Alexander Knox at SemEval-2023 Task 5: The comparison of prompting and standard fine-tuning techniques for selecting the type of spoiler needed to neutralize a clickbait

Mateusz Woźny and **Mateusz Lango**
Institute of Computer Science
Poznan University of Technology
Poznan, Poland
mateusz.wozny@student.put.poznan.pl

Abstract

Clickbait posts are a common problem on social media platforms, as they often deceive users by providing misleading or sensational headlines that do not match the content of the linked web page. The aim of this study is to create a technique for identifying the specific type of suitable spoiler - be it a phrase, a passage, or a multipart spoiler - needed to neutralize clickbait posts. This is achieved by developing a machine learning classifier analyzing both the clickbait post and the linked web page. Modern approaches for constructing a text classifier usually rely on fine-tuning a transformer-based model pre-trained on large unsupervised corpora. However, recent advances in the development of large-scale language models have led to the emergence of a new transfer learning paradigm based on prompt engineering. In this work, we study these two transfer learning techniques and compare their effectiveness for clickbait spoiler-type detection task. Our experimental results show that for this task, using the standard fine-tuning method gives better results than using prompting. The best model can achieve a similar performance to that presented by Hagen et al. (2022).

1 Introduction

1.1 Problem description

Clickbaits are social media posts designed to grab the reader's attention in order to visit a linked web page. As they are used by publishers to increase the number of views, clickbait posts are becoming more and more common in the webspace.

Even though some clickbait posts are harmless, a significant amount of them is misleading and help spread fake news. Therefore, some research works have been devoted to clickbait detection problem (Potthast et al., 2016), leading e.g. to the proposal of an additional plug-in for the browser to alert the user of a clickbait header (Chakraborty et al., 2016). However, clickbait detection alone is not enough, as it only identifies the problem, but does not solve it.

Therefore, Hagen et al. (2022) proposed using automatically generated *spoilers* to overcome the issue of clickbait posts. Spoilers are short texts that provide the most necessary information from the web page linked to the clickbait, completely neutralizing the clickbait post. Three types of spoilers are put forward (Hagen et al., 2022):

- phrase – a single word or phrase from the linked page
- passage – one or a few sentences from the linked page
- multipart – a few non-consecutive phrases or passages extracted from the linked page

For instance, for a given clickbait post "Obama Admin Just Outlawed Wildly Popular Product Found in Millions of Bathrooms", the corresponding spoiler is a phrase "antibacterial soaps" constructed from the content of the linked page.

The current paper tackles the first step in designing a successful spoiler generator, i.e. it concerns spoiler-type classification for the given clickbait post and the linked page. The purpose is to determine the type of spoiler needed to completely neutralize a clickbait post.

1.2 The explored methods

In this paper, the task of identifying the type of spoiler needed to neutralize a clickbait is treated as a text classification task i.e. we predict the type of needed spoiler based on the text of a social media post and the textual content of the linked web page. The methods for text classification range from classical bag-of-word approaches to more recent deep-learning ones. In particular, for many text classification tasks the state-of-the-art results are obtained by means of transfer learning from

Proceedings of the The 17th International Workshop on Semantic Evaluation (SemEval-2023), pages 1470–1475

self-supervised models pre-trained on large corpora.

A transfer learning method that is currently gaining popularity is prompting (Gao et al., 2020). It is usually applied within the few-shot learning paradigm and involves transforming a task (such as text classification) into a text generation task. A selected Large Language Model (LLM) such as GPT-3 (Brown et al., 2020), BLOOM (Scao et al., 2022) or LLaMA (Touvron et al., 2023), is queried with a human-designed prompt that introduces several examples into the model and redesigns a task to a text completion. Therefore, transfer learning through prompting does not require model fine-tuning or having a large supervised dataset and only a human-designed prompt.

Therefore, we decided to verify the usefulness of prompting to the clickbait spoiler-type classification problem. Apart from designing and testing the performance of prompts for the task, we also explored the possibility of generating spoilers by a LLM and using it as an additional input to the classifier. More concretely, LLM was used to generate spoilers for all the examples in the validation set[1] basing on a prompt with few examples taken from the training set[2]. Later, the final training set for the classifier was constructed by combining the examples with original spoilers and examples with artificially generated spoilers. Finally, a standard model fine-tuning was performed on the such created training dataset and the effectiveness of the model was tested only on the test data with artificially generated spoilers.

The last classification method explored in this paper is the standard model fine-tuning. Fine-tuning a model for a classification task is a process of adapting a pre-trained language model to a specific text classification problem (Howard and Ruder, 2018) by performing several updates of the model weights through e.g. several epochs of standard backpropagation on supervised data. Such a process allows leveraging the general linguistic knowledge of the pre-trained model to construct a better model for a given task.

The performed experiments demonstrate that the last method i.e. classical model fine-tuning obtains superior classification performance in comparison to approaches based on prompting.

[1] *validation.jsonl* file taken from Zenodo.org

[2] *train.jsonl* file taken from Zenodo.org

2 Background

2.1 Problem definition

The main aim of this work is to construct a machine learning model to classify the type of spoiler needed to neutralize a given clickbait post. Three spoiler types are considered: phrase, passage and multipart spoilers. Even though for some types of clickbait posts, it is easy to specify the appropriate type of spoiler, in general, the task poses many challenges due to the diversity of language itself and the various topics that the posts may cover.

For instance, for the following post "NASA sets date for full recovery of ozone hole" it is relatively easy to determine what the spoiler should look like since some specific date should be expected. On the other hand, for many clickbait posts, it is much more challenging to determine the suitable spoiler type. Let us consider the "Should you drink Red Wine?" clickbait post. Such a question could be answered shortly (e.g. "yes") without any justification or can be neutralized by a short passage containing some reasons to support the claim. Neutralization of this clickbait can even require a multipart spoiler that lists several arguments, both supporting and opposing the claim.

2.2 Related works

To the best of our knowledge, only the work of Hagen et al. (2022) have so far addressed the problem of the spoiler type detection. The authors employed both classical machine learning methods such as logistic regression and neural models such as transformer networks to tackle this task. They also explored the possibilities of using transfer learning techniques, but their work was limited only to the classical fine-tuning of Large Language Models. In this work, we also used a fine-tuned RoBERTa model to construct the classifier, but also explored the possibilities of prompt-based transfer learning.

2.3 Prompt engineering

Prompt engineering is the process of designing and crafting high-quality prompts for language models. The goal of prompt engineering is to optimize the performance of language models on specific tasks, by providing tailored prompts that effectively guide the model toward the desired outputs [3]. This

[3] Text generated by ChatGPT, OpenAI, ChatGPT

makes it possible to use previously trained Large Language Models without fine-tuning them and to extract the knowledge gained by the model during pretraining. Prompts for few-shot learning are typically constructed based on the manually designed textual template filled with the example and the expected answer. The test example is concatenated to such prompt also by filling the same template with the example, but without providing the answer. The task of the model is to complete the missing part of such prepared text. Currently, one of the most popular tools that can be queried using prompts is ChatGPT which partly created this subsection.

3 Prompting engineering for spoiler-type classification

3.1 Prompts for classification

Prompts needed for the classification task should clearly instruct the LLM that its task is to return one of the specified classes. In this work we used the following prompt structure for the classification task: "Question {clickbait post} Type of answer {spoiler type}". We tested this prompt template in a few-shot learning fashion by concatenating corresponding texts for several training examples for each class. We consider prompts with 2, 5, 10 or 15 examples for each class. Prediction of the new class consisted of completing the text "Question {clickbait post} Type of answer".

3.2 Prompts for data augmentation

Having a spoiler before predicting its type would obviously be very helpful for classification performance. One idea could be to use spoilers from the dataset, but they are not available during the testing phase, as the whole task is to facilitate easier spoiler generation. Therefore, we used LLMs to generate (noisy) spoilers for each clickbait post and used them as additional input to the classifier.

Prompt structure for spoiler generation has the following form: "Question {clickbait post} Context {main content of linked web page} Answer {spoiler}". Due to the varying number of characters (sometimes very large) in linked web pages, their content was limited to the first 2000 characters. In this setting, few-shot learning with 2 examples was used. Generation of new spoiler consisted of completing the text "Question {clickbait post} Context {main content of linked web page} Answer". Spoilers generated in this way were added to the corresponding clickbait posts, constructing a new training set for fine-tuning or training a classifier. The dataset contains training examples in the form of "Clickbait {clickbait post} Spoiler {spoiler}" texts together with assigned gold-standard class (i.e., spoiler type).

4 Spoiler-type classification using fine-tuning

As a baseline, standard fine-tuning of the pretrained MLM model was used. Fine-tuning typically requires a larger training dataset (in comparison to few-shots only), but also tends to be a more robust approach, since it allows the model to learn from many examples and adapt to the task at hand.

4.1 Structure of training examples

To construct training examples for MLM, we converted each training instance to a textual form. Each example looks as follows "Clickbait {clickbait post} Article {main content of linked web page}', where the content of a linked web page was limited to its first 2000 characters. This enabled faster training of the model and reduced memory requirements.

5 Experimental evaluation

Due to the fact that the test dataset for subtask 1 at SemEval 2023 Task 5 (Fröbe et al., 2023a) has not been made publicly available, the following experiments use the provided validation examples as the test dataset and as validation dataset a part of training data was used.

As an LLM, we used BLOOM (Scao et al., 2022) since it is:

- open-source
- trained to complete text from a prompt on vast amounts of text data using industrial-scale computational resources
- can be used to perform text tasks it hasn't been explicitly trained for, by casting them as text generation tasks

For standard fine-tuning, MLM models: RoBERTa (Liu et al., 2019), DistilBERT (Sanh et al., 2019), and ALBERT (Lan et al., 2019) with a softmax layer attached to the first token were used as classifiers. Each model was fine-tuned with 7 learning epochs only, since it was observed that increasing the number of epochs did not result in higher performance on the validation set. The

transformers[4] library provided by HuggingFace company was used to implement the model.

5.1 Performed experiments

In order to determine the most effective method for spoiler-type classification, three different experiments were conducted. For each experiment, balanced accuracy (Acc.) over all three spoiler types, as well as precision (Pr.), recall (Rec.), and F1 score (F1) for each class separately were reported. For model fine-tuning performed in 5.3 and 5.4 the following hyperparameters were used: learning rate (2e-5), batch size (8 in 5.3 and 6 in 5.4), number of training epochs (7), and weight decay (10).

5.2 Spoiler-type classification using LLM

In this experiment, the spoiler-type classification task was converted to a text generation task, which was then to be solved by the selected LLM (BLOOM). The method of conversion was presented in 3.1. The experiment checked a different number of training examples for each class. The passing of 2, 5, 10 and 15 examples per class was tested.

The method used does not need a validation dataset (model fine-tuning is not performed). According to the 3.1 structure, the training examples were sentences built on the basis of a set of clickbait posts, which included i.e. five observations from each class (the class is the spoiler type). The source of training data was a file *train.jsonl*[1] and test examples were all observations from the file *validation.jsonl*[1].

Table 1: The classification performance of prompting BLOOM on spoiler type detection task. The results are computed on 800 test posts created from *validation.jsonl*, where n is the number of examples for each class.

n	Acc.	Phrase			Passage			Multi		
		Pr.	Rec.	F1	Pr.	Rec.	F1	Pr.	Rec.	F1
2	0.34	0.43	0.81	0.56	0.47	0.21	0.29	0.35	0.08	0.13
5	0.40	0.44	0.28	0.34	0.44	0.64	0.31	0.27	0.29	0.11
10	0.37	0.43	0.60	0.50	0.00	0.00	0.00	0.22	0.52	0.31
15	0.34	0.43	0.49	0.46	0.44	0.14	0.21	0.18	0.41	0.25

The results in table 1 show that an increase in the number of training examples does not necessarily increase the effectiveness of the model. Increasing the number of examples, and thus the length of the input example causes an increase of inference time.

5.3 Spoiler-type classification using generated spoilers and model fine-tuning

Due to the diversity of websites regarding volume, two training examples were selected for each class from the *train.jsonl*[1]. A spoiler was then generated for each example from the file *validation.jsonl*[1] using BLOOM and then use instead of the spoiler included in the file. Next fine-tuning of the selected model was performed for which the training and validation datasets contained a total of 2,500 examples with spoilers included in *train.jsonl*[1] and 300 examples with generated spoilers (the datasets were divided in a ratio of 80:20). The test dataset consisted only of 500 examples with generated spoilers. The reason why testing is used only 500 examples is that 300 examples (with generated spoilers) are used for training. This is because the fine-tuning model should see in training some of the samples with generated spoilers, not only with spoilers from the file.

In this experiment, spoilers were generated based on the structure shown in 3.2, and then using examples with original and generated spoilers, fine-tuning of the model was performed with training examples created based on the scheme shown on the end of the section 3.2.

Table 2: The classification performance of fine-tuned model with artificially generated spoilers by prompting. The results are computed on 500 test posts created from *validation.jsonl* (training: 2240; validation: 560).

Model name	Acc.	Phrase			Passage			Multi		
		Pr.	Rec.	F1	Pr.	Rec.	F1	Pr.	Rec.	F1
DistilBERT	0.37	0.46	0.41	0.46	0.59	0.51	0.32	0.17	0.22	0.18
RoBERTa	0.45	0.49	0.47	0.48	0.54	0.67	0.55	0.27	0.15	0.14
ALBERT	0.38	0.46	0.41	0.43	0.46	0.59	0.51	0.32	0.17	0.22

The results presented in table 2 indicate that using the fine-tuning of the model together with the generated spoilers (in the case of RoBERTa) a better result is obtained than using only the prompting alone. Obtaining better results in this case, however, is biased with an increase in computation time. Generating spoilers alone (even with only 2 training examples per class) takes much more time than querying the model based only on clickbait post and attached spoiler type.

Unfortunately in more cases generated spoilers

[4] https://huggingface.co/docs/transformers/index

are not good. This seems they were not such helpful as they could be. Generated spoilers with correspondent clickbait posts and original spoilers are available at the link generated spoilers.

5.4 Spoiler-type classification using model fine-tuning

In the last experiment, only the fine-tuning of the model was checked. The input data was prepared as described in 4.1. As model input was used the text of clickbait posts and the main content of the connected article. Training and validation datasets were created based on *train.jsonl*[2] file and test dataset was created based on *validation.jsonl*[1].

Table 3: The classification performance of fine-tuned models on spoiler type classification on 800 test posts created from *validation.jsonl* (training: 2560; validation: 640).

Model name	Acc.	Phrase			Passage			Multi		
		Pr.	Rec.	F1	Pr.	Rec.	F1	Pr.	Rec.	F1
DistilBERT	0.65	0.70	0.61	0.66	0.70	0.61	0.66	0.64	0.60	0.62
RoBERTa	0.73	0.75	0.71	0.73	0.70	0.77	0.73	0.76	0.70	0.73
ALBERT	0.66	0.68	0.64	0.66	0.65	0.70	0.67	0.65	0.64	0.64

Comparing the results of this experiment (table 3) with those in table 1 and table 2 shows that the current results are significantly better.

5.5 Comparison of results

The results obtained by model fine-tuning definitely outperform the other two approaches. For each of the tested models, a clear increase in accuracy can be observed. Ultimately, the best model turns out to be RoBERTa. Based on the presented experiments, RoBERTa was used as the final model for subtask 1 at SemEval 2023 Task 5. The above observations may be due to several factors. One of them may be the improper structure of the prompts. It is difficult to explicitly measure the effectiveness and matching of a manually created prompt to a given dataset and Large Language Model.

6 Results on the SemEval-2023 test set

This section will present the model results obtained for the test dataset for subtask 1 at SemEval 2023 Task 5. The results were generated using TIRA (Fröbe et al., 2023b).
The results obtained on the test dataset for the selected model are presented in table 4. Comparing the results presented in this work with those in the paper Hagen et al. (2022) shows a slightly worse result for the same model (RoBERTa, 71.57%). This may be due to a different training data structure, limited computing resources, or different preprocessing methods. In paper Hagen et al. (2022), it was not precisely described for which values of hyperparameters the best model was obtained. Currently, the reproducibility of neural models is a fairly common problem. The problem of reproducibility of neural network models is the difficulty in reproducing the same model and results using the same code and data. This is due to the many sources of variation that can occur in the training process, such as optimization algorithms, as well as differences in hardware and software configurations.

Table 4: Overview of the effectiveness in spoiler type prediction (subtask 1 at SemEval 2023 Task 5) measured as balanced accuracy (Acc.) over all three spoiler types and precision (Pr.), recall (Rec.), and F1 score (F1) for a phrase, passage, and multi spoilers on the test dataset.

Acc.	Phrase			Passage			Multi		
	Pr.	Rec.	F1	Pr.	Rec.	F1	Pr.	Rec.	F1
0.70	0.71	0.78	0.75	0.73	0.72	0.72	0.78	0.60	0.68

7 Summary

Prompting is certainly a very useful technique that allows you to query the model even with a small training dataset. However, the presented experiments obtained worse results than the standard fine-tuning method. Keep in mind that creating new prompts is not trivial and sometimes even requires thorough domain knowledge. One solution is the automatic creation of prompts presented in the paper Shin et al. (2020). It has not been tested as part of this work and maybe a potential study in the future.
Another possibility would be to check out another model based on the Transformer architecture. Hagen et al. (2022) also checked the DeBERTa model, but it was not verified in this work due to limited computational resources.

The source code and the best model can be accessed via the links given in the appendix.

Ack. Authors were supported by AI Tech project sponsored from POPC programme POPC.03.02.00-00-0001/20

A Appendix

Our implementation can be found at https://github.com/mateusz-wozny/clickbait-spoiling-alexander-knox.
The model can be downloaded from https://huggingface.co/MateuszW/alexander-knox-classification

References

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.

Abhijnan Chakraborty, Bhargavi Paranjape, Sourya Kakarla, and Niloy Ganguly. 2016. Stop clickbait: Detecting and preventing clickbaits in online news media. *CoRR*, abs/1610.09786.

Maik Fröbe, Tim Gollub, 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.

Tianyu Gao, Adam Fisch, and Danqi Chen. 2020. Making pre-trained language models better few-shot learners. *CoRR*, abs/2012.15723.

Matthias Hagen, Maik Fröbe, Artur Jurk, and Martin Potthast. 2022. 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.

Jeremy Howard and Sebastian Ruder. 2018. Fine-tuned language models for text classification. *CoRR*, abs/1801.06146.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. ALBERT: A lite BERT for self-supervised learning of language representations. *CoRR*, abs/1909.11942.

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. *CoRR*, abs/1907.11692.

Martin Potthast, Sebastian Köpsel, Benno Stein, and Matthias Hagen. 2016. Clickbait detection. In *European Conference on Information Retrieval*.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of BERT: smaller, faster, cheaper and lighter. *CoRR*, abs/1910.01108.

Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, Jonathan Tow, Alexander M. Rush, Stella Biderman, Albert Webson, Pawan Sasanka Ammanamanchi, Thomas Wang, Benoît Sagot, Niklas Muennighoff, Albert Villanova del Moral, Olatunji Ruwase, Rachel Bawden, Stas Bekman, Angelina McMillan-Major, Iz Beltagy, Huu Nguyen, Lucile Saulnier, Samson Tan, Pedro Ortiz Suarez, Victor Sanh, Hugo Laurençon, Yacine Jernite, Julien Launay, Margaret Mitchell, Colin Raffel, and et al. 2022. Bloom: A 176b-parameter open-access multilingual language model.

Taylor Shin, Yasaman Razeghi, Robert L. Logan IV, Eric Wallace, and Sameer Singh. 2020. Autoprompt: Eliciting knowledge from language models with automatically generated prompts. *CoRR*, abs/2010.15980.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. Llama: Open and efficient foundation language models.