Sabrina Spellman at SemEval-2023 Task 5: Discover the Shocking Truth Behind this Composite Approach to Clickbait Spoiling!

Simon Birkenheuer and **Jonathan Drechsel** and **Paul Justen** and **Jimmy Pöhlmann** and **Julius Gonsior** and **Anja Reusch**
Technische Universität Dresden
firstname.lastname@mailbox.tu-dresden.de

Abstract

This paper describes an approach to automatically close the knowledge gap of Clickbait-Posts via a transformer model trained for Question-Answering, augmented by a task-specific post-processing step. This was part of the SemEval 2023 Clickbait shared task (Fröbe et al., 2023a) - specifically task 5. We devised strategies to improve the existing model to fit the task better, e.g. with different special models and a post-processor tailored to different inherent challenges of the task. Furthermore, we explored the possibility of expanding the original training data by using strategies from Heuristic Labeling and Semi-Supervised Learning. With those adjustments, we were able to improve the baseline by 9.8 percentage points to a BLEU-4 score of 48.0%.

1 Introduction

This project was part of the SemEval 2023 Clickbait shared task (Fröbe et al., 2023a). Specifically, we focused on the task of generating spoilers for clickbait posts. We viewed this task as an extractive question-answering problem, since a knowledge gap (or question) posed by a headline shall be closed or answered by a passage from an article. Based on this assessment, a transformer-based approach has been chosen since these approaches tend to perform well for question-answering problems (Shao et al., 2019). To improve the performance of our models, additional data for use in the training process has been generated. In addition, multiple models have been trained using different approaches. Another challenge is that some of the examples require multiple spoilers to be generated as an answer. To address this issue, a post-processing approach to generate multiple relevant answers that are not too similar to each other has been developed. Furthermore, a specific pattern could be identified and used to develop a heuristic in the pre- and post-processing step to find solutions to these examples independent of the model.

2 Background

The task we attempted to solve was as follows: given a post text that introduces a so-called knowledge gap that piques the reader's interest in reading the accompanying article, our objective was to identify the passages within this article that best addresses this knowledge gap.

2.1 Clickbait Dataset

A dataset was provided for solving the task (Hagen et al., 2022) to which we will refer to as the "Clickbait Dataset". This dataset consists of 3200 training samples and 800 validation samples. Another 1000 test samples were withheld and were only accessible through the evaluation portal TIRA. Each of these samples contains a *postText* which is the text of a clickbait post. A clickbait post refers to an article that shall be read. Therefore, the title and also the text of the article are provided by the sample. Note that the title and the clickbait post might be the same, but that mostly depends upon the author of the post whether they judged if the title is interesting enough or not, i.e. it introduces a knowledge gap that makes the reader want to open the article to fill the gap. Each sample was additionally labeled with a spoiler, i.e. one or more passages from the article that were judged as filling the knowledge gap. Those spoilers were classified into 3 categories, which are the following:

1. **Phrase:** A single spoiler that consists of at most 5 words
2. **Passage:** A single spoiler that consists of more than 5 words
3. **Multi:** A collection of spoilers that consists of any number of words

Proceedings of the The 17th International Workshop on Semantic Evaluation (SemEval-2023), pages 969–977

A label that defines this desired spoiler type was therefore provided as well.

3 System Overview

For this task, a typical BERT-architecture consisting of a tokenization- and an encoding step was extended by a task-specific post-processing step. As mentioned in previous sections, we regard this task as a kind of question-answering problem. Consequently, we interpret the post-text as a kind of question with regard to the article due to the Knowledge-Gap it contains. Therefore, we fine-tuned a pre-trained transformer for question-answering. In the following, we will describe how those models were trained and then how the mentioned post-processing step was designed.

3.1 Pre-processing & Additional Data

To have more data available to train the models upon, we created the HuffPuff dataset. It consists of 190,168 articles which were scrapped from the Huffington Post website[1]. This website was selected as the representative source for clickbait-style examples, based on its disproportionate representation in the original dataset. However, the HuffPuff dataset is unlabeled, i.e. it doesn't contain any gold labels that could be used for training our model for question answering. Therefore, we used multiple approaches to automate the labeling process.

3.1.1 Cloze-Style Post Generation

Many clickbait posts follow a template that is similar to Cloze-Style Questions, such posts we will call Cloze-Style Clickbait Posts (CSCP). Cloze-Style Questions are sentences in which missing spans of text need to be predicted or rather retrieved from the text. Such Cloze-Style Questions have already been shown to achieve significant improvements when training with them for regular question-answering on Wikipedia articles. Dhingra et al. were able to exploit the structure of such articles to create large amounts of labelled data (Dhingra et al., 2018). However, the news articles we scrapped differed in structure from Wikipedia articles such that we needed to change the generation approach. Additionally, we found the idea of mechanically creating CSCPs quite promising for our scenario.

[1] https://www.huffpost.com

In CSCPs, hypernyms get used instead of blanks to create an interesting knowledge gap for the reader.

> You need to see **this Twitter account** that predicted Beyoncé's pregnancy
> **This Guy** Cheated On 'The Price Is Right' And Forced Them To Change Their Entire System

For the generation of such posts, we created the following pipeline:

1. A Named Entity Recognition (NER) model identifies Named Entities (NE) in the title.
2. When one is found, it gets looked up whether the same named entity can be found in the article.
3. If the NE was found in the title as well as in the article it was replaced in the title with a related hypernym (e.g. a person's name gets replaced with "`this person`") and the first appearance of the NE in the article was assumed to be the gold spoiler.

With this approach, we were able to create 79,136 newly labeled samples, which look like the following:

> Making **this food**: A Trend That Keeps On Trending
>
> Experts Answer "Why Isn't **this person** More Famous?!"

3.1.2 Enumeration Spoiler

Among the scrapped articles, a small portion of samples follows an enumeration structure i.e. those articles are similar to a list of multiple viable spoilers extended by an explanation, description, etc. We used this structure to generate 11,721 new Mutli-Spoilers. The exact process is described in more detail in Section 3.4.
However, we noticed while training that those samples did more harm than good. We assume that this is because the heuristics we used to identify the Multi-Spoilers were too obvious to the model, such that it over-fitted on those heuristics rather than finding new patterns and generalizing.
Therefore, we employed a strategy we call "Cover Your Tracks" (CYT) with which we simply remove from those samples all the reference points we have used to identify those spoilers. For this, we used

a CYT-Factor to define from how many samples those reference points shall be removed. For example, a CYT factor of 0.4 implies that we remove from 40% of all samples the reference points. We found that when employing this CYT-Strategy (with a sufficiently high CYT-Factor to mitigate the negative effects of the Enumeration Data) the impact on the results still fluctuates, i.e. sometimes it has a positive impact on the final model performance and sometimes a negative one. The effects depend strongly on the configuration of the other hyper-parameters, as described in Appendix A.1.1. We assume that this is because the rules and patterns that get learned through this kind of training are already hard-coded in our post-processing step, as described in Section 3.4. This means that it only helps in the rare cases where either the post-processor fails or when similar patterns appear in other contexts, like while predicting phrase-spoilers, which is rare but not impossible.

3.2 Model

For our study, we employed a DeBERTa model that had already undergone pre-training on a large corpus of English texts to serve as our underlying language model. Additionally, we found that a specialization of the models by the spoiler-type they are meant to predict can lead to significantly improved results, due to allowing the models to further specialize. Accordingly, we developed and trained three distinct models, each designed to predict one of three specific types of spoilers: *Phrase*, *Passage*, or *Multi-Spoilers*. Consequently, those models were only trained on the prediction of the spoiler-types they are meant to predict. We then did some lightweight hyper-parameter tuning to find the best configurations for our models' hyper-parameters and how to use the additional data which we provided. While doing so, we found that both the phrase and multi-spoiler model profited from the generated CSCP and enumeration data. We then used those datasets in a step we call pre-tuning, i.e. a fine-tuning step which is applied between the pre-training on generic text data and the final fine-tuning on the high-quality manually labelled Clickbait data (Hagen et al., 2022). Additionally, we found that a BERT model outperformed the DeBERTa model on the task of predicting multi-spoilers. We replaced therefore the DeBERTa model in the model for predicting the multi-spoilers, for the prediction of the other spoilers DeBERTa has been found to outperform BERT by a significant amount. Further details are described in Appendix A.1.

3.3 Post-Processor

To transform the output of the model into a textual spoiler, we use a task-related post-processor. A schematic overview is visualized in Figure 4. Depending on the spoiler type, different approaches are used to create the spoiler. In the case of a phrase or passage-spoiler, the goal is to find only a single continuous snippet from the article. Since the output of the model, called *logits*, describe how likely a token of the text is at the start or the end of the spoiler respectively, it is easy to find a sorted list of spoiler candidates. Then the most likely spoiler is returned which satisfies further constraints such as not containing a line break. Additionally, if the spoiler type is a phrase, the spoiler can not contain more than five words. Similarly, the spoiler for passages must contain at least six words. In the case of a multi-spoiler, another distinction is made regarding whether the article is in an enumeration-like form. Those multi-spoilers containing enumerations can be automatically detected, as described in detail in the next section. Then a rule-based approach is used to identify the enumeration spoiler text. In case this approach fails, as well as for all other multi-spoilers, the multi-spoiler entries are generated iteratively. In each iteration, a spoiler candidate is created using the same process as for phrase and passage-spoilers (without text length constraints). If this candidate adds new information to the spoilers already found, this candidate will be added to the spoiler list. The way similar spoilers are identified is explained in detail in Appendix A.2. After a maximum of seven iterations or with five spoilers found, the generation is finished. Otherwise, the logits are modified to disable the tokens of the current spoiler in the next iterations.

3.4 Enumeration Spoiler Prediction

The multi-spoilers proved to be the most difficult type of spoiler to correctly predict using the model. This led to an investigation into approaches to support the model, enabling it to make better predictions. The basis for these approaches would be a heuristic based on some pattern identified in the data.

3.4.1 Enumeration Type Questions

> **7 Things** We Know About the New Season of Stranger Things

53 out of 143 multi-spoiler samples in the validation data follow a pattern like the example above. This represents a considerable portion of the multi-spoiler samples and therefore provides an opportunity to develop an effective heuristic. The main focus here is the cardinal number preceding the object of the sentence. Spacy (Honnibal et al., 2020) was used to detect and extract the cardinal number from these examples. This information could be used to identify that a certain format of the answer is expected, where a number of similar points are simply enumerated. A fourth tag was added to these samples, identifying them as both multi-spoilers and enumeration type spoilers. Additionally, the extracted number directly informs about the number of expected answers.

3.4.2 Enumeration Spoiler Generation

Furthermore, explicit enumerations can also be identified in the article text by using Spacy (Honnibal et al., 2020) to look for cardinal numbers, followed by some form of punctuation. In the next step, ascending or descending sequences of these cardinal numbers were identified. Afterward, a short passage of text following the cardinal number was returned. This method was used to generate multi-spoilers for all samples that were labelled as enumeration type questions. In the following example the detected cardinal number is printed in bold, followed by the identified spoiler in italic.

> 29 of the most beautiful TV quotes of all time:
> **1.** *"You can't live your life according to maybes."* – Poussay Washington, Orange Is the New Black

A further use case was found in the generation of new training data. As explained in Section 3.1.2, the additional HuffPuff dataset is unlabeled. However, with this algorithm we are able to search the dataset for enumerations, extract multi-spoilers and label the data with these multi-spoilers as desired solutions.

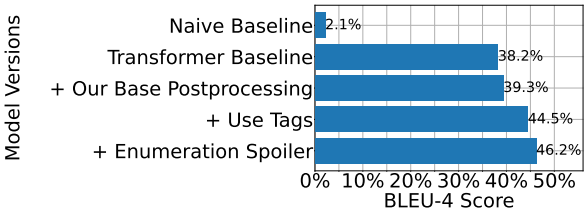
4 Experimental Setup

The training for all our models was done using one A100 GPU. Multiple models were trained for each version of the model, and afterwards, the best performing model on the validation data was selected. To evaluate the performance of a model, we used the BLEU-4 score (Papineni et al., 2002) which is a similarity measure for strings using cumulative n-grams up to $n = 4$.
The models were trained with different combinations of input data. Furthermore, for some models, hyper-parameter tuning was performed (see Table 5).

For running the spoiler prediction, a trained model is required. Our models were packaged together with the code into docker containers and then submitted to TIRA (Fröbe et al., 2023b).

The following evaluation in Section 5 is based on the validation Clickbait Dataset containing 800 samples.

Figure 1: Evaluation of our post-processing based on the transformer baseline model.



5 Results

Two baselines are available for comparison (Fröbe et al., 2023a): a naive one that returns the title of the article as a spoiler and a transformer-based one that, like ours, uses the question-answering approach. Figure 1 shows the results of these baselines as well as the evaluation of our post-processing using the transformer baseline model. The transformer baseline (38.2%) significantly outperforms the naive one (2.1%). Overall, our post-processing improves the transformer baseline to a BLEU-4 score of 46.2%. Most of the improvement is caused by the use of the spoiler tags (text length limit and multi-spoiler iterations) and the rule-based enum-spoiler generation.

Additionally, we evaluated the performance for two of our released versions compared to the transformer baseline model using our post-processing. The first version uses only a single DeBERTa-model for all spoiler types, and is therefore universal. In contrast, the second version uses a specialized model for each spoiler type (DeBERTa-model

Figure 2: Evaluation of different models on different subsets of the validation dataset.

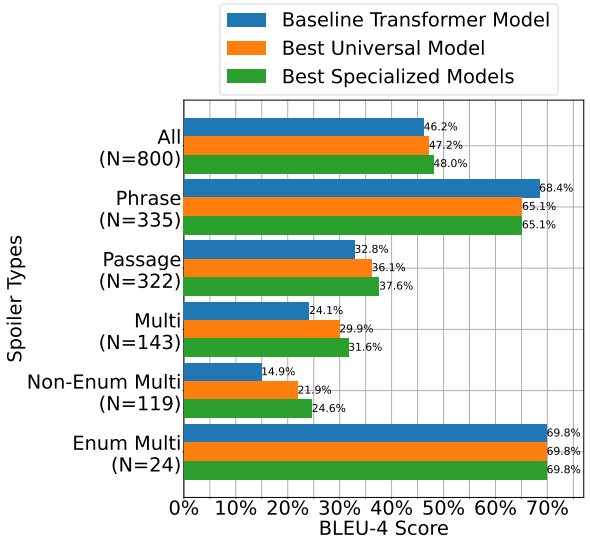
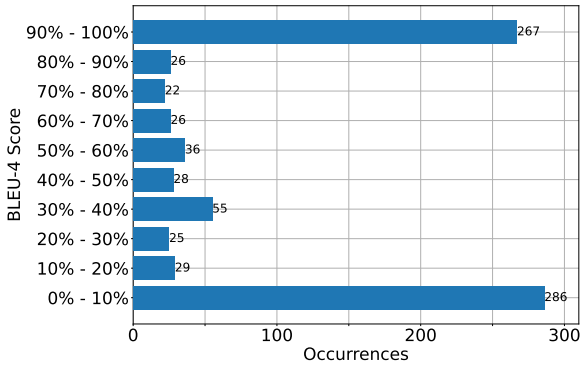


Figure 3: The distribution of the BLEU-4 score on the validation dataset for our best model.



for phrase and passage, BERT-model for multi). The results for different subsets of the validation dataset are shown in Figure 2.

Both the universal model and the specialized models improve the baseline with our post-processing to 47.2% and 48.0% respectively. The advantage of using specialized models can be clearly seen for passage and multi-spoilers. However, in the case of phrase-spoilers, our specialized model could not improve the universal one and is even worse than the baseline. The enumeration spoiler generation achieves a BLEU-4 score of 69.8% on the relevant 24 samples. Compared to the otherwise low scores for multi-spoilers, this represents a remarkable improvement.

An interesting property related to the distribution of the BLEU-4 score across each sample of the validation dataset is shown in Figure 3. Our model finds (almost) perfect spoiler texts in about one third of the samples. On the other hand, another third performs really badly. The remaining samples are almost evenly spread between 10% and 90% BLEU-4 score. Further results based on the test dataset can be found in Appendix A.5.

6 Conclusion

With all the improvements applied to our baseline, we were able to increase the BLEU-4 score for our predictions by 9.8 percentage points from 38.2% to 48.0%.

Using specialized models for different spoiler types provides the opportunity to configure the training process for each model individually. The presented post-processing approach leverages model outputs to improve on these results further. Even on the transformer baseline, this approach shows a solid improvement of 8 percentage points in the BLEU-4 score. Adding in the enumeration spoiler generation has proven to provide a considerable improvement in the results. Overall, the use of transformer-based language models show solid results for this task. Combining this approach with additional heuristics for specific cases and creating a composite approach shows great promise.

A number of opportunities exist to improve this approach even further. For example, at the moment the enumeration spoiler generation as part of the post processing is only performed on samples tagged as enumeration type questions. Refining the heuristic and applying it to all samples could yield further improvements (see Appendix A.4).

An additional possibility to enhance the performance of the model is improving the dataset further, e.g. by using a better NER model that can differentiate more classes than the one we used. By checking for classes like *Actors*, *Tips*, and *Twitter Accounts* we could find more CSCP candidates and create a larger, as well as more diverse CSCP Dataset, which represents the official CSCPs better.

7 Acknowledgments

We are grateful to the Center for Information Services and High Performance Computing [Zentrum für Informationsdienste und Hochleistungsrechnen (ZIH)] at TU Dresden for providing the HPC resources for this project.

References

Bhuwan Dhingra, Danish Pruthi, and Dheeraj Rajagopal. 2018. Simple and effective semi-supervised question answering.

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.

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.

Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. 2020. spaCy: Industrial-strength Natural Language Processing in Python.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation.

Taihua Shao, Yupu Guo, Honghui Chen, and Zepeng Hao. 2019. Transformer-based neural network for answer selection in question answering. *IEEE Access*, 7:26146–26156.

A Appendix

A.1 Model Training Configurations

We employed a hyper-parameter tuning step to find out how to best use the training data that was available to us. For this we performed an exhaustive search over the search space defined in Table 1 (With some pruning to ignore some, to us, obviously uninteresting configurations like different values for the CYT-Factor when no Enumeration Data was used). Each configuration was trained multiple times and averaged to reduce random error. Through this step, we ended up with the following discoveries and strategies to train our specialized models.

Table 1: Search space for hyper-parameter tuning. *Model* refers to the underlying language model that was used. *Train Epochs* refers to the number of epochs the model was trained upon the Clickbait Dataset. The *Cloze* and *Enumeration Epochs* refer similarly to the number of epochs the model was trained on the CSCP and Enumeration data respectively. *CYT* refers to the CYT-Factor that was employed when trained on enumeration data. *Use Only* refers to the degree of specialization all data underwent before letting the model train upon, i.e. *Phrase* implies the model was only trained on phrase-spoilers (at most 5 words long), the same applies to *Passage* and *Multi*. The option All means that there was no exclusion at all.

	Options
Model	BERT, DeBERTa
Train Epochs	1,2,3,4,5
Cloze Epochs	0,1,2
Enumeration Epochs	0,1
CYT	0,0.3,0.6,0.9
Use Only	Phrase,Passage,Multi,All

A.1.1 Enumeration Spoilers & The Impact of CYT

One of the first discoveries we had due to the hyper-parameter tuning step was that a high value for the CYT-Factor performs best. The CYT-Factor is the factor that defines from how many samples of the Enumeration dataset shall all the reference points be removed. 'Reference points' refers here to anything our heuristic used to identify the enumeration spoilers, like numbers and line-breaks. The impact of different CYT-Factors on an example configuration is shown in Table 2. As can be seen in this table, a CYT-Factor of 0.9 was found to perform best, i.e. we removed from 90% of all samples all reference points.[2]. Even though a high CYT-Factor helps to mitigate the negative effects of the Enumeration Data while preserving the positive ones, the impact of this data still depends largely on what other data was used and which type of spoiler needs to be predicted. Therefore, all the following models use either Enumeration Data with a CYT-Factor of 0.9 or no Enumeration Data at all.

[2]A few CYT factors higher and lower around 0.9, e.g. 0.999, were shortly tested as well but when first results were worse than the ones achieved with 0.9 we decided on not investing further resources and time in getting this value perfect. Therefore, those other values for CYT aren't included in the table

Table 2: BLEU-4 score in percent of a model when trained on no Enumeration Data (-) and when trained on Enumeration Data with different CYT-Values

CYT	Overall	Phrase	Passage	Multi
-	**42.3**	**61.5**	**35.0**	13.7
0.0	35.4	52.9	27.5	12.4
0.3	39.9	59.5	31.4	13.5
0.6	40.2	60.4	31.1	13.7
0.9	41.0	60.9	32.2	**14.4**

Table 3: BLEU-4 score in percent of a model when trained without and with Cloze-Style Post Data.

	Overall	Phrase	Passage	Multi
No Cloze	**36.4**	62.1	**21.0**	**10.5**
Cloze	36.1	**63.1**	19.7	9.9

A.1.2 Phrase-Spoiler-Model

As can be seen in Table 3 and 4, we found that pre-tuning the Phrase-Spoiler-Model on the Cloze-Style Post dataset let to improved performance for predicting phrase-spoilers. Further pre-tuning on Enumeration Data had minor positive effects as well, despite its difference in style to phrase-spoilers. We observed that Enumeration Data was most helpful when only using multi-spoilers that were also phrase-spoilers, i.e. the ones that contain at most five words. We assume that this minimized the negative effects of generalization while preserving the positive ones. Therefore, we pre-tuned this model on the Cloze-Style Post dataset for one epoch, then for another epoch on the Enumeration Data with a CYT-Factor of 0.9 as described in Section 3.1.2 and finally we fine-tuned this model on the Clickbait Dataset (Hagen et al., 2022) for an additional epoch.

A.1.3 Passage-Spoiler-Model

For the passage-spoiler prediction, neither the CSCP data nor the enumeration data led to any significant improvements. Therefore, we fine-tuned this model only on the passage-spoilers of the manually labelled Clickbait dataset for 2 epochs. (Hagen et al., 2022)

Table 4: Average performance when predicting Phrase-Spoilers of the best found config (Cloze+Enum) compared to when only pre-tuned on enumeration data, when only pre-tuned on Cloze-Data and without any pre-tuning (-). Note: All configurations were still fine-tuned on the Clickbait Dataset (Hagen et al., 2022).

	-	Cloze	Enum	Both
BLEU-4	62.1%	63.1%	62.3%	**63.7%**

A.1.4 Multi-Spoiler-Model

While tuning our Multi-Spoiler Model, BERT has surprisingly been found to perform better than DeBERTa. After some adjustments[3], we were able to increase the BLEU-4 score on Multi-Spoilers by more than 6 percentage points, i.e. from 20.7% with DeBERTa to an average of 26.9% with BERT. Therefore, we used for the prediction of Multi-Spoilers a large BERT model in the background instead of DeBERTa. This model was then pre-tuned for 1 epoch on the Cloze-Style Post dataset, then for 1 epoch on all the Enumeration Data with a CYT-Factor of 0.9, and finally it was fine-tuned for 2 epochs on the Clickbait dataset. Unfortunately, we found that those high gains in performance strongly correlate with the gains achieved by our post-processing step, which will be explained in the following sections. Therefore, the achieved gain through those adjustments dropped to less than 2 percentage points for our final model.

A.2 Similarity Measures

As mentioned before, in the case of multi-spoilers, where multiple phrases of the text have to be returned, a problem arises. Since the model returns a pool of phrases that are likely to be spoilers, reoccurring phrases in the text may appear multiple times in the pool of predictions, like names or locations. Just choosing the top results of the pool may result in choosing a similar phrase multiple times, while neglecting other possibilities that may not have scored as high as the others in likelihood to contain a spoiler, but contain different information, which would improve the functionality of the model. The process of rooting out similar spoilers starts with the candidate list of spoilers, ordered by their achieved score, high to low, and an empty collection of selected spoilers. Each candidate is compared to all previously selected spoilers and is judged on its similarity to them. If the similarities with all selected spoilers are below a certain threshold, the candidate is added to the collection of selected spoilers. Since the similarity of content was the metric that had to be judged, spoilers

[3] *Adjustments* refers here to switching from DeBERTa to a large BERT and a BERT-Base model that were compared to each other. And performing a drastically shortened hyperparameter tuning step (i.e. we only chose the configurations that were found to perform already well on DeBERTa).

containing the same content in meaning, but different wording should also score high on similarity. Therefore, each spoiler was stripped from unnecessary stop-words, punctuation and capitalization before being compared to each other. To judge how the process of calculating this similarity score would affect the overall performance of the model, multiple methods of producing a score were chosen. Those were:

- Cosine Similarity [4]
- The 'TheFuzz' package [5]
- The Jaccard Index [6]
- The 'Spacy' package [7]
- A combination of all the above

When the impact of these different approaches was explored, we realized it was not as much as believed and 'TheFuzz' was chosen as the go-to method for determining the similarity between spoilers. It is based on Levenshtein-Distance and allows comparing strings in their full length, partially, with the words sorted beforehand and removing recurring words. Instead of choosing one of these methods, we just applied all and used the highest achieved similarity.

A.3 Packages used

In all occurrences of spacy the "en_core_web_lg" model version "3.4.1" was used.[8]
The transformer baseline model used in Section 5 can be found on *Hugging Face*.[9]

A.4 Further Opportunities for Improvement

Given the fact a third of our model predictions are almost correct, a third are almost completely false, and the remaining third are evenly distributed in between (Figure 3), developing a method to predict the confidence for each generated spoiler could be useful. It would allow for discarding low confidence spoilers or developing further methods to specifically improve in these areas.
In our current models, the tag is not included in the input for the training process. Adding this information might increase the performance of the models, enabling the model to better distinguish between different types of expected answers. Additionally, this might add additional utility to identifying and tagging enumeration type questions. The number of expected answers generated for enumeration type questions also is of limited utility, since the number of answers for multi-spoiler questions is capped at 5. Therefore, the additional information is only really useful in cases with less than 5 expected answers. This is a property of the data for this task, and little can be done about it.
The enumeration spoiler generation as part of the post-processing has proven to provide a considerable improvement in the results. The heuristic could be expanded to look for different formats of enumeration, or possibly even to detect section headers within the text. A possible approach might be to learn the features for this detection by creating a training dataset for this use case. Currently, the enumeration spoiler generation is only performed on samples previously identified as enumeration type questions. However, enumerations in the text could occur in the rest of the samples as well. Performing the spoiler generation on all multi-spoiler samples could be investigated. Looking for enumerations in these other samples might introduce a source of false positives. However, especially when combined with improving the heuristic as mentioned above, the risk of false positives from performing the enumeration spoiler detection on all multi-spoiler samples could be reduced and the overall score improved.

A.5 More Results

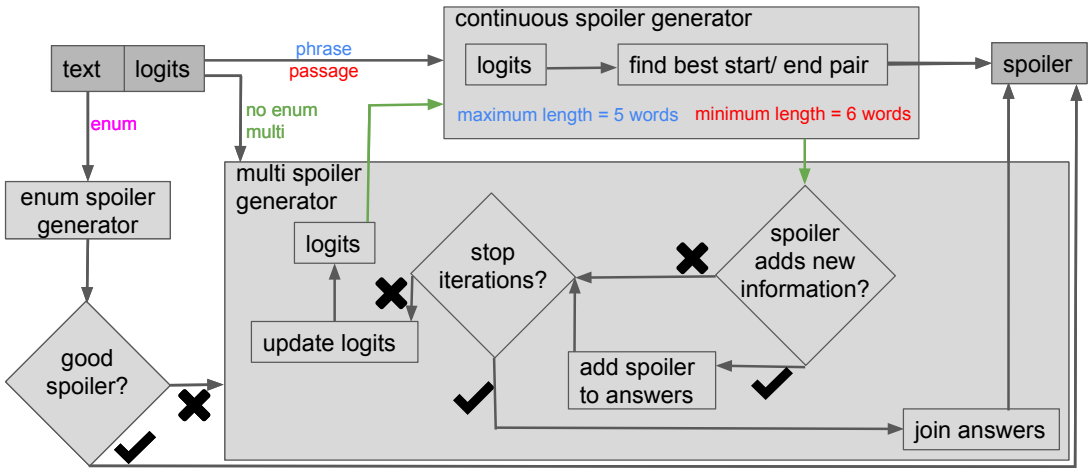
Table 5 shows detailed results for some of our officially submitted models as part of SemEval 2023 Task 5. Note that the models were tested using the test Clickbait Dataset which has been unknown to us while working on the challenge. The fourth submission of Table 5 is equal to the specialized models of Figure 2. Comparing these results to

[4] https://www.machinelearningplus.com/nlp/cosine-similarity/
[5] https://github.com/seatgeek/thefuzz
[6] https://www.statology.org/jaccard-similarity/
[7] https://spacy.io
[8] https://github.com/explosion/spacy-models/releases/download/en_core_web_lg-3.4.1/en_core_web_lg-3.4.1.tar.gz
[9] https://huggingface.co/webis/clickbait-spoiling-with-question-answering/tree/debertalarge-all-cbs20-both-checkpoint-1200
[10] https://huggingface.co/microsoft/deberta-v3-large
[11] https://huggingface.co/bert-large-uncased

Table 5: Evaluation of the different submitted models for subtask 2 at SemEval 2023 Task 5. The used metrics are BLEU-4 (BL4), BERTScore (BSc.) and METEOR (MET). Multiple tests were conducted over all clickbait-posts as well as over the three subsets that only use phrase, passage, or multi-spoilers from the test set (N=1000). The used language models are a large DeBERTa model (DB) [10] and a large BERT model (B) [11]. All models use multiple specialized models except the first one which uses the same model to predict phrase-, passage- and multi-spoilers alike. The last two models are the result of a hyperparameter-tuning process (HPT) the previous ones not. The models are trained on training data (T), validation data (V), Cloze-Style Post data (C) and enumeration data (E).

Submission			**All**			**Phrase**			**Passage**			**Multi**		
Models	HPT	trained on	BL4	BSc.	MET	BL4	BSc.	MET	BL4	BSc.	MET	BL4	BSc.	MET
B	no	T	0.26	0.89	0.26	0.42	0.92	0.44	0.16	0.88	0.31	0.08	0.85	0.18
DB-DB-DB	no	TC	0.46	0.93	0.52	0.64	0.95	0.70	0.34	0.91	0.49	0.30	0.90	0.53
DB-DB-DB	no	TVC	0.47	0.93	0.53	0.64	0.95	0.68	0.36	0.91	0.51	0.30	0.90	0.55
DB-DB-B	yes	TCE	0.47	0.93	0.51	0.65	0.95	0.70	0.35	0.91	0.51	0.30	0.90	0.47
DB-DB-B	yes	TVCE	0.47	0.93	0.50	0.64	0.95	0.69	0.36	0.91	0.51	0.30	0.90	0.45

Figure 4: A schematic overview of the post-processing which transforms the original text and the logits of the model into a textual spoiler.



those from Figure 2, which are based on the validation Clickbait Dataset, the BLEU-4 score is slightly worse, but overall the performance of the last four submissions is about the same on validation and test data. Therefore, our models are not overfitted.