Clickbait Spoiling using Question Answering and Summarization models

Ankur Chemburkar **Aravind Krishnan** **Nehal Muthukumar**

Siraj Sandhu **Somil Kiran Jain**

University of Southern California
Viterbi School of Engineering

Abstract

We address the task of "clickbait spoiling," which involves creating a brief text that satisfies the curiosity aroused by a clickbait post. Clickbait posts intend to redirect inquisitive people to their websites and give long verbose texts rather than providing informative summaries. Our contributions are approaches to spoil different types of spoilers using different approaches. We employ question answering models to generate phrase spoilers and introduce two new methods to generate multi-part spoilers namely GPT-based prompt engineering model and a BART model.

1 Introduction

We are addressing the problem of clickbait spoiling - generating a short text that satisfies the curiosity generated by a clickbait post. The task is to understand the question posed by the clickbait, summarize the text on the linked web page by locating essential text spans, and extract or generate a concise, informative response that eliminates the need to click on the link. Any model tackling this task requires various elements of Natural Language Processing like text understanding and generation. This problem exists in the intersection of question-answering and text summarization sub-domains of NLP. We use both of these approaches - question-answering to address generating a phrasal spoiler by considering the clickbait as the 'query' and the linked article as the 'context' to extract the answer; and summarization to generate a passage or multi-part spoilers using decoder architectures.

Clickbait headlines often use sensational language, misleading information, or exaggerated claims to provoke curiosity and compel readers to click on the link. The conclusion of the linked web page is also usually less exciting than the clickbait portrays. Clickbaits have become a pervasive issue on the internet that raises ethical concerns, as they can manipulate readers' emotions and exploit their vulnerabilities for financial gain. In addition, clickbait can contribute to spreading fake news and disinformation, as it can lure readers into clicking on links that offer misleading or biased content. To address this issue, we generate spoilers that can quench the curiosity induced by clickbait posts. Clickbait spoilers can be either a phrase, short paragraph, or a multi-part answer, depending on the content of the data provided in the clickbait link.

This project aims to provide users with an efficient way to avoid clickbait articles that often disappoint or waste time by providing misleading or trivial information. The project also has the potential to promote ethical and honest content creation by reducing the incentives for clickbait tactics. Previous works have focused on phrase and passage-type spoilers, but we explored further to develop generative models for multi-part spoilers. [1]

2 Related Work

Viewing the problem as a question-answering problem, two benchmark datasets were considered by Hagen et al. (2022) - SQuAD (Rajpurkar et al., 2016), which compiles 107,785 questions and answers based on 536 Wikipedia articles—93.6% of which are factual in nature, such as names, noun phrases and numbers, and TriviaQA, which contains 95,000 question–answer pairs based on challenging trivia. The models used in their experiments were chosen for their state-of-the-art performance on the above benchmarks.

Another way to formulate the problem can be topic-focused single document summarization. Narayan et al. (2018) introduced an abstractive model conditioned on the document's topics and based on convolutional neural networks. Deutsch and Roth (2019) studied a two-step abstractive

[1] Code: https://github.com/nehalmuthu/Clickbait-Spoiling

model that first extractively selects a small number of sentences and then abstractively summarizes them. They observed that topical and partial summaries in the steps help the models identify relevant content.

3 Problem Statement

Clickbaits are intrusive techniques used to spread disinformation, manipulate reader's emotions and exploit their vulnerabilities. We improve upon this task by introducing two new methods to generate multi-part spoilers - GPT-based prompt engineering and BART transformer.

4 Methods

4.1 Datasets

Our project will use the existing dataset Webis-Clickbait-22, which is publicly available at https://webis.de/data/webis-clickbait-22.html.

The dataset is in JSON Lines format (.jsonl), with each line containing a clickbait post, manually cleaned versions of the linked documents, and extracted spoilers for each clickbait post. The dataset includes fields such as postText, targetParagraphs, targetTitle, humanSpoiler, spoiler, spoilerPositions, and tags.

We utilized specific columns from the original dataset, which are as follows: **Spoiler:** contains the spoiler answer that was extracted from the linked web page, providing the necessary information for the Q&A model to generate accurate clickbait responses. **spoilerPositions**: provides the character position in which the spoiler answer occurred in the linked web page, allowing the Q&A model to locate and extract the answer correctly. **postText:** contains the text of the clickbait post that needs to be spoiled, which is used as the question for the Q&A model.

We are also using SQuAD for pretraining models based on Question Answering approach.

4.2 Question Answering Approach for Phrasal Spoilers

4.2.1 Data Processing

The idea is to generate clickbait responses based on a given post with an external link that leads to a web page with relevant content. The model then extracts the answer from the linked web page and forms a clickbait response using the post as the question.

To prepare the data for the question answering models, we performed some preprocessing steps, which involved converting the data frame to a specific format with columns: 'question,' 'context,' and answers. The 'question' column was derived from the 'postText' column of the original data frame, while the 'context' column was obtained from the 'targetParagraphs' column.

For generating 'answers,' we created a dictionary that contains the Spoiler start position and the corresponding spoiler text extracted from the 'Spoiler' column of the original data. This approach allowed us to build a comprehensive dataset for the Q&A model, where the model can utilize the provided 'question' and 'context' columns to generate clickbait responses.

4.2.2 Models and setup

To implement our clickbait spoiling approach, we used the state-of-the-art models, namely BERT, RoBERTa, and DeBERTa, which have been shown to achieve excellent performance in similar tasks by Hagen et al. (2022).

To prepare the data for input into the question answering models, we first added special tokens using the AutoTokenizer in the Transformers library. We then fine-tuned the models on SQuAD first and then the custom dataset.

Although most of the hyper-parameters were left at their default values, we conducted a grid search to identify the optimal combination of hyper-parameters that maximized the performance of our models. The grid search involved varying the learning rate (1e-5, 2e-5, 4e-5, 1e-4), batch size (4, 8, 16, 32), number of epochs (1 to 5), and maximum sequence length (256, 384, 512). By evaluating the performance of the models for different hyper-parameter combinations, we were able to identify the hyper-parameter values that yielded the best results.

4.3 Summarization Approach for Multi-Part Spoilers

4.3.1 Few-Shot Prompt Engineering

With the recent advances in Large Language Models (LLMs) like GPT-3 (Brown et al., 2020) and GPT-4 (OpenAI, 2023), the applications of prompt engineering have been immensely boosted. This relatively new discipline allows us to efficiently use LLMs by developing and optimizing text prompts. Prompt engineering improves the capacity of LLMs on applications such as question answering and

logical and arithmetic reasoning. Recently Prompt Engineering has also been used in domains such as code summarization (Ahmed et al., 2023) and healthcare clinical report summarization (Chuang et al., 2023). We explore the performance of prompt engineering for our task of summarization of multi-part clickbait spoilers.

There has been extensive research (Gu et al., 2022) on the topic of designing semantically informative prompts. There are many methods for designing prompts such as Zero-shot, Few-shot, Chain-of-thought to feed in relevant information to the LLM. In our study, we experiment with Zero-shot and Few-shot prompting. As an overview, instructing an LLM without providing any examples of your task in the prompt is zero-shot prompting. In few-shot prompting, apart from an instruction telling the model what task to perform, some examples with answers are also provided in the prompt. This helps the model find patterns between examples and gives the model a template of the output format expected.

The prompt used for the samples in Table 1 is:

"""You are given a clickbait text and the corresponding linked large body of text. Extract the most important phrases from the linked text that can sarcastically answer the clickbait text.

Your answer must strictly be a list with more than one phrases. The phrases should strictly be brief (one word or few words). Follow the example given below and answer the next.
CLICKBAIT TEXT: *clickbait from training set*
LINKED TEXT: *corresponding linked text*
ANSWER: *answer to the clickbait*
CLICKBAIT TEXT: *clickbait from test set*
LINKED TEXT: *corresponding linked text*
ANSWER: """

4.3.2 BART

The clickbait post and content is treated as an input and the clickbait spoiler is given as the expected output in the form of a summary. There are 559 training examples and 143 test/validation examples of multi part spoilers in the dataset.

BART (Bidirectional and Auto-Regressive Transformers) is a transformer-based language model architecture that combines elements of both bidirectional and auto-regressive models. BART consists of an encoder-decoder structure, where the encoder uses a bidirectional transformer to encode input sequences in both forward and backward directions, while the decoder uses an auto-regressive transformer to generate output sequences in a step-by-step manner. BART is pre-trained using a denoising autoencoder objective, where the model is trained to reconstruct corrupted versions of its input. This pre-training approach helps BART to learn robust and generalizable representations of language, which can be fine-tuned for various downstream natural language processing tasks. Overall, BART's unique combination of bidirectionality and auto-regression has made it a popular choice for a wide range of language generation and comprehension tasks.

We skipped hyperparameter optimization due to computational load for these models. Instead, we tested with BART and BART-large. All hyperparameters were left at default values and run for 3 epochs with a batch size of 4 on our dataset.

5 Experimental Results

5.1 Baseline methods

We optimized the performance of the fine-tuned models by running them on the Colab GPU environment. We used BERT as the baseline for our Q&A models.

For our experiments, we leverage the GPT-3.5-turbo from the OpenAI API. This method provides a good benchmark for comparing the performance on list type or multi-part spoilers.

The GPT-3.5 has a token capacity of 4096. Out of the 143 list type clickbait in our test dataset only 7 samples exceed this limit. Hence, our results exclude these 7 samples.

In our experiments, we find that few-shot prompting works better than zero-shot prompting. This is because in our dataset, the output spoilers are expected in the form of a list of phrases. The phrases also are usually not that verbose. LLMs tend to generate long sequences of text which are overly verbose. Hence, an example tells the model how verbose of an output is expected. We also find that using just one example is sufficient and the performance worsens with additional examples. One reason for this could be that as GPT-3.5 is already more than capable of summarization, it only needs the example for the output format.

5.2 Evaluation protocols

BLEU is a weighted geometric mean of all the modified n-gram precisions, multiplied by a brevity penalty. Modified n-gram precision unduly gives a high score for candidate strings that are "tele-

Clickbait text	"Apple iPhone 7 has two nasty surprises"
Ground truth spoiler	1: Goodbye Sapphire? 2: New Home Button, New Limitations
Predicted spoiler	1) No more sapphire glass. 2) New home button has limitations.
Clickbait text	"What No One Tells You About Marriage"
Ground truth spoiler	There will be fights. There will be anger. There will be score-keeping. there will be hard times. It's OK that marriage is hard.
Predicted spoiler	1) Marriage is hard work. 2) There will be fights. 3) It's not always romantic and easy. 4) It's OK that it's hard. 5) You will get through it.

Table 1: Sample examples from GPT-3.5 turbo Prompt Engineering.

graphic", that is, containing all the n-grams of the reference strings, but for as few times as possible. The METEOR metric is based on the harmonic mean of unigram precision and recall, with recall weighted higher than precision. It also has several features that are not found in other metrics, such as stemming and synonymy matching, along with the standard exact word matching.

5.3 Results

For phrasal spoilers, we compared the performance of BERT, RoBERTa, and DeBERTa, using BLEU and METEOR. The results are shown in Table 2. Our results corroborate with the findings of Hagen et al. (2022) and show that DeBERTa out performs RoBERTa in both the metrics for this task.

For the multi-part spoilers, our prompt engineering benchmarking method shows promising results. The BLEU score shown in Table 3 is also comparable to RoBERTa and DeBERTa's score despite multi-part spoilers being much harder to generate than phrasal spoilers.

Our implementation of BART architecture for this task shows the difficulty in generating list-type spoilers. Table 3 shows a large gap between BLEU and METEOR scores when compared with Prompt-GPT.

Model	BLEU	METEOR
BERT (baseline)	58.25	55.24
DeBERTa	**65.43**	**66.04**
RoBERTa	63.92	65.40

Table 2: BLEU and METEOR Scores of Question Answering models

Model	BLEU	METEOR
Prompt-GPT (baseline)	**63.38**	**57.53**
BART-lg	50.87	48.65

Table 3: BLEU and METEOR Scores of summarization models

6 Conclusions and Future Work

Using different approaches to clickbait spoiling, we have explored the effectiveness of existing state-of-the-art solutions and introduced new summarization models to handle multi-part spoilers.

Large pre-trained Summarization models to handle multi-part spoilers work in par with question-answering models, with Prompt-GPT outperforming BART.

Future work could involve exploring the use of other models like T5 and Pegasus and comparing their performance to the models tested in this project. Additionally, further investigation into the use of different training datasets and approaches could be tested.

7 Division of Labour

Ankur: Performed dataset preprocessing and implemented GPT Prompt Engineering Model for Multi-part spoiler generation.

Aravind: Implemented and evaluated the three question-answering models and corroborated results of base paper.

Nehal Muthukumar: Finetuned BERT, RoBERTa, and DeBERTa models. Successfully matched the SOTA results. Wrote scripts to evaluate BLEU and METEOR scores. Researched on methods for handling multi-part spoilers.

Siraj Sandhu: Dataset preprocessing design, evaluation scripts implementation, and BART adaptation research.

Somil Kiran Jain: Performed data preprocessing, implementation and hyperparameter tuning for multi-part spoiler generation using BART. Explored different summarization techniques to generate multi-part spoilers.

References

Toufique Ahmed, Kunal Suresh Pai, Premkumar Devanbu, and Earl T. Barr. 2023. Improving few-shot prompts with relevant static analysis products.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners.

Yu-Neng Chuang, Ruixiang Tang, Xiaoqian Jiang, and Xia Hu. 2023. Spec: A soft prompt-based calibration on mitigating performance variability in clinical notes summarization.

Daniel Deutsch and Dan Roth. 2019. Summary cloze: A new task for content selection in topic-focused summarization. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3720–3729, Hong Kong, China. Association for Computational Linguistics.

Yuxian Gu, Xu Han, Zhiyuan Liu, and Minlie Huang. 2022. PPT: Pre-trained prompt tuning for few-shot learning. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8410–8423, Dublin, Ireland. Association for Computational Linguistics.

Matthias Hagen, Maik Fröbe, Artur Jurk, and Martin Potthast. 2022. Clickbait spoiling via question answering and passage retrieval. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7025–7036, Dublin, Ireland. Association for Computational Linguistics.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1797–1807, Brussels, Belgium. Association for Computational Linguistics.

OpenAI. 2023. Gpt-4 technical report.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.