Mr-wallace at SemEval-2023 Task 5: Novel Clickbait Spoiling Algorithm Using Natural Language Processing

Vineet Saravanan
Cranbrook Schools
Bloomfield Hills, MI
vineetsaravanan@gmail.com

Steven Wilson
Oakland University
Rochester, MI
stevenwilson@oakland.edu

Abstract

Clickbait creates a nuisance in the online experience by creating a lure towards poor content in order to generate ad revenue. With the use of natural language processing models, we can save users time and reduce the need to follow clickbait links. Task 5 at SemEval-2023 focused on precisely this problem and was broken into two steps: identifying the clickbait spoiler type and then identifying the clickbait itself. Our approach involves the use of fine-tuned text classification and question-answering models. Our classification model is able to determine the type of clickbait with 65.3% accuracy. The question-answering model exactly spoiled clickbait generated around 42.5% of the time. Efforts toward solving this task may have an impact by helping to save users' time and quickly give an insight into the answer of what the clickbait/article is about.

1 Introduction

Clickbait, in social media, are exaggerated headlines whose main motive is to mislead the reader to "click" on them. They create a nuisance in the online experience by creating a lure towards poor content. Online content creators are utilizing more of them to get increased page views and, thereby, more ad revenue without providing the backing content (Agrawal, 2016). Clickbait articles usually appeal to a user's emotions and curiosity. These articles are often lacking in quality and information accuracy. Clickbait articles have a considerably higher click rate than non-clickbait articles. This has been a rising problem in the 21st century with the growth in internet usage and social media.

In SemEval-2023, the tasks of clickbait spoiling task is split into two subtasks, as shown in Figure 1, the first task is to identify the clickbait type, and the second task is to actually spoil the clickbait by identifying the sequence(s) of text in the source article that provide the information that the reader is being lured to read about given a particular clickbait post (Fröbe et al., 2023a). For Task 1, which is the Spoiler Type Classification, the input is the clickbait post and the linked document, and the task is to classify the spoiler type that the clickbait post warrants (either "phrase," "passage," or "multi"), which can be approached using a supervised text classification approach. For Task 2, which is on Spoiler Generation, the input is the clickbait post and the linked document. The task is to generate the spoiler for the clickbait post, which can be approached using an information extraction or question-answering approach where the question is in the title of the post. More information about the source datasets can be found in Figure 2.

We fine-tuned a DistilBERT (Sanh et al., 2019) model for task 1, which was one of the top ten in terms of precision for classifying the "multi" type of clickbait. The submission was made on the Tira platform (Fröbe et al., 2023b). While we were unable to make an official submission for task 2, we developed a BERT-based (Devlin et al., 2018) question-answering-based model for this task and evaluated it on our own held-out validation set.

2 Background

Previous work has been done using different Question Answering like abstractive QA and extractive QA, with abstractive QA models extracting the answer to a given question from within a provided context, and abstraction QA models which generate new text and do not require a labeled answer span (Johnson et al., 2022). In this paper, we will discuss and implement an abstractive QA model since this will give the user a more relevant answer which is guaranteed to be in the clickbait article. There have also been applications of passage retrieval models (Hagen et al., 2022b). Previous work has been done to check whether a certain tweet is clickbait or not. One such high-accuracy system utilizes a neural network architecture with bidirectional gated recur-

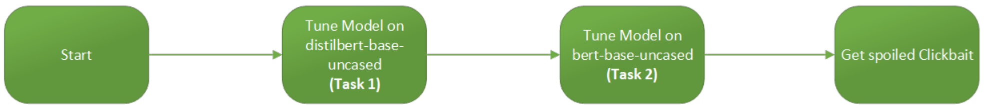
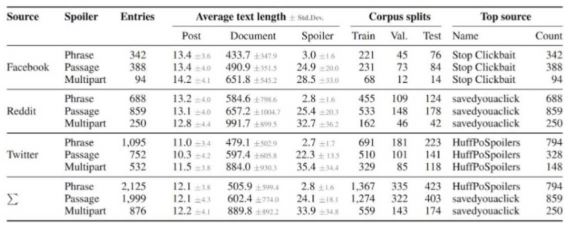

Start
Tune Model on distilbert-base-uncased (Task 1)
Tune Model on bert-base-uncased (Task 2)
Get spoiled Clickbait

Figure 1: Steps in the clickbait spoiling task.



Source	Spoiler	Entries	Average text length ± Std.Dev.			Corpus splits			Top source	
			Post	Document	Spoiler	Train	Val.	Test	Name	Count
Facebook	Phrase	342	13.4 ±3.6	433.7 ±347.9	3.0 ±1.6	221	45	76	Stop Clickbait	342
	Passage	388	13.4 ±4.0	490.9 ±351.5	24.9 ±20.0	231	73	84	Stop Clickbait	388
	Multipart	94	14.2 ±4.1	651.8 ±545.2	28.5 ±33.0	68	12	14	Stop Clickbait	94
Reddit	Phrase	688	13.2 ±4.0	584.6 ±798.6	2.8 ±1.6	455	109	124	savedyouaclick	688
	Passage	859	13.1 ±4.0	657.2 ±1004.7	25.4 ±20.3	533	148	178	savedyouaclick	859
	Multipart	250	12.8 ±4.4	991.7 ±899.5	32.7 ±36.2	162	46	42	savedyouaclick	250
Twitter	Phrase	1,095	11.0 ±3.4	479.1 ±502.9	2.7 ±1.7	691	181	223	HuffPoSpoilers	794
	Passage	752	10.3 ±4.2	597.4 ±605.8	22.3 ± 13.5	510	101	141	HuffPoSpoilers	328
	Multipart	532	11.5 ±3.8	884.0 ±930.3	35.4 ±34.4	329	85	118	HuffPoSpoilers	148
$\sum$	Phrase	2,125	12.1 ±3.8	505.9 ±599.4	2.8 ±1.6	1,367	335	423	HuffPoSpoilers	794
	Passage	1,999	12.1 ±4.3	602.4 ±774.0	24.1 ±18.1	1,274	322	403	savedyouaclick	859
	Multipart	876	12.2 ±4.1	889.8 ±892.2	33.9 ±34.8	559	143	174	savedyouaclick	250

Figure 2: Dataset statistics from Hagen et al. (2022a).

rent units (biGRU) and a self-attention mechanism to assess clickbait strength (Zhou, 2017; Potthast et al., 2018).

3 System Overview

In this section, we provide details about our approach.

3.1 Task 1

To read and utilize the data, specific fields have to go through preprocessing. The first is from the tag column. This column gives us the clickbait type as a string, which we convert to an integer value representing the class labels as 0, 1, and 2. We also combine many of the input fields that are available from the article itself (targetTitle, text, targetKeywords, and Provenance). The targetTitle is the title of the article, the text is the article itself, the targetKeywords are the keywords for the article itself, and the Provenance is the information about who spoiled the clickbait for the dataset. Also, we have to preprocess the text to add periods where it is needed. This was usually at the end of the article itself since there was sometimes punctuation missing at the end. After that, we concatenate all the fields together into one string to increase the simplicity of training the model. Next, we use the default tokenizer for the Distilbert-base-uncased, available from Hugging Face,[1] to tokenize the text. DistilBERT is a distilled version of BERT (Devlin et al., 2018), which means that it has been trained to achieve similar performance to BERT but with fewer parameters and faster training times, as it uses much more memory and computes efficiently to use. In addition to these efficiency concerns,

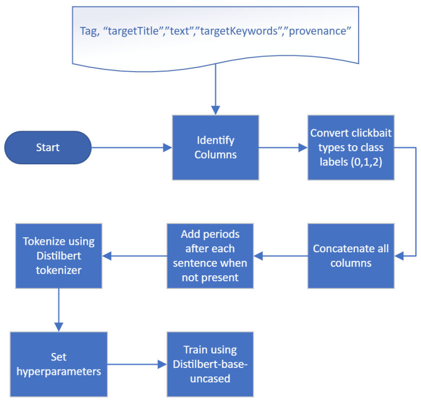

Tag, "targetTitle","text","targetKeywords","provenance"
Start
Identify Columns
Convert clickbait types to class labels (0,1,2)
Concatenate all columns
Add periods after each sentence when not present
Tokenize using Distilbert tokenizer
Set hyperparameters
Train using Distilbert-base-uncased

Figure 3: Task 1 Flow Chart for steps of preprocessing and training

pre-trained language models like DistilBERT have achieved strong performance on a variety of natural language processing tasks, including text classification. Therefore, we expect DistilBERT to yield high-quality results for this task. Memory efficiency is also helpful since we run our scripts on Google Colab,[2] where resources such as computational time and memory may be limited. Google Colab is a free cloud-based platform that allows users to code in Python notebooks. It is beneficial for small-scale natural language processing experiments since it provides access to resources such as GPUs and TPUs, which can speed up the training of large NLP models. For our training, We only utilized GPU and CPU backends. Figure 3 shows how many parameters DistilBERT uses compared to other well-known models.

We use the `TrainingArguments` and `Trainer` classes from the transformers library (Wolf et al., 2019) to train a model. The `TrainingArguments` object specifies various hyperparameters and settings for the training process, such as the output directory, learning rate, batch size, number of epochs, and others. The `Trainer` object is then created with the model, training arguments, training and evaluation datasets, tokenizer, data collator, and a function for computing evaluation metrics. The training method of the Trainer object is then called

[1] https://huggingface.co/

[2] https://colab.research.google.com/

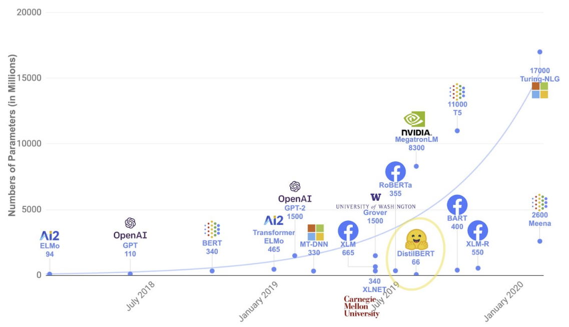


Figure 4: Number of parameters in NLP models over time from (Cistac et al., 2020). This figure emphasizes the relative size of the DistilBERT model that we use for Task 1.

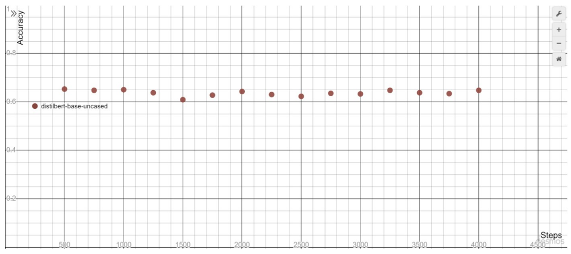


Figure 5: Development accuracy every 250 steps during training for clickbait classification.

to start the training process. The training process will involve iterating over the training dataset for a specified number of epochs, in which the model is presented with batches of examples, and the weights of the model are updated based on the predicted scores and the actual labels. The model's performance on the evaluation dataset will be evaluated using the `compute_metrics` function, the results will be logged, and the model achieving the highest value on the development set will be used. Figure 5 shows the accuracy plotted every 250 steps over 4000 training steps.

3.2 Task 2

For the second task, the data was pre-processed differently. We combine all the parts of the targetParagraph into one string as the question-answering models that we use expected a single sequence as input. We then get the first element of the list of postText. We also need to get the absolute positions of where the clickbait spoiler is relative to the entire targetParagraph. We do this by looping through the elements in the spoilerPositions and adding up the lengths of the paragraphs in targetParagraphs until it reaches the start and end positions of the spoiler.

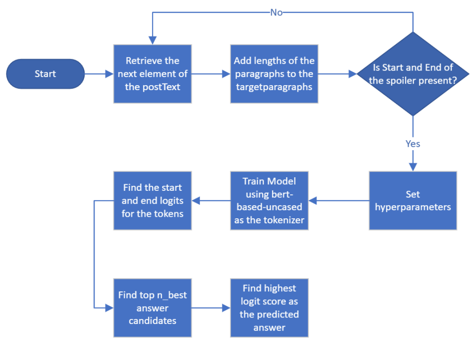
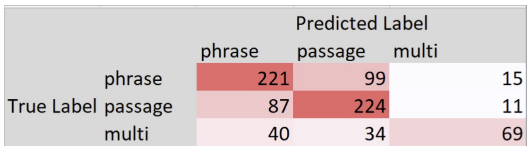

Start
Retrieve the next element of the postText
Add lengths of the paragraphs to the targetparagraphs
Is Start and End of the spoiler present?
No
Yes
Set hyperparameters
Train Model using bert-based-uncased as the tokenizer
Find the start and end logits for the tokens
Find top n_best answer candidates
Find highest logit score as the predicted answer

Figure 6: Task 2 Flow Chart for steps of preprocessing and training

The code then goes on to predict the answers for a given set of examples. We first set some hyperparameters, such as the maximum length of the input text and the stride of the tokenization process. To train the model, we have the context in the targetParagraph, the question in postText, and the answer in the spoiler. We then use the default tokenizer for bert-based-uncased as our tokenizer. We then iterate through the examples, finds the start and end logits for the tokens, and use these to find the top `n_best` answer candidates. It skips candidates that are not entirely within the context or have a length outside of a specified range. It then selects the answer with the highest logit score as the predicted answer and stores it in a list. Bert-based-uncased can be tuned for a question-answering model. There are several advantages to using BERT as a pre-trained model for a question-answering model. BERT has learned the English language's structure and patterns thanks to its pre-training on a sizable dataset. This pre-training enables BERT to perform well without the need for task-specific training data on a variety of NLP tasks. Since BERT is a transformer model, it can efficiently process lengthy text sequences. Due to the potential length and complexity of the questions and answers, this is crucial for question answering. Because of its bidirectional architecture, BERT can process text while taking into account previous and subsequent words, enabling it to build a representation of a word's meaning in the context of the entire sentence. When performing tasks like answering questions, where the distinction between upper and lower case can be significant, the "cased" version of BERT preserves the case of the words in the input text.



		Predicted Label		
		phrase	passage	multi
True Label	phrase	221	99	15
	passage	87	224	11
	multi	40	34	69

Figure 7: Confusion matrix for the clickbait type classification task

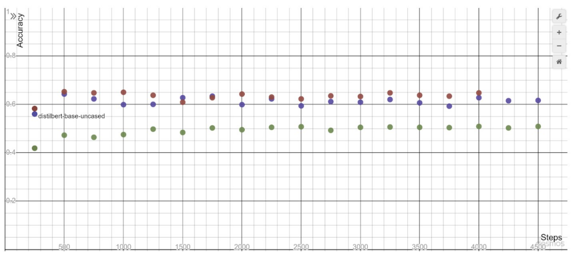


Figure 8: distilbert-base-uncased (red) vs bert-base-cased (purple) vs bert-base-NER (green)

4 Experimental Setup

4.1 Task 1

We calculate the accuracy of the model every 250 steps in the training process. We define a function `compute_metrics()` that takes in a list of tuples `eval_preds` and returns the accuracy of the predictions made. Predictions are determined by taking the argmax of the logits, which are output by the model. Finally, the function computes and returns the accuracy of the predictions using the compute method of the metric object, passing in the predictions and labels as arguments. Early on, we noticed that the model's accuracy, when graphed, was plateauing early on, and we didn't need to train the models past five epochs due to this.

4.2 Task 2

In general, using BERT as the base model for question-answering can give the model a solid foundation and boost performance. We tokenize the text by adding a `[CLS]` to the start, and `[SEP]` is used as a separator token. It is used at the end of the title and the end of the passage. This is used so the model can differentiate between the title and passage. The `preprocess_training_examples()` function takes a dictionary of training examples as input and returns a dictionary of preprocessed inputs that can be used by the model. The function then processes the spoiler and spoiler-Positions fields of the input examples to generate `start_positions` and `end_positions`, which are the start and end positions of the correct answer in the tokenized sequences. These start and end positions are appended to the input dictionary and returned by the pre-processing function. We train the model by mapping the raw dataset using a hugging face function and get the predicted spoilers returned.

5 Results

Our system performed in the top ten in terms of precision for the "multi" class classification. Figure 7 shows how misclassifications were made by our model. The "multi" classification was the classification in task 1, where the spoiler is a list of multiple items. We were only given the ranking for this specific classification.

While the model is training, our script outputs the accuracy rate every 250 steps, so we can see whether the model is overfitting or underfitting. Figure 8 is a comparison of distilbert-base-uncased vs. bert-base-uncased and bert-base-NER^3 models tuned on this task. BERT-based NER (Named Entity Recognition) is used to identify and classify named entities in text using BERT (Bidirectional Encoder Representations from Transformers) language model. Named entities can be people, organizations, locations, etc. From the figure, we can deduce that these types of models aren't the best for this specific task.

We use the F1 score for task 2 to improve upon our model. The F1 score is a metric for evaluating the performance of a model on a binary classification task. The harmonic mean of the model's precision and recall is used to calculate the F1 score. Precision is calculated as the model's total positive predictions divided by the number of true positive predictions. The recall is calculated by dividing the total number of positive examples in the dataset by the number of true positive predictions the model made. The F1 allowed us to see the overlap between the correct spoiler vs the generator spoiler. The F1 score was a better metric since it gave insight into the overall message/meaning of the spoiler instead of just word-for-word correctness. On the test split of the dataset, the model received an F1 score of 0.85, but generally, the results save users time and give an insight into the answer to what the clickbait/article is about.

^{3}https://huggingface.co/dslim/bert-base-NER

6 Conclusion

This paper presents a model for clickbait spoiling, which aims at generating short texts that satisfy the curiosity induced by a clickbait post. The model is split into two tasks: identifying the clickbait type and spoiling the clickbait. The first task is to classify the spoiler type that the clickbait post warrants, and the second task is to generate the spoiler for the clickbait post. The model utilizes the Distilbert-base-uncased model for the first task and the Bert-base-uncased model for the second task. The trained model is optimized through trial and error on different model selections, and hyperparameters and results are presented in a confusion matrix. From the figure 7, we were able to deduce that the phrase and passage classification were being misclassified as the other. There was also a low accuracy for the multi-classification, which could be attributed to the lower amount of data for the multi compared to the phrase or multi-classifications. This gave us insight into how to further tune the model. The main reason we utilized Distilbert-base-uncased is that it analyzes words in the context of whats around it. The objective of this model is to save readers time and spoil the clickbait of different articles they may see on different platforms like Twitter and Reddit.

References

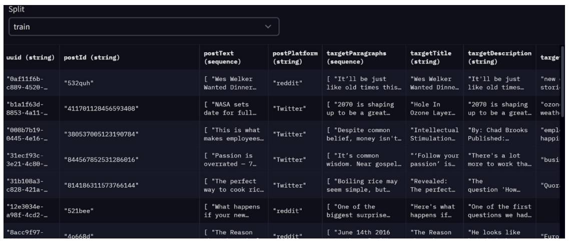
Amol Agrawal. 2016. Clickbait detection using deep learning. In *2016 2nd international conference on next generation computing technologies (NGCT)*, pages 268–272. IEEE.

Pierric Cistac, Anthony Moi, and Victor Sanh. 2020. How hugging face achieved a 2x performance boost for question answering with distilbert in node.js.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Maik Fröbe, Tim Gollub, Benno Stein, Matthias Hagen, and Martin Potthast. 2023a. SemEval-2023 Task 5: Clickbait Spoiling. In *17th International Workshop on Semantic Evaluation (SemEval-2023)*.

Maik Fröbe, Matti Wiegmann, Nikolay Kolyada, Bastian Grahm, Theresa Elstner, Frank Loebe, Matthias Hagen, Benno Stein, and Martin Potthast. 2023b. Continuous Integration for Reproducible Shared Tasks with TIRA.io. In *Advances in Information Retrieval. 45th European Conference on IR Research (ECIR 2023)*, Lecture Notes in Computer Science, Berlin Heidelberg New York. Springer.



Split

train

uuid (string)	postId (string)	postText (sequence)	postPlatform (string)	targetParagraphs (sequence)	targetTitle (string)	targetDescription (string)	targe
"0af11f6b-c889-4520-…	"532quh"	["Wes Welker Wanted Dinner…	"reddit"	["It'll be just like old times this…	"Wes Welker Wanted Dinne…	"It'll be just like old times…	"new stori
"b1a1f63d-8853-4a11-…	"411701128456593408"	["NASA sets date for full…	"Twitter"	["2070 is shaping up to be a great…	"Hole In Ozone Layer…	"2070 is shaping up to be a great…	"ozon weath
"008b7b19-0445-4e16-…	"380537005123190784"	["This is what makes employees…	"Twitter"	["Despite common belief, money isn't…	"Intellectual Stimulation…	"By: Chad Brooks Published:…	"empl happi
"31ecf93c-3e21-4c80-…	"844567852531286016"	["Passion is overrated — 7…	"Twitter"	["It's common wisdom. Near gospel…	"'Follow your passion' is…	"There's a lot more to work tha…	"busi
"31b108a3-c828-421a-…	"814186311573766144"	["The perfect way to cook ric…	"Twitter"	["Boiling rice may seem simple, but…	"Revealed: The perfect…	"The question 'How…	"Quor
"12e3034e-a98f-4cd2-…	"521bee"	["What happens if your new…	"reddit"	["One of the biggest surprise…	"Here's what happens if…	"One of the first questions we had…	
"8acc9f97-	"4o668d"	["The Reason	"reddit"	["June 14th 2016	"The Reason	"He looks like	"Euro

Figure 9: Example of the format in which we stored data in the Hugging Face datasets hub.

Matthias Hagen, Maik Fröbe, Artur Jurk, and Martin Potthast. 2022a. Clickbait Spoiling via Question Answering and Passage Retrieval. In *60th Annual Meeting of the Association for Computational Linguistics (ACL 2022)*, pages 7025–7036. Association for Computational Linguistics.

Matthias Hagen, Maik Fröbe, Artur Jurk, and Martin Potthast. 2022b. Clickbait spoiling via question answering and passage retrieval. *arXiv preprint arXiv:2203.10282*.

Oliver Johnson, Beicheng Lou, Janet Zhong, and Andrey Kurenkov. 2022. Saved you a click: Automatically answering clickbait titles. *arXiv preprint arXiv:2212.08196*.

Martin Potthast, Tim Gollub, Matthias Hagen, and Benno Stein. 2018. The clickbait challenge 2017: Towards a regression model for clickbait strength. *arXiv preprint arXiv:1812.10847*.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and Jamie Brew. 2019. Huggingface's transformers: State-of-the-art natural language processing. *CoRR*, abs/1910.03771.

Yiwei Zhou. 2017. Clickbait detection in tweets using self-attentive network. *arXiv preprint arXiv:1710.05364*.

A Data storage format

We used the Hugging Face datasets library [4] to host our training data while developing models. This made it easy to access and load the dataset from anywhere and use a common API to easily interface with other Hugging Face libraries. Figure 9 shows an example of some of the data in this format.

[4]https://huggingface.co/datasets