

Clickbait Spoiling: Spoiler Type Classification

Belén Agüera
UPV/EHU
San Sebastián-Donostia
email@domain

Sara Han Díaz
UPV/EHU
San Sebastián-Donostia
sdiaz043@ikasle.ehu.eus

Abstract

This work is part of the clickbait spoiling challenge. Our aim is to propose an approach to classify what kind of spoiler to satisfy readers' curiosity is needed (i.e., a phrase, a passage or multiple non-consecutive pieces of text) by a clickbait post. In addition, we explore different configurations of data preprocessing to study the best way to deal with texts length beyond the established maximum. The results suggest that text cleaning is beneficial for classification, whereas with the combination of all the proposed techniques at the same time, performance is impaired probably because of the loss of some useful information.

1 Introduction

Clickbait refers to a form of web content that employs writing formulas and linguistic techniques in headlines to trick readers into clicking links (Rony et al., 2017). These misleading or sensational headlines arouse readers' curiosity to promote their content, rather than providing helpful summaries. In the context of the 17th International Workshop on Semantic Evaluation (SemEval-2023), a Clickbait spoiling task was proposed, whose aim was to identify or generate a short text (spoiler) that satisfy the curiosity induced by a clickbait post.

Hagen et al. (2022) suggest a two-step strategy to clickbait spoiling that first classifies a clickbait post according to its spoiler type, to later apply different approaches depending on what will work best for each spoiler type. In this paper, we address the first step, which corresponds to the subtask 1 of the Clickbait Challenge: Spoiler Type Classification. Our objective is therefore to train a model that gets as input the clickbait post and the linked document and classify whether the spoiler warranting the post clickbait is a phrase, a passage or a multipart spoiler (Hagen et al., 2022).

The dataset used for the training contains clickbait posts, and for each, a cleaned version of the




Clickbait tweet	Spoiler
 Lifehacker @lifehacker How to keep your workout clothes from stinking: lifehac.kr/57Y0uEZ	"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 @CNBCMakelt) cnb.cx/2TG6zeX	"1. Added sugar" [...] "2. Fried foods" [...] "3. High-glycemic-load carbohydrates" [...] "4. Alcohol" [...] "5. Nitrates" [...]"

Figure 1: Examples of clickbait tweets and spoilers for them of the three types extracted from the respective linked web page (Hagen et al., 2022).

linked document, the extracted spoiler and the spoiler type. Because of our limited computational resources and memory, we have used a tiny-BERT model (Sanh et al., 2020).

The first step was to preprocess the data. In relation to tokenization, as the length of the documents was longer than the maximum sequence length, the execution warned about possible indexing errors. Hence, we have studied how this affects the performance and how to reduce the length of the sequences to be tokenized. We have checked whether working with a cleaned version of the data set improves or not its performance, lemmatizing and removing punctuation and stop words from the texts with Spacy. Another approach studied for this problem was to use the special tokens provided by the tokenizer to separate the tag from the informative fields. Finally, we have manually entered these same special tokens between the different fields under the assumption that they would help to further minimize the length problem.

2 Related work

The development of the transformers has had a significant impact on the field of Natural Lan-

guage Processing (NLP) since their introduction by Vaswani et al. (2017). This type of neural network architecture uses self-attention mechanisms to model long-range dependencies in input sequences more effectively than other more traditional neural networks architectures. They have been used in a wide range of NLP tasks, and particularly the BERT (Bidirectional Encoder Representations from Transformers) model (Devlin et al., 2019) has achieved state-of-the-art performance on a variety of NLP benchmarks, demonstrating the effectiveness of transformers. In the context of spoiler type classification, transformers are relevant because of their potential capturing contextual information and identifying relationships in the input data.

In recent years, this progress in NLP has been accompanied by the advances in the development of frameworks for text analysis. One such tool is Spacy, an open-source library for NLP in Python, which offers functionalities for text processing, including tokenization and lemmatization, among others. It is designed to be fast and efficient, specifically for processing large volumes of text, making it a good choice for applications with limited computing resources.

3 Datasets

The dataset (Hagen et al., 2022) used during this work is the one provided by Clickbait Challenge at SemEval 2023, and as their authors state, was created for this purpose and is available in Zenodo¹.

3.1 Qualitative description

This dataset is composed of 5000 clickbait posts extracted from Twitter, Reddit, and Facebook annotated with three tags: phrase, passage and multi. Phrase referred to spoilers consisting of a single word or phrase from the complete text; passage, one or more sentences; and, multipart, more than one non-consecutive phrases or passages.

In addition, several fields were available, although, for our purpose, we just focus on five: 1) the uuid of the entry, which identifies the post; 2) the postText with the clickbait post; 3) the targetParagraphs with the content of the webpage; 4) the targetTitle with the title of the webpage; and 5) tags, which indicates the associated label.

¹<https://zenodo.org/record/6362726#.ZELeF3ZBy3B>

3.2 Quantitative description

As we have already mentioned, the complete corpus has 5000 clickbait posts randomly divided into train, validation, and test with a proportion of 80/20/20. In Table 1, we can observe the number of samples for each split. Moreover, we should not that most of the spoilers are phrases and passages, but due to the size of the corpus and that the data is unbalanced in all the splits, no methods to balance it were applied.

	Entries	1	2	3
Train	3200	1367	1274	559
Validation	800	335	322	143
Test	1000	423	403	174

Table 1: Number of entries and balance of the labels (1=phrase, 2=passage, 3=multipart) per set.

4 Methodology

In this section, we will describe in detail the steps followed to carry out the spoiler type classification. Thus, we will start by indicating the preprocessing steps used before starting to train our model, continue by indicating the parameters of our model and end with the testing and predictions.

It is worth noting that in preparing our model, one of the main questions was how to deal with the concatenation of the data so that the model could understand properly the word embeddings, as the 'targetParagraphs' in particular were relatively long in length and the tokenizer from HuggingFace has a maximum length of 512, although truncation can be applied. For this reason, while solving Clickbait spoiling task 1, this became our second research question. To achieve this, we have tested the combination of three parameters:

- Spacy. Used in 'targetParagraphs' to perform a first reduction of the length of this field by eliminating punctuation and stopwords thanks to the use of this library. In addition, lemmatization is also applied.
- Manually added special tokens. Concerning the addition of the special tokens manually, so that [CLS] is used at the beginning to indicate each new entry and [SEP] between each column text, to indicate that they are different fields. The result would be similar to the following: '[CLS] postText [SEP] targetParagraphs [SEP] targetTitle'.

- Automatically added special tokens. Concerning the addition of the aforementioned special tokens by the AutoTokenizer which works as follows: '[CLS] postText targetParagraphs targetTitle [SEP]'.

4.1 Preprocessing

Using the right data to train our model and get the best performance is essential. Therefore, we can divide this section into two subsections.

Firstly, we started by creating the suitable datasets to be able to manage the data. So, we loaded the .jsonl files in a dataframe to make it easier and more visual to work with them, and we removed those fields that we were not interested in. Moreover, since this data was in a list, which in the future does not fit the required format, it was casted to strings. Spacy was then applied to clean up the 'targetParagraphs', if set, and the fields 'postText', 'targetParagraphs' and 'targetTitle' were concatenated in the chosen way. Finally, we split the 'validation data' into validation and test, since as mentioned above, it has not yet been published in the SemEval task, and the required and resulting fields ('uuid', 'text' and 'labels') were saved again as .jsonl files.

On the other hand, we loaded the resulting files as a Dataset object and converted our labels, which were categorical, to integers using `ClassLabel` for easy reading by the transformer. We then applied the 'distilbert-base-uncased' AutoTokenizer with the appropriate parameters to tokenize the data again and vectorize it for the model.

4.2 Training and validating

So, having preprocessed our data, we began to fine-tune our model. In this work, it has been decided to use a tiny-BERT, namely the `AutoModelForSequenceClassification` 'distilbert-base-uncased' used for text classification.

Once we had chosen the model, we established the training arguments. Of these, we should note that we used batch size of 16 and 10 epochs due to resource constraints, although we believe these are adequate to answer our research questions. In addition, the optimizer 'adamw_torch' has been used.

Finally, we chose accuracy as the metric used to check the performance of our model when training. Thus, having set all the required parameters, we

started to train the model using the train set and evaluating with the validation set.

4.3 Testing and predicting

Since our models were saved, we decided to test their performance with the test set we created at the beginning. In case you want to use another dataset, it is worth remembering first that you need to fit that data into the right format.

So, we loaded the desired tokenizer and model from the ones we saved and made the predictions. Then, to evaluate it, we compared the predicted labels with the real ones, and we got two outputs: the first one in which we indicate the uuid with the corresponding true label and predicted label: and the second one in which we calculate the accuracy, precision, recall and f1-score.

Finally, we added a section to make predictions only. Again, the input must comply with the required format; while the output follows the guidelines set by the Clickbait spoiling task: "uuid": "<UUID>", "spoilerType": "<SPOILER-TYPE>".

5 Results

As mentioned above, different tests were performed depending on whether Spacy, manual special tokens or automatic special tokens were used, with at least one type of the latter two having to be present. As a result of combining these, six different models were obtained, the results of which are shown in Table 2 and 3.

1. **M1:** spacy=False, manualtokens=False, automatictokens=False. We don't use Spacy, don't add tokens manually, and add automatic tokens.
2. **M2:** spacy=True, manualtokens=False, automatictokens=False. We use Spacy, don't add tokens manually, and add automatic tokens.
3. **M3:** spacy=True, manualtokens=True, automatictokens=False. We use Spacy, add tokens manually, and add automatic tokens.
4. **M4:** spacy=True, manualtokens=True, automatictokens=True. We use Spacy, add tokens manually, and don't add automatic tokens.
5. **M5:** spacy=False, manualtokens=True, automatictokens=True. We don't use Spacy, add tokens manually, and don't add automatic tokens.

6. **M6**: spacy=False, manualtokens=True, automatictokens=False. We don't use Spacy, add tokens manually, and add automatic tokens.

		Accuracy
M1	Train	0.654
	Validation	0.681
M2	Train	0.662
	Validation	0.673
M3	Train	0.604
	Validation	0.654
M4	Train	0.659
	Validation	0.685
M5	Train	0.678
	Validation	0.696
M6	Train	0.660
	Validation	0.653

Table 2: Performance on the train and validation sets.

	Accuracy	Precision	Recall	F1-score
M1	0.669	0.668	0.665	0.665
M2	0.725	0.728	0.698	0.709
M3	0.688	0.726	0.684	0.696
M4	0.644	0.612	0.620	0.614
M5	0.606	0.606	0.634	0.616
M6	0.669	0.679	0.646	0.659

Table 3: Performance on the test set.

6 Analysis

Thus, if we analyse the results obtained, we can see that in training, the model with the best accuracy was model 5, in which only the manual tokens were applied, followed by the model 2. While the worst reported performance was the model 3. Looking at these results alone, two conclusions can be drawn. The first is that, of course, it is relevant to consider the length of the input data and the use of special tokens so that it can be vectorized correctly (no information is missed), and the model can understand it, and thus obtain better results. The second is that by far the model with the worst performance is the one in which all options have been applied, which may mean that not only the use of a double addition of special tokens but also the elimination of connections between words by preprocessing is detrimental to the model's result when they are combined. After all, the performance of each of them individually obtained similar results, and when double addition was used, but not Spacy, the accuracy was not affected either.

On the other hand, in the test, model 2 stood out (Spacy and the special automatic tokens were

applied) reaching an accuracy of 0.725 and an F1-score of 0.709. This result continues to support our hypotheses. In fact, model 2 was the second-best model during training. Furthermore, we can also add a new observation: automatically added tokens work better than manually added tokens, even though there is no clear separation between the different fields. We can see this because both model 4 and 5 obtained worse and similar results when tested.

7 Conclusions

In summary, throughout this paper, we have tested different models to complete task 1 of the Clickbait Challenge at SemEval 2023. In this way, we have described the methodology we have used and come to the conclusion that the best option is to pre-process the data and clean the corpus, and then apply the tokenization provided by the transformers.

As this spoiler type classification is framed within the broader task of generating clickbait posts' spoilers, it is important to take into account that the full text may be necessary to extract or create the spoiler itself, so cleaning up the texts may present some problems for the development of task 2. For this reason, the future research objectives are several.

Firstly, because an official test set has not yet been published by the SemEval team, we would like to test this model with such data and check whether our conclusions would still apply. Secondly, we would like to conduct further tests with better computational resources and other pretrained models that would allow us to assess whether these findings are applicable to other transformer models. Finally, going one step further, it would be ideal to test other word embeddings structures for concatenating data of various types (these being the fields we use in the current tests or other relevant fields also provided for training), as better results could be achieved with them.

References

- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. [Bert: Pre-training of deep bidirectional transformers for language understanding](#).
- Matthias Hagen, Maik Fröbe, Artur Jurk, and Martin Potthast. 2022. [Clickbait spoiling via question answering and passage retrieval](#).

Md Main Uddin Rony, Naeemul Hassan, and Mohamad Yousuf. 2017. [Diving deep into clickbaits: Who use them to what extents in which topics with what effects?](#) In *Proceedings of the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2017*, ASONAM '17, page 232–239, New York, NY, USA. Association for Computing Machinery.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2020. [Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter](#).

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. [Attention is all you need](#).