Unveiling Clickbait: A Study on Spoiler Type Classification and Generation

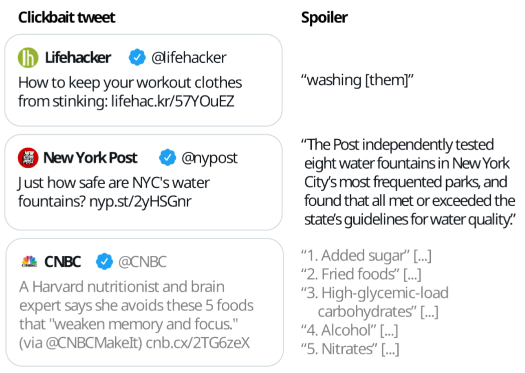
Hanwen Zhang

Abstract

In this study, I introduce and explore the concept of clickbait spoiling, which involves generating a concise text that satisfies the curiosity aroused by clickbait posts. Clickbait typically teases content to lure readers, rather than providing informative summaries. My research focuses on classifying the required type of spoiler (phrase or passage) and generating appropriate spoilers. Through a comprehensive evaluation and error analysis on a dataset of manually spoiled clickbait posts, our spoiler type classifier achieves an impressive accuracy of 69%. Notably, the question answering model RoBERTa-large outperforms all other models in generating spoilers for both types. These findings shed light on the effective use of natural language processing techniques for addressing clickbait spoiling tasks.

1 Introduction

Clickbait refers to social media posts designed to inappropriately lure readers to a web page. This is typically accomplished through sensationalist believed to create a 'curiosity gap'. Clickbait is often deemed inappropriate due to its typically mundane or trivial resolution, often consisting of nothing more than a phrase, a brief passage, or a list that could have been easily included in the post itself. This observation leads us to introduce the task of clickbait spoiling, which involves identifying or generating a spoiler for a clickbait post. Figure 1 shows how clickbait and its spoiler looks like.



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 and spoiler

This paper presents my exploration into the realm of clickbait spoiling and offers the following contributions:

1. There is a dataset provided by Webis Clickbait Spoiling Corpus 2022 (Webis-Clickbait-22), which contains a collection of 5,000 clickbait posts, their associated linked pages, and a spoiler text within them. Professor build a collection based on it which contains less data. I will use it as my dataset.

2. I will train many appropriate models to classify the dataset. The categories include phrase, passage, or multi. After that, I will discuss the performance of every models.

3. I will train multiple models for generating clickbait spoilers. Additionally, I will analyze the performance of each model.

2 Related Work

For the classification task, there are currently various methods available, including Classic Feature-based Models such as Naive Bayes(Murphy, 2006), Logistic Regression(Wright, 1995), and Neural Network models like CNN(Yunchao Wei, 2014) and BERT(Jacob Devlin, 2019). I will systematically try out each of these models and attempt to find the one that performs the best for the task at hand.

For the generation task, it can be viewed as a question-answering system. Currently, there are various models capable of implementing question-answering systems, such as BM25(Andrew Trotman, 2014), among others. I will attempt to try out multiple models of this kind. Then, I will experiment with using the large language model GPT(Luciano Floridi, 2020) and try to find the model that performs the best for the generation task.

3 Approach and Experiments: Spoiler Type Classification

My approach to clickbait spoiling centers around the recognition of three distinct spoiler types: phrase spoilers, passage spoilers, and multipart spoilers. We firmly believe that crafting specific strategies for each type of spoiler will yield the most effective results. Nevertheless, a crucial requirement for this endeavor is the accurate classification of clickbait. Consequently, our initial focus lies in exploring the predictability of a clickbait post's spoiler type.

To address the spoiler classification subtask, we conducted experiments using both classic feature-based models, including Naïve Bayes, and Logistic Regression, as well as neural models such as CNN, and BERT. By employing this diverse set of models, we aimed to explore the effectiveness and comparative performance of traditional feature-based approaches and neural models in classifying spoilers accurately. Each model brings its own strengths and capabilities, and our experimentation sought to determine which approach yields the best results for the specific task of spoiler classification.

3.1 Classic Feature-based Models

I primarily used the "postText" and "targetParagraph" elements from the training dataset. I preprocessed both elements by removing stopwords and non-alphabetic content from the strings, using the stopwords library from nltk. After preprocessing, I concatenated the processed strings together. When necessary, I employed techniques such as word2vec to convert the strings into vectors.

My first step is to evaluate the effectiveness of the preprocessing. So, I used a Naïve Bayes classifier to classify the data, both preprocessed and unprocessed. You can find my work at Table 1.

Based on the results, the preprocessing only provided a slight improvement.

I will need to test the impact of different classification methods on the model next. This is a three-class classification problem. I can approach it in two ways:

1. I can directly use the models to classify the data into three classes.

2. I can first divide the data into two groups, one for multi and others, and then further classify the others into either phrase or paragraph. This approach allows me to convert it into a series of binary classifications.

The advantage of the second method is that it simplifies the problem by breaking it down into a sequence of binary classification tasks. You can find the results in Table 2.

The conclusion is that there is not a significant difference in accuracy between the two methods.

I tried various classifiers, including Naïve Bayes and Logistic Regression. Results shown in Table 3.

After analyzing the above discussion, I found that preprocessing has a minor impact on accuracy, and both multi-classification and a series of binary classifications result in lower accuracy. Using only the "postText" element tends to achieve better accuracy compared to using "postText" combined with "targetParagraph", with an average improvement of around 10%. The influence of different models on accuracy is relatively small, generally less than 5%. Among them, the best-performing model is the multi-classification logistic regression model with preprocessed data, achieving an accuracy of 56%.

This enlightens us that classic feature-based models do not possess sufficient capability to handle this complex classification task. Instead, we need to employ more sophisticated neural network models to tackle this challenge effectively.

3.2 CNN

CNN, short for Convolutional Neural Network, is a deep learning architecture. The key components of a CNN are convolutional layers, pooling layers, and fully connected layers. In the convolutional layer, filters (also called kernels) are applied to the input image to detect different features. These filters slide across the image, producing feature maps that capture local patterns. The pooling layer down samples the feature maps, reducing their spatial

	postText	**postText + targetParagraph**
Preprocessed + Vectorized	55%	46%
Preprocessed	54%	46%
Non-preprocessed	55%	45%

Table 1: This table shows how preprocessing affect accuracy. Accuracy based on Naïve Bayes and dataset is default training set and validation set. Accuracy calculated by accuracy$_{s}core()fromlibrarysklearn.metrics.$

Model	**postText**	**postText + targetParagraph**
Multiple classify	54%	46%
One-vs-rest	53%	46%

Table 2: This table shows how different classify methods affect accuracy. Accuracy based on Naïve Bayes and dataset is preprocessed training set and validation set. Accuracy calculated by accuracy$_{s}core()fromlibrarysklearn.metrics.$

Model	**postText**	**postText + targetParagraph**
Naïve Bayes	54%	46%
Logistic Regression	56%	51%

Table 3: This table shows how different models affect accuracy and dataset is preprocessed training set and validation set. Accuracy calculated by accuracy$_{s}core()fromlibrarysklearn.metrics.$

Parameters	**postText**	**postText + targetParagraph**
Epoch = 5	54%	47%
Epoch = 10	51%	52%
Epoch = 15	51%	51%
Activation = relu	51%	52%
Activation = sigmoid	51%	43%
Activation = tanh	52%	52%
Number of layers = 3	51%	52%
Number of layers = 4	53%	49%
Number of layers = 5	53%	47%

Table 4: This table shows how different parameters affect accuracy. The default parameters is Epoch = 10, Activation = relu, Number of layers = 3. Dataset is preprocessed training set and validation set. Accuracy calculated by accuracy$_{s}core()fromlibrarysklearn.metrics.$

dimensions and preserving the most important information. Finally, the fully connected layers use the extracted features to classify the image into different categories.

I tried several parameters for CNN and results shown in Table 4.

After attempting various CNN models with different parameters, I found that the accuracy was around 51%, and the differences between different parameter settings were not significant. It seems that this classification task may not be well-suited for CNNs. While CNNs are primarily designed for image analysis, they can also be applied to sequences, such as text strings, since they can be treated as one-dimensional sequences. However, the experimental results suggest that CNNs may not be sufficient to handle such a complex classification task. The performance of the CNN models might be limited due to the nature of the task and the complexity of the data. It is possible that other types of models or more advanced techniques are required to achieve higher accuracy in this classification problem.

3.3 BERT

BERT (Bidirectional Encoder Representations from Transformers) is a revolutionary natural language processing (NLP) model introduced by Google. Unlike traditional language models, BERT uses a bidirectional approach during pre-training, allowing it to understand the context of words in a sentence better. It generates contextualized word embeddings, which greatly improves its performance on various NLP tasks. BERT's versatility and impressive results have made it a game-changer in NLP, setting new standards and advancing the field significantly.

RoBERTa is an extension of BERT, building on its bidirectional pre-training and contextual word embeddings. It further improves performance through larger model size and training data. With a larger model size and scalability, RoBERTa excels at learning complex patterns, achieving state-of-the-art results on various NLP tasks. Results shown in Table 5.

Clearly, RoBERTa achieves an impressive accuracy of 69%, far surpassing other models, demonstrating its competence in this complex classification task. I experimented with various combinations of elements for training, and the results indicate that the differences in training outcomes among different element combinations are minimal.

3.4 Summary

After comparing various models, I found that the Roberta model performs significantly better than others. Other models, whether machine learning-based or neural networks, show inferior performance. This highlights the importance of using NLP-specific models to achieve desired results.

Interestingly, preprocessing the input data does not seem to significantly improve the accuracy. Moreover, using only the "postText" element often yields better results than using a combination of multiple elements. This indicates that adding other elements not only fails to help the model in classification but may also introduce noise and interfere with the model's performance.

4 Approach and Experiments: Spoiler Generation

Through research, it can be found that the task of generating spoilers is essentially about extracting some text from the context to use as a spoiler. This can be viewed as a kind of question-answering system, where the title is treated as the question, the paragraph as the context, and the spoiler as the answer.

A question-answering (QA) system is an artificial intelligence application designed to process and understand human language and provide accurate and relevant answers to user queries. These systems aim to mimic human-like comprehension and knowledge retrieval abilities, allowing users to interact with machines in a more natural and intuitive way.

QA systems typically consist of two main components: a question understanding module and an answer generation module. The question understanding module takes in user queries, analyzes their structure and meaning, and converts them into a format that the system can process. This module also identifies the key information and entities relevant to the question.

Once the question is understood, the answer generation module leverages context, to find the most suitable response. It uses various techniques, such as information retrieval, natural language processing, and machine learning algorithms, to search and rank potential answers based on their relevance and accuracy.

RoBERTa	Accuracy
postText	69%
postText + targetTitle	69%
postText + targetTitle + targetParagraph	67%

Table 5: This table shows how the results for RoBERTa and dataset is preprocessed training set, validation set and test set. Accuracy calculated by the result of Kaggle submission.

In this section, I will attempt to use multiple question-answering models and try to find the one that performs the best.

4.1 PyTerrier

PyTerrier is an open-source Python library designed for efficient and flexible information retrieval research. It provides a wide range of tools and functionalities to work with various information retrieval models and datasets. One of the key strengths of PyTerrier is its ability to seamlessly integrate with popular retrieval systems like Terrier, Anserini, and Elasticsearch, allowing researchers to experiment with different models effortlessly.

I used multiple models for spoiler generation through PyTerrier, and the results are shown in Table 6. Here are the introduction for models in table.

- BM25: BM25 (Best Matching 25) is a widely used probabilistic retrieval model. It is based on the bag-of-words representation and incorporates term frequency, document frequency, and document length normalization to calculate the relevance score between a query and a document. BM25 is effective for ranking documents based on their relevance to a given query.

- TF-IDF: TF-IDF (Term Frequency-Inverse Document Frequency) is a classic information retrieval technique used to represent the importance of a term in a document within a collection. It calculates the product of term frequency and inverse document frequency to assign higher weights to rare and important terms while reducing the impact of common terms.

- DirichletLM: Dirichlet Language Model is a probabilistic language model that estimates the probability of generating a document from a language model. It smooths the term probabilities by incorporating a background distribution derived from the entire collection of documents.

- Hiemstra_LM: Hiemstra Language Model is another probabilistic language model that aims to improve the retrieval performance by considering both term frequency and document frequency in a document collection. It employs a parameter to balance between these two factors.

- PL2: PL2 is a divergence-from-randomness model that measures the divergence between the observed term distribution in a document and a random term distribution. It is effective for ranking documents by considering term dependencies and frequency.

- InL2: InL2 (Information-Based Language Model) is a language modeling approach that employs a mixture of language models. It combines the document language model with a collection language model to improve retrieval accuracy.

- DLH13: DLH13 is a divergence-from-randomness model based on the Poisson distribution. It models term occurrence in a document using the Poisson distribution and ranks documents based on this modeling.

- LemurTF_IDF: LemurTF_IDF is a variant of the classic TF-IDF model. It uses a logarithmic term frequency weighting scheme to dampen the impact of highly frequent terms while boosting the importance of rare terms.

- DPH: DPH (Divergence from Poisson Hypothesis) is a statistical retrieval model that measures the divergence between the term distribution in a document and a Poisson distribution. It effectively captures term dependencies for more accurate ranking.

- In_expB2: In_expB2 is an information-based language model that incorporates both query

Model	Accuracy
BM25	16%
TF_IDF	17%
DirichletLM	17%
Hiemstra_LM	16%
PL2	16%
InL2	17%
DLH13	16%
LemurTF_IDF	17%
DPH	17%
In_expB2	17%

Table 6: Based on PyTerrier. Use postText as question, targetParagraphs as context, spoiler as result. Dataset is test set. Accuracy calculated by the result of Kaggle submission.

term frequency and document term frequency to estimate document relevance. It is designed to optimize retrieval performance in various scenarios.

The results indicate that none of the multiple models demonstrated a significantly high accuracy rate, which compels us to consider trying more complex models.

4.2 GPT-3

GPT-3 (Generative Pre-trained Transformer 3) is a cutting-edge language model developed by OpenAI. It is part of the GPT series of models and is known for its exceptional natural language processing capabilities. GPT-3 is a powerful language model with extensive pre-trained knowledge, contextual understanding, and large-scale capabilities. It excels in few-shot learning, and offers ease of use through APIs. These advantages make it an excellent choice for building advanced and versatile question-answering systems.

The first aspect I focused on was the impact of preprocessing on the results. I prepared three datasets: one was the raw, unprocessed dataset, the second one had punctuation removed, and the third one had both punctuation and stopwords removed. The results for the three datasets are shown in Table 7.

From the results, it can be observed that removing punctuation does not lead to an improvement in accuracy. Additionally, removing both punctuation and stopwords actually results in a significant decrease in accuracy.

After experimenting with GPT-3, I was astonished by its remarkable generation capabilities. Compared to other models, it significantly improved the accuracy, achieving a spoiler generation accuracy as high as 29%. It can be considered the best question-answering system for this task. Apart from the high accuracy, GPT-3 also offers the following advantages:

- Contextual Understanding: GPT-3's ability to grasp context allows it to generate more contextually relevant and coherent answers, enhancing the overall quality of responses.

- Few-Shot Learning: GPT-3's few-shot learning capability means it can adapt quickly to new tasks with minimal examples, reducing the need for extensive training data.

Overall, GPT-3's impressive generation results, contextual understanding, few-shot learning make it a standout choice for this question-answering task.

4.3 Summary

After comparing the results, it was evident that GPT-3 is better suited for this question-answering task, achieving an accuracy of 29%. In contrast, the performance of other models was relatively poor, with an average accuracy of 17%. I believe this is likely due to GPT-3's superior pre-training, which allows it to better comprehend the textual content.

Interestingly, despite the presence of some noise in the dataset, preprocessing the data did not improve accuracy; in some cases, it even led to a decrease in accuracy. This suggests that GPT-3's ability to handle unprocessed data and its context-aware understanding may have contributed to its better performance compared to traditional retrieval models.

Preprocessing	Accuracy
None	29%
Remove Punctuation	27%
Remove Punctuation and stopwords	17%

Table 7: Based on GPT3 and dataset is test set. Accuracy calculated by the result of Kaggle submission.

5 Conclusion

The Clickbait Spoiler project can help social media users avoid falling into clickbait traps. In this study, I explored multiple models and analyzed the results. It is evident that there are various ways to generate spoilers, but many models did not perform well in practice. I believe that large language models outperform other models in this task. Therefore, the core of this task should focus on large language models. It is worth noting that there are several pre-trained large language models available, such as GPT, which outperform other models significantly. In the future, this project should concentrate on how to better utilize large language models to help generate spoilers effectively.

References

Blake Burgess Andrew Trotman, Antti Puurula. 2014. Improvements to bm25 and language models examined.

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

Massimo Chiriatti Luciano Floridi. 2020. Gpt-3: Its nature, scope, limits, and consequences.

Kevin P. Murphy. 2006. Naive bayes classifiers.

Raymond E. Wright. 1995. Logistic regression.

Junshi Huang Bingbing Ni Jian Dong Yao Zhao Shuicheng Yan Yunchao Wei, Wei Xia. 2014. Cnn: Single-label to multi-label.