by Vaswani et al. In the original Transformer model, *dmodel* is set to 512, *dkv* is set to 64, and *dff* is set to 2048. These dimensions play a crucial role in determining the performance and size of the Transformer-based models, including the T5 model. example of t5 model can be found in figure 7.

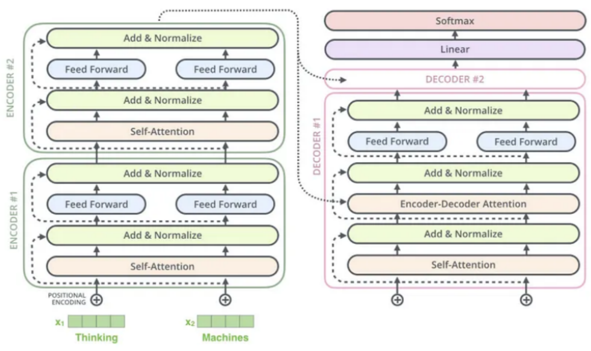


Figure 6: T5 Model Architecture

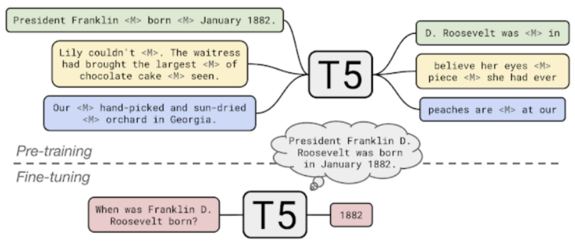

President Franklin <M> born <M> January 1882.
Lily couldn't <M>. The waitress had brought the largest <M> of chocolate cake <M> seen.
Our <M> hand-picked and sun-dried <M> orchard in Georgia.
T5
D. Roosevelt was <M> in
believe her eyes <M> piece <M> she had ever
peaches are <M> at our
Pre-training
Fine-tuning
President Franklin D. Roosevelt was born in January 1882.
When was Franklin D. Roosevelt born?
T5
1882

Figure 7: T5 Example

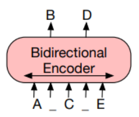
Table 1: T5 Model Size Variants

Model	Parameters	# Layers	dmodel	dff	dkv	# heads
t5-small	60M	6	512	2048	64	8
t5-base	220M	12	768	3072	64	12
t5-large	770M	24	1024	4096	64	16
t5-3b	3B	24	1024	16384	128	32
t5-11b	11B	24	1024	65536	128	128

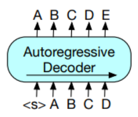
BART (Bidirectional and Auto-Regressive Transformers) (Lewis et al., 2019): BART is a denoising autoencoder that utilizes bidirectional encoder and auto-regressive decoder transformers. It is pretrained on a large corpus of text using denoising objectives and is known for its ability to generate coherent and fluent text. BART is particularly effective for text generation tasks and has shown promising results in various natural language generation tasks. Table 2 shows different size variants of the BART model, which is a variant of the Transformer architecture. The models are fine-tuned for specific tasks and datasets. BART outperforms

RoBERTa in several fine-tuning tasks. The number of parameters of the latter is mentioned in table 4

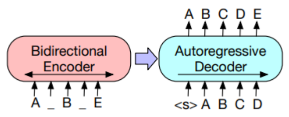
BART uses a standard Transformer-based neural machine translation architecture which, despite its simplicity, can be seen as generalizing BERT (due to the bidirectional encoder), GPT (with the left-to-right decoder), and many other more recent pretraining schemes (see figure 8. In total, BART contains roughly 10% more parameters than the equivalently sized BERT model s (see figure 3.



(a) BERT: Random tokens are replaced with masks, and the document is encoded bidirectionally. Missing tokens are predicted independently, so BERT cannot easily be used for generation.



(b) GPT: Tokens are predicted auto-regressively, meaning GPT can be used for generation. However words can only condition on leftward context, so it cannot learn bidirectional interactions.



(c) BART: Inputs to the encoder need not be aligned with decoder outputs, allowing arbitary noise transformations. Here, a document has been corrupted by replacing spans of text with mask symbols. The corrupted document (left) is encoded with a bidirectional model, and then the likelihood of the original document (right) is calculated with an autoregressive decoder. For fine-tuning, an uncorrupted document is input to both the encoder and decoder, and we use representations from the final hidden state of the decoder.

Figure 8: Comparison of Bart with Bert and GPT

Table 2: BART Model Size Variants

Model	Descriptions	# Params
BART-base	with 6 encoders and decoder layers	140M
BART-large	with 12 encoder and decoder layers	400M
BART-large-mnli	finetuned on MNLI	400M
BART-large-cnn	finetuned on CNN/DailyMail	400M
BART-large-xsum	finetuned on XSum	400M

Table 3: BERT Model Size Variants

Model	Description	# Params
BERT-Base	12 encoders with 12 bidirectional self-attention heads	110M
BERT-Large	24 encoders with 16 bidirectional self-attention heads	340M

PEGASUS (Zhang et al., 2020): PEGASUS is a transformer-based sequence-to-sequence model pretrained on a large corpus of text from the web. It is specifically designed for abstractive text summarization tasks, making it suitable for spoiler generation in our context. However, pegasus requires huge computation to fine tune it. different pegasus models are comapred in the table 5 where L (Layers), H (Hidden size), F (Feed-Forward Size), A (Attention Heads), # layer (Layers of Encoders and Decoders)

Evaluation Metrics (Task 2):

For evaluating the generated spoilers, we employed two widely used evaluation metrics:

ROUGE (Recall-Oriented Understudy for Gisting Evaluation)(Lin, 2004): ROUGE measures the similarity between the generated text and the reference text based on n-gram overlap. It includes ROUGE-1, ROUGE-2, and ROUGE-L, which evaluate unigrams, bigrams, and the longest common subsequences, respectively. The ROUGE score is computed as follows:

$$\text{ROUGE} = \frac{\text{Exact matches} + \text{Paraphrases} + \text{Word order matches}}{\text{Total number of words in reference text}}$$

METEOR (Metric for Evaluation of Translation with Explicit Ordering) (Banerjee and Lavie, 2005): METEOR is an automatic metric that considers exact matches, paraphrases, and word order similarity. It provides a comprehensive evaluation of the generated text's quality compared to the reference text. The METEOR score is calculated as follows:

$$\text{METEOR\_score} = \frac{\text{Exact matches} + \text{Paraphrases} + \text{Word order similarity}}{\text{Total number of words in reference text}}$$

Table 4: RoBERTa Model Size Variants

Model	# Params
roberta-base	125M
roberta-large	355M
roberta.large.mnli	355M
roberta.large.wsc	355M

Table 5: PEGASUS Model Size Variants

Model	L	H	F	A	Batch Size	# layers	# Parameters
Pegasus-Base	12	768	3072	12	256	6	223M
Pegasus-Large	16	1024	4096	16	8192	12	568M

5 Experimental Setup

5.1 Task1 experimental setup

For the experiments, we used a Kaggle notebook equipped with a T4x2 GPU having 16GB graphic memory, 13GB RAM, and a disk size of 73GB. The experimental setup involved data preprocessing and the implementation of various machine learning, deep learning and transformer models to classify spoilers into three classes.

Data Preprocessing We followed the data preprocessing steps mentioned earlier, which involved tokenizing the clickbait titles, converting them to lowercase, removing punctuation, numbers, and spaces, and lemmatizing the words to bring them to their base form.

Machine Learning Models We employed several machine learning algorithms such as MNB, SVM, Random Forest, Ensemble techniques.

Deep Learning Model We employed several machine learning models for the spoiler type classification task. The Convolutional Neural Network (CNN) used an embedding layer followed by a 1D convolutional layer with ReLU activation. The output was then pooled using a global max pooling layer and passed through dense layers with ReLU activation to produce a softmax output layer for classification.

The Long Short-Term Memory (LSTM) model also used an embedding layer, followed by an LSTM layer to capture sequential information from the text. The LSTM output was fed into dense layers with ReLU activation for classification.

The Gated Recurrent Unit (GRU) model had a similar architecture to the LSTM model but used a different type of recurrent unit to capture sequential patterns in the data.

Transformer Model We utilized a transformer-based model with BERT embedding and fine-tuned it using a feedforward bidirectional LSTM network. The transformer model processed the input text through a pre-trained BERT encoder to generate contextualized word embeddings. The embeddings were then passed through several bidirectional LSTM layers to capture contextual information from both directions. The final output was obtained through dense layers with leaky ReLU activation, followed by a softmax layer for multi-class classification.

Training Configuration For all models, we used the ADAM optimizer with categorical cross-entropy as the loss function. The dataset was split into training and validation sets for model training and evaluation.

5.2 Task 2 Experimental Setup

For Task 2, we used the pre-trained transformer models from Hugging Face's "transformers" library. We imported the required libraries needed to run the models from Hugging Face. We first loaded the JSON dataset and converted it into the required format needed for the model. We also used the ROUGE and METEOR metrics from the Hugging Face datasets library for evaluating the generated spoilers.

The corpus was tokenized using the AutoTokenizer module from the transformers library. Below are the configurations for fine-tuning the models on our dataset:

For T5-small, T5-base, and bart_base, we used a batch size of 4 and gradient accumulation steps of 4. The use of gradient accumulation effectively increased the batch size, allowing for better optimization on the limited GPU memory.

For Pegasus, due to its computational complexity, we used a batch size of 1 and gradient accumulation steps of 1. Additionally, we trained Pegasus for only 5 epochs, whereas the other models were trained for 25 epochs.

We integrated the "wandb" (Weights & Biases) APIs to capture training and validation visualizations during runtime, facilitating performance monitoring and analysis.

We trained the models with the following **hyper-parameters:** a batch size of 4 (except for T5-small, which used a batch size of 16), gradient accumulation steps of 4 to increase the effective batch size, a learning rate of 2e-5 (or any appropriate value for the specific model), a weight decay of 0.01, 5 training epochs for Pegasus and 25 for other models, enabled sequence generation during evaluation using the predict_with_generate flag, utilized fp16 (mixed precision training) for faster training on supported hardware, and set save_total_limit to 3 to save the three best checkpoints during training.

The fine-tuned models were then evaluated using the ROUGE and METEOR metrics to assess their performance in generating spoilers.

6 Results

6.1 Task 1 Results

For machine learning algorithms, we experimented with Random Forest Classifier, Logistic Regression, Support Vector Machine (SVM), Multinomial Naive Bayes (MNB), and an ensemble comprising Logistic Regression, SVM, and MNB. The achieved accuracies for these models are summarized in Table 6 where RF stands for Random Forest, LR stands for Logistic Regression, SVM stands for Support Vector Machine, MNB stands for Multinomial Naive Bayes. Additionally, we also evaluated different deep learning algorithms, including Convolutional Neural Network (CNN), Gated Recurrent Unit (GRU), and Long Short-Term Memory (LSTM), after training them for 10 epochs. The corresponding accuracies are listed in Table 7. Furthermore, we explored the use of Transformer models and recorded accuracy, precision, recall, and F1 Score on both training and validation sets, as shown in Table 8.

Based on the results, we can say that the Transformer models have shown superior performance compared to the traditional machine learning and deep learning models in our experiments. Given the data complexity, the machine and Deep learning models failed and transformers were able to achieve better results because of having self attention mechanism which are good at handling long-range dependencies and capturing complex contextual relationships. we achieved the F1 Score of 0.61140 on the test dataset.

Table 6: Accuracy comparison of different machine learning algorithms

Model	RF	LR	SVM	MNB	Ensemble (LR, SVM, MNB)
Accuracy	0.4775	0.4975	0.4975	0.4575	0.49

Table 7: Accuracy comparison of different deep learning algorithms (10 epochs)

Model	CNN	GRU	LSTM
Accuracy	0.4825	0.3650	0.4100

Table 8: Using Transformer model (5 epochs)

	Accuracy	Precision	Recall	F1 Score
Training	0.8884	0.842	0.8191	0.8304
Validation	0.76	0.6466	0.6175	0.632

6.2 Task 2 Results

The results for the Task 2 i.e. spoiler generation is shown in table 9. Among the models, T5-base achieved the highest ROUGE-1 F-measure of 0.4772, indicating its effectiveness in generating spoilers that align well with human-created spoilers. However, it obtained a relatively lower METEOR score of 0.1785, suggesting that there is still room for improvement in terms of capturing semantic similarity with the ground truth.

Bart-base, while scoring lower on ROUGE-1 F-measure with 0.4383, demonstrated a higher METEOR score of 0.3163. This suggests that Bart-base better captures the overall meaning and fluency of the generated spoilers, despite slightly deviating from the reference spoilers. for Pgasus-x-base model, we got the F1 Score of 0.262 just on 5 epochs which suggests that if trained for more number of epochs, it would yield better meteor score.

Table 9: Results on Task 2: Spoiler Generation

Model Name	METEOR	ROUGE-1
BART_base_25	0.3163	0.4383
Pegasus_x_base_5	0.262	0.3477
T5_base_25	0.1785	0.4772
T5_base_25	0.1298	0.3450

The training loss and METEOR score on the validation dataset are shown in Figure 9 and Figure 10, respectively. From the graphs, it can be observed that the T5-base model performs better on the validation dataset compared to the bart-base model. However, the T5-base model does not perform well on the test data, indicating a lack of generalization. This suggests that further fine-tuning is required to improve its performance on unseen data.

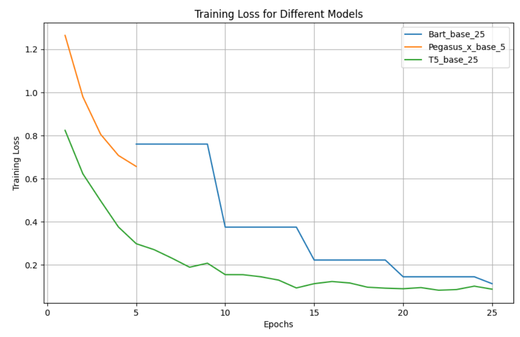


Figure 9: Training Loss of different transformer models for Task2

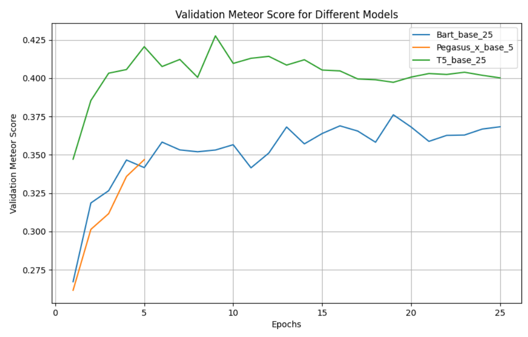
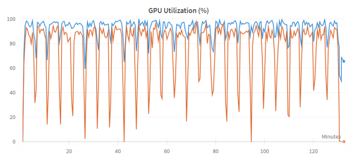
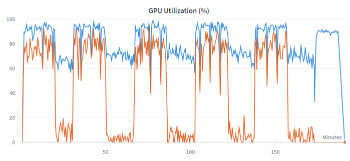


Figure 10: Meteor score of different transformers models for task 2

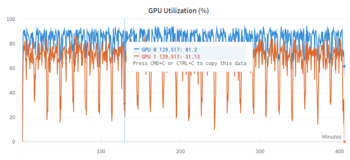
The time taken for training the model and its GPU utilization is shown in figure 11. The graph shows that the Bart-base model has taken less time to train as compared to pegasus-base and T5-base.



(a) Graph 1: Bart_base



(b) Graph 2: Pegasus_x_base



(c) Graph 3: T5_base

Figure 11: Performance comparison of three models

7 Conclusion and Future Scope

In this project, we conducted experiments utilizing various Language Models (LLMs) for spoiler type classification and spoiler generation tasks. Given the limited computation, we were able to build the classifier using the BERT embedding and then fine-tuning it using the the Bidirectional feed forward LSTM neural network. for spoiler generation task, we were able to generate the best meteor score using the Bart-Base model.

For the spoiler type classification task, further enhancements can be done by training BERT or ROBERTA models from scratch instead of solely relying on fine-tuning. Nonetheless, it is essential to consider computational resources before opting for training LLMs from scratch. To achieve the balance between performance and efficiency, we can try exploring the distill version of the LLMs.

For task 2, we obtained the best results using the Bert-base model with 25 epochs, but the results can still be improved. we could try training different models and then testing the results individually using those models and select the results that is best in all of the model. this can be achieved usign majority voting mechanism. we can also try using ensemble techniques to get the best spoiler generation. we should use other metrics such as Bertscore and Bleuscore for evaluating the spoiler generation.

Acknowledgements

I would like to extend my heartfelt gratitude to Professor Olga Vechtomova for her invaluable guidance and unwavering support throughout this course. I am also grateful to our Teaching Assistant Gaurav Sahu, whose continuous assistance and prompt responses to our queries have been immensely helpful.

8 References

References

Satanjeev Banerjee and Alon Lavie. 2005. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 65–72, Ann Arbor, Michigan. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding.

Maik Fröbe, Benno Stein, Tim Gollub, Matthias Hagen, and Martin Potthast. 2023. SemEval-2023 task 5: Clickbait spoiling. In *Proceedings of the The 17th International Workshop on Semantic Evaluation (SemEval-2023)*, pages 2275–2286, Toronto, Canada. Association for Computational Linguistics.

HuggingFace. Hugging_face. `https://huggingface.co/`. Accessed on July 25, 2023.

Vijayasaradhi Indurthi and Vasudeva Varma. 2023. Francis wilde at SemEval-2023 task 5: Clickbait spoiler type identification with transformers. In *Proceedings of the The 17th International Workshop on Semantic Evaluation (SemEval-2023)*, pages 1890–1893, Toronto, Canada. Association for Computational Linguistics.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer.

SemEval. Semeval. https://pan.webis.de/semeval23/pan23-web/clickbait-challenge.html. Accessed on July 25, 2023.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2023. Attention is all you need.

Sushilkumar Yadav. Github. https://github.com/sushilyadav1998/ClickBait. Accessed on July 30, 2023.

Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter J. Liu. 2020. Pegasus: Pre-training with extracted gap-sentences for abstractive summarization.

9 Appendices

Table 10: Summary of Teams' Submissions for Clickbait Spoiling Task 2

Team Name	**Algorithms Implemented**
Billie Newman	RoBERTa models (fine-tuned on SQuAD v2) with rule-based approach
Brooke English	DeBERTa models for spoiler generation
Diane Simmons	DeBERTa and BLOOM models for spoiler generation
Gallagher	T5, Long-T5, and Flan-T5 models in an ensemble
Jack Flood	RoBERTa model with passage retrieval approach
Jack Ryder	Zero-shot question-answering models as an ensemble
John Boy Walton	Ensemble of DistilBERT, BERT-base, and DeBERTa models
John King	Ensemble of T5 and Flan-T5 models
Morbo the Annihilator	RoBERTa-base models for phrase, passage, and multi-part spoilers
Monique Marmelstein	RoBERTa in a multi-task learning setup
Nancy Hicks Gribble	RoBERTa models pre-trained on SQuAD dataset
Sabrina Spellman	DeBERTa-base model with additional dataset for training
Walter Burns	Ensemble of RoBERTa and DeBERTa models

Table 11: Summary of Teams' Submissions for Clickbait Spoiling Task 1

Team Name	**Algorithm Implemented**
Alexander Knox	Few-shot learning with BLOOM (RoBERTa, ALBERT, DistilBERT)
Billie Newman	RoBERTa with additional features (Regular Expressions)
Billy Batson	DeBERTa-based type classification
Chick Adams	RoBERTa and DeBERTa models
Clark Kent	PIXEL model (Pre-trained vision transformer)
Mr. Fosdick	Transformer model and random forest (GPT3, DeBERTa)
Francis Wilde	RoBERTa for spoiler-type classification
Gallagher	T5, Long-T5, and Flan-T5 (T5-Large model)
Jack Flood	BiLSTM model (RoBERTa)
John Boy Walton	Ensemble of DistilBERT, BERT-base, and DeBERTa (DistilBERT, BERT-base, DeBERTa)
Machamp	Multi-task-learning model (Machamp framework)
Matt Bai	BERT-base for classification
Mr. Wallace	DistilBERT-base model
Monique Marmelstein	RoBERTa in multi-task learning
Morbo the Annihilator	Naive Bayes, Logistic Regression, and SVM (Term frequency, TF-IDF)
Nancy Hicks Gribble	RoBERTa with one-vs-rest models
Perry White	BERT, RoBERTa, and DeBERTa
Sam Miller	XLNET for classification
Stephen Colbert	DeBERTa-Large (version 3)
Walter Burns	Ensemble of RoBERTa models (RoBERTa)

Table 12: Sample Dataset Entry

Field	**Value**
uuid	0af11f6b-c889-4520-9372-66ba25cb7657
postText	Wes Welker Wanted Dinner With Tom Brady, But Patriots QB Had Better Idea
targetParagraphs	It'll be just like old times this weekend for Tom Brady and Wes Welker. Welker revealed Friday morning on a Miami radio station that he contacted Brady because he'll be in town for Sunday's game between the New England Patriots and Miami Dolphins at Gillette Stadium. It seemed like a perfect opportunity for the two to catch up. But Brady's definition of "catching up" involves far more than just a meal. In fact, it involves some literal "catching" as the Patriots quarterback looks to stay sharp during his four-game Deflategate suspension. "I hit him up to do dinner Saturday night. He's like, 'I'm going to be flying in from Ann Arbor later (after the Michigan-Colorado football game), but how about that morning we go throw?' " Welker said on WQAM, per The Boston Globe. "And I'm just sitting there, I'm like, 'I was just thinking about dinner, but yeah, sure. I'll get over there early and we can throw a little bit.' " Welker was one of Brady's favorite targets for six seasons from 2007 to 2012. It's understandable him and Brady want to meet with both being in the same area. But Brady typically is all business during football season. Welker probably should have known what he was getting into when reaching out to his buddy. "That's the only thing we really have planned," Welker said of his upcoming workout with Brady. "It's just funny. I'm sitting there trying to have dinner. 'Hey, get your ass up here and let's go throw.' I'm like, 'Aw jeez, man.' He's going to have me running like 2-minute drills in his backyard or something." Maybe Brady will put a good word in for Welker down in Foxboro if the former Patriots wide receiver impresses him enough.
targetTitle	Wes Welker Wanted Dinner With Tom Brady, But Patriots QB Had A Better Idea
targetUrl	`http://nesn.com/2016/09/wes-welker-wanted-dinner-with-\tom-brady-but-patriots-qb-had-\better-idea/`
humanSpoiler	They Threw A Football
spoiler	how about that morning we go throw?
spoilerPositions	[[3, 151], [3, 186]]
tags	passage