John Boy Walton at SemEval-2023 Task 5: An Ensemble Approach to Spoiler Classification and Retrieval for Clickbait Spoiling

Maksim Shmalts
Department of Linguistics, University of Tübingen
maksim.shmalts@student.uni-tuebingen.de

Abstract

Clickbait spoiling is a task of generating or retrieving a fairly short text with a purpose to satisfy curiosity of a content consumer without their addressing to the document linked to a clickbait post or headline. In this paper we introduce an ensemble approach to clickbait spoiling task at SemEval-2023. The tasks consists of spoiler classification and retrieval on Webis-Clickbait-22 dataset. We show that such an ensemble solution is quite successful at classification, whereas it might perform poorly at retrieval with no additional features. In conclusion we outline our thoughts on possible directions to improving the approach and shape a set of suggestions to the said features.

1 Introduction

Clickbait is a term for a descriptive piece of information (a headline, a snippet, a post in social media etc.) aiming to arouse users' interest by summarizing the contents of a linked document in a deceptive and/or misleading manner. The summary does not suffice to satisfy users' curiosity, whereas the linked document is in a way advertized to do so, which encourages viewers to click the link (and thus cover the arisen curiosity gap). Using clickbait is frowned upon as the contents of the linked page does not usually correspond to the expectations of users and in some cases may be inappropriate, offensive or harmful. Hagen et al. (2022) address the issue with an idea of clickbait spoiling: 1) predicting the type of a spoiler that would satisfy users' curiosity and thus withhold them from following the link; 2) retrieving the spoiler itself from the document. Upon developing a solution Hagen et al. (2022) compiled a dataset containing "5,000 "spoilable" clickbait posts" in English collected from Twitter, Reddit, and Facebook, that then were manually spoiled and classified: The Webis Clickbait Spoiling Corpus 2022 (Webis-Clickbait-22). SemEval-2023 Task 5 suggests to introduce a model for clickbait spoiling and bring new insights to the existing approaches, basing on Webis-Clickbait-22 and (optionally) additional resourses (Fröbe et al., 2023).

Our team presents an ensemble solution called WebSemble (**Web**is-Clickbait-22 En**semble**) mainly following the steps outlined in Hagen et al. (2022) except for an additional (and optional) summarization of the documents before the classification. Below we briefly list actions undertaken with our approach upon execution (more in section System Overview):

1. Data splits are transformed into SQuAD1.1 (Rajpurkar et al., 2016) format and preprocessed;
2. The contents of the linked web pages are summarized if summarization is used; summarization is conducted with a single model;
3. The clickbait posts are spoiled:
   3.1. Types of spoilers are predicted from either summarized documents (if summarization is used) or their titles; whether summarization is used is one of the parameters of our model;
   3.2. For each dataset entry, top k spoilers are retrieved from the web page contents;
4. Retrieved sets of k spoilers are postprocessed with respect to predicted labels.

Note that the ensemble approach mentioned above comes into action only at steps 3.1 and 3.2. For that, groups of either taken "out of the box" or fine-tuned on Webis-Clickbait-22 dataset transformers (Wolf et al., 2020) models for text classification and question answering (QA) make separate predictions that are afterwards collapsed into a single ensemble decision.

Implementations of WebSemble within SemEval-2023 Task 5 were submitted to TIRA (Fröbe et al., 2023) dockerized. Source code of the latest version is open and available under https://github.com/cicl-iscl/SemEval23-Task5-John-Boy-Walton. Dockerized environment can be pulled from https://hub.docker.com/repository/

Proceedings of the The 17th International Workshop on Semantic Evaluation (SemEval-2023), pages 2100–2106

docker/maxschmaltz/websemble (tag 0.28.amd).

2 Related Work

In our research we mostly oriented ourselves at Hagen et al. (2022). That approach put a significant value on the fact that spoilers can generally be divided into two classes: phrase and passage. And so the authors pay much attention to predicting the label type, depending on which the next steps are forked into two branches, where the spoilers are retrieved differently: phrases are predicted in terms of question answering in SQuAD1.1 format, and passages are obtained with passage retrieval models.

Liello et al. (2022) present three novel objectives for pre-training transformer-based models, that aim to analyze sentence- and paragraph-level semantics "within and across documents". Even though it is not related to clickbait spoiling directly, we reckon it might be quite helpful for improving existing and future clickbait spoiling models, since answer sentence selection models pre-trained with the objectives 1) have a higher performance; 2) are capable of defining information about whether a sentence belongs to a certain paragraph.

We believe the distinction between phrases and passages is meaningful and justified enough yet given a slightly heavy weight. We try to show that spoilers might be obtained within a single pipeline agnostic to their types and that the type is more important during postprocessing rather than upon spoiler generation.

WebSemble refers to the following models:

Summarization:

- `Pegasus` (Zhang et al., 2019)

Text classification:

- `bert-base-uncased-MNLI`: a fine-tuned on MNLI (Williams et al., 2018) `BERT base model (uncased)` (Devlin et al., 2018)
- `deberta-v3-base-tasksource-nli` (Sileo, 2023)
- `DistilBERT base model (uncased)` (Sanh et al., 2019)

Question answering:

- `BART (base-sized model)` (Lewis et al., 2019)
- `bert-large-uncased-whole-word-masking-finetuned-squad` (a fine-tuned on SQuAD1.1 `BERT large model (uncased)`)
- `distilbert-base-cased-distilled-squad` (a fine-tuned on SQuAD1.1 `DistilBERT base model (cased)`)
- `roberta-base-squad2` (a fine-tuned on SQuAD2.0 (Rajpurkar et al., 2018) `RoBERTa` base model (Liu et al., 2019))

In future we might want to use answer sentence selection models pre-trained with the objectives proposed in Liello et al. (2022) for dealing with mulipart spoilers (more in Conclusion).

3 Task Setup

3.1 Corpus

In our solution we use no additional data and stick to Webis-Clickbait-22 dataset. The dataset includes 5000 clickbait posts crawled from Twitter, Reddit, and Facebook, with the main contents and titles of the linked web pages, manually retrieved spoilers and their types and a large amount of optional extra fields referring to additional information such as urls of the web pages, key words, metadata etc. All entries are in English and no data in another language is supposed to be provided for the task. Software is expected to yield only spoiler types for the classification subtask and textual spoilers for the retrieval subtask. Below is a simplified example of prototypical test input and output (human-readable):

```
Test input: {
  "uuid": "87239hf373f",
  "postText": ["This is how to set up
  a startup in 3 month"],
  "targetParagraphs": ["Twice a year
  Y Combinator invests $500,000 per
  company in a large number
  of startups.",
  "You can now apply for S2023 batch
  at Y Combinator!"],
  "targetTitle": "Y Combinator created
  a new model for funding
  early stage startups",
  "targetUrl": "https://www.yc.com"
}

Output: {
  "uuid": "87239hf373f",
  "spoilerType": "phrase",
  "spoiler": "apply for
  S2023 batch at Y Combinator"

}
```

Webis-Clickbait-22 introduces 3 types of spoilers, namely, "phrase": a shorter single-span spoiler, "passage": a longer single-span spoiler, "multi": multiple-span spoiler. The dataset is not balanced by the classes and counts 2,125, 1,999 and 876 class entries respectively. Lastly, SemEval-2023

Task 5 divides data as follows: 3200 entries for training, 800 entries for validation and 1000 entries for test (inaccessible for the participants).

3.2 Baselines and Preprocessing

The participants of the tasks are provided with naive and transformers baselines. The baselines themselves are not used with our team as they are extremely simple (which, however, will not be discussed in the paper) but manage to prepare a firm ground for preprocessing. As the first stage, transformers baseline for the retrieval subtask transforme Webis-Clickbait-22 dataset (see the structure above) into SQuAD1.1 format and then tokenize the textual data. We decided to keep the output format since it fits perfectly the input structure our models expect to be passed. The only edit we make is incorporating spoiler type labels directly into that dataset in order not to create a second one for classification but rather to use a single one for both of the subtasks. To summarize, after preprocessing our main features are: tokenized contents and title of the web pageg; spans of the spoilers.

4 System Overview

As mentioned in Introduction, WebSemble takes the following steps: 1) preprocessing (described in Baselines and Preprocessing); 2) optional summarization with a single model; 3) spoiler type prediction with an ensemble of text classification models; 4) spoiler retrieval with an ensemble of question answering models; 5) postprocessing. In this section steps 3-5 will be discussed; step 2 will be omitted because the model is used "out of the box".

4.1 Ensemble Configuration

The behaviour of WebSemble is configured by a set of JSON files, so-called instructions. Each instruction corresponds to a model and defines whether it is used and, if so, whether and with which parameters it should be fine-tuned (here and further: on Webis-Clickbait-22 dataset) and applied. The fields of instructions with descriptions are provided in Appendix.

4.2 Models

Models overview was given in section Related Work. However, not all the models are being fine-tuned:

For each model, training goes on the train split of the dataset 2500 to 5000 steps with validation being executed on the validation split every 250 steps; batch sizes for training and validation are equal within one model but may vary for different ones: that depends on how computationally intensive the models are (the more computationally intensive the model is, the least the batch size for it is). Some models we use are taken already fine-tuned for the respective task (e.g. *distilbert-base-cased-distilled-squad* is already fine-tuned for QA) and are not fine-tuned further on Webis-Clickbait-22 dataset to prevent overfitting. Accuracy and BLEU score are chosen as validation metrics for the classification and retrieval subtasks respectively. Models in the latest version were fine-tuned on a 14GB GPU in a Google Colab notebook.

model	fine-tuned	epochs	batch size
bert-base-uncased-MNLI	True	6.25	4
deberta-v3-base-tasksource-nli	False		
DistilBERT base model (uncased)	True	12.5	8
BART (base-sized model)	True	12.5	8
bert-large-uncased-whole-word-masking-finetuned-squad	False		
distilbert-base-cased-distilled-squad	False		
roberta-base-squad2	False		

Table 1: Numbers of Fine-tune Epochs.

4.3 Ensemble Approach

For both subtasks we use the same approach to calculating joint ensemble predictions. All models are applied independent from each other and, since we work within PyTorch framework, return tensors of logits that are yet to be postprocessed. First the tensors are truncated/padded if necessary and then collapsed into a single tensor, respresenting the desicion of an ensemble, by calculating mean values. That means no weightings are applied and no model has a priority.

4.4 Postprocessing

One of the key features of our approach is referring to predicted spoiler types not before, but rather after spoilers prediction.

The text classification ensemble output does not require any sophisticated postprocessing: we retrieve labels as indices (that correspond to ids of the labels) with the greater logit values.

The question answering ensemble returns two tensors of logits: they correspond to start positions and end positions of the characters (not tokens) in the document contents respectively. Iterating over pairs of start and end positions logits, we can cal-

culate the scores of pairs and, hence, get a ranking of spoiler spans, that is, spoilers themselves. Now looking at the ranking, we can often see a couple of challenges:

1. A considerable amount of spoilers is meaningless. Those can be empty strings, sequences of punctuation marks, fragments of actual words etc.

2. A lesser yet noticeable number of spoilers makes sense but does not satisfy a form of a "good" sentence. For instance, there can be quite a few spoilers where a bracket was never closed.

3. There are spoilers that should be cleaned in a sense they may contain residuals such as excessive whitespaces and punctuation marks.

In our approach we deal with those issues with heuristics, mostly regular expressions. We prefer to remove spoilers from categories 1 and 2 above for they are hardly human-comprehensible; from spoilers from category 3 we cut off said residuals. The only ill-formed type of spoilers we have to keep for now is a fragment of a token: we do not see an adequate way to check meaningfulness heuristically, so those cases are not being dealt with yet.

After cleaning the spoilers we can proceed further, where we face another issue. Upon iteration over spoilers we encounter numerous overlapping spans, which we might not want to keep as they often differ only by a couple of tokens (e.g., "single word" vs "a single word"). That is where predicted spoiler types might come into action. Since passages are believed to be longer sentences, hence, be captured by longer spans, we resolve overlapping spans greedily if we need to predict a passage. That means in that case from two overlapping spans we would want to pick the longest one. However, that does not work in reverse with phrases, because in that case most phrase spoilers turn out to be one-character strings.

From the remaining spoilers the postprocess function takes top k (default to 5) scored ones and in case of label "multi", random n of them are concatenated at current state of WebSemble; a more sophisticated way to treat multipart spoilers will be proposed in Conclusion.

5 Experimental Setup

As we make software submissions, we must include all models into the environment. That leads to a necessity to fine-tune models locally and incorporate them into image only after. For details of fine-tuning refer to Models. Required dependencies (versions specified) can be found in project repository under `https://github.com/cicl-iscl/SemEval23-Task5-John-Boy-Walton`.

WebSemble takes multiple arguments that are described both in repository README and in Appendix.

6 Results

6.1 Quantitative Assessment

To investigate how different arguments and/or instructions impact on performance of WebSemble, we explore the space of configurations that can be constructed with 4 main parameters:

- Whether summarization is conducted;
- Models used within the ensembles;
- Number of fine-tuning steps (where applicable);
- Whether heuristics specified in Postprocessing are used.

Not all possible combinations are being tested for several reasons. Submitting different configurations on TIRA revealed that the only GPU available there is too small for summarization with Pegasus so we have to omit this parameter. Concerning ensembles, we prefer that the QA ensemble consists of the same models in every configuration. Besides, we increase/lower the number of fine-tuning steps simultaneously for all models. Therefore, not even a half of the space is explored so far.

Judging by metrics obtained in TIRA on the test split (2), we can briefly depict the impact of the parameters on WebSemble performance (omitting summarization). Adding fine-tuned models significantly increases balanced accuracy: cf. accuracy 0.322 with a single not fine-tuned `deberta-v3-base-tasksource-nli` vs accuracy 0.577 with an ensemble consisting of it and two fine-tuned at 5000 steps models. Now the impact of the number of steps is unclear because we do not have two runs with the same models but different number of steps; increase of accuracy from 0.562 to 0.577 with a greater number of steps for `DistilBERT` might as well be explained by adding to it `bert-base-uncased-MNLI` and/or `deberta-v3-base-tasksource-nli`.

In opposite to our intuitions, removing overlapping spans and using heuristics in general seriously harms BLEU score. With this regard the last row of

the table is representative: since predicted spoiler types are used for spoilers retrieval only if heuristic postprocessing is enabled (which is the case), it is remarkable how even with the best spoiler class predictions BLEU collapses by more than 0.2 (0.081 to 0.059).

The motivation to test exactly these three configurations is that they cover the space of configurations in the most compact and representative way; the three combinations involve the three parameters listed at the beginning of the section (except for whether summarization is conducted) so that we may make observations and assumptions about how different parameters of WebSemble impact its performance with the lowest computational costs. Therefore, even though we do not have metrics on such a configuration (yet), from these three tests we may assume the best combination of parameters above would be 5000 fine-tuning steps, the whole ensemble for each subtask but no heuristics.

configuration (no summarization)	**balanced accuracy**	**BLEU score**
2500 fine-tuning steps or less		
DistilBERT base model (uncased) *+ the whole spoiler retrieval ensemble*	0.562	**0.081**
5000 fine-tuning steps		
deberta-v3-base-tasksource-nli (NB! No fine-tuning) *+ the whole spoiler retrieval ensemble* *+ heuristic postprocessing*	0.322	0.039
+ bert-base-uncased-MNLI *+ DistilBERT base model (uncased)*	**0.577**	0.059

Table 2: Balanced Accuracy and BLEU Score with Different WebSemble Configurations

6.2 Qualitative Assessment

Comparison of WebSemble performance on the two subtasks reveals a gap in its ability to solve the given subtasks at a comparable measure. When balanced accuracy at the classification task lies approximately in the middle of the teams' results range, BLEU score at the spoiler extractive generation subtask leaves concerns about whether the model can be applied for it. This may be related to inability of heuristic and overlapping filters to detect proper sentences and arouse a necessity to re-estimate their usefulness and move towards more sophisticated methods. As WebSemble cannot detect grammatically correct and meaningful patterns, it lets sentences, that would not have been accepted by a human, pass. There are some examples from predictions on the validation split (uuid and spoiler type are omitted):

```
[
    "For the",
    "le And Other",
    "or so in"
]
```

However, it would be unfair to assume WebSemble cannot generate adequate spoilers at all:

```
[
    "He Rescued This Bizarre Creature
    From a Sidewalk, But He Had
    No Idea What It Would Grow Into",
    "Loyola Marymount University",
    "guest speakers"
]
```

7 Conclusion

At SemEval-2023 Task 5 we developed an ensemble-based model aimed to spoil clickbait in four stages (preprocessing excluded): 1) optional summarization; 2) spoiler type prediction; 3) top k spoilers retrieval agnostic to predicted spoiler types; 4) top k preprocessing with respect to spoiler classes. We show that this approach is capable of adequate spoiler type prediction, but in some cases fails to postprocess retrieved spoilers in a way the result could be accepted by a human, which encourages us to try different implementations and conduct further research.

We see the following potential courses of further development:

- Replacing heuristic spoiler filters with an advanced model pre-trained to predict grammatical and semantic acceptability of a sentence (in a way, determine if a sentence is "good");
- Grid search aimed to define which models should be used and which ones not; some models can have a strongly negative effect on the whole ensemble decision, which can be hard to detect;
- Weighting of single models outputs: ensemble quality could profit if some models had less "votes" in the joint decision;
- Using answer sentence selection models pre-trained on objectives proposed in Liello et al. (2022): given top k predictions and label "multi", picking spoilers that relate to the clickbait post at most;
- Exploring additional fields of Webis-Clickbait-22 dataset such as key words, platform etc. Trying to benefit from web scraping web sites cited as relative.

References

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805. Hugging Face model card: `https://huggingface.co/bert-base-uncased`.

Maik Fröbe, Matti Wiegmann, Nikolay Kolyada, Bastian Grahm, Theresa Elstner, Frank Loebe, Matthias Hagen, Benno Stein, and Martin Potthast. 2023. 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.

Maik Fröbe, Tim Gollub, Benno Stein, Matthias Hagen, and Martin Potthast. 2023. SemEval-2023 Task 5: Clickbait Spoiling. In *17th International Workshop on Semantic Evaluation (SemEval-2023)*.

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.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2019. BART: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *CoRR*, abs/1910.13461. Hugging Face model card: `https://huggingface.co/facebook/bart-base`.

Luca Di Liello, Siddhant Garg, Luca Soldaini, and Alessandro Moschitti. 2022. Pre-training transformer models with sentence-level objectives for answer sentence selection. In *EMNLP 2022*.

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. Hugging Face model card: `https://huggingface.co/deepset/roberta-base-squad2`.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for SQuAD. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 784–789, Melbourne, Australia. Association for Computational Linguistics.

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.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *ArXiv*, abs/1910.01108. Hugging Face model card: `https://huggingface.co/distilbert-base-uncased`.

Damien Sileo. 2023. tasksource: Structured dataset preprocessing annotations for frictionless extreme multi-task learning and evaluation. *arXiv preprint arXiv:2301.05948*. Hugging Face model card: `https://huggingface.co/sileod/deberta-v3-base-tasksource-nli`.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter J. Liu. 2019. Pegasus: Pre-training with extracted gap-sentences for abstractive summarization. Hugging Face model card: `https://huggingface.co/google/pegasus-xsum`.

A Appendix

A.1 Instruction Fields

For the reference to instruction fields see 3. Default label mapping:

```
{
    "phrase": 0,
    "passage": 1,
    "multi": 2
}
```

A.2 WebSemble Arguments

For the reference to WebSemble arguments see 4 or project README.

field	description	value	required by	
			text classification models	question answering models
name	Name of the model to be printed.	any string	True	
use	Whether to use the model.	True/False	True	
fine-tune	Whether to fine-tune the model.			
input_model_path	Path to the model before fine-tuning; ignored if fine-tune is False.	True	True if fine-tune is True, otherwise False	
output_model_path	Path to the model after fine-tuning; model from this path is used for prediction.		True	
training_kwargs	Parameters to be passed to transformers.TrainingArguments; ignored if fine-tune is False.	dictionary with valid parameters	True if fine-tune is True, otherwise False	
trainer_kwargs	Parameters to be passed to transformers.Trainer; ignored if fine-tune is False.			
test_batch_size	Batch size upon prediction	any positive integer.	True	
label_mapping	Mapping from label id to its human-readable representation.	dictionary	True	False
notes	Anything.	any	False	
url	Url to the model card on Hugging Face.	any url	False	

Table 3: Instruction Fields Reference.

argument	description	required/optional	values	default
input_dir	Directory containing .jsonldatasets to be preprocessed. Should obligatory contain train.jsonl (X_train) and validation.jsonl (X_dev) if mode=="train", input.jsonl (X_test) if mode=="test".	required	any valid path	"./webis22_run"
output_dir	Directory to store output in. Output: run.jsonl with joint predictions for the subtask and, if mode=="train", labels.json and top_k.json with labels and top-k spoilers respectively.	required	any valid path	"./out"
subtask	"1" is for the subtask 1 (spoiler classification), "2" is for subtask 2 (spoiler retrieval).	required	one of "1", "2"	"2"
-i, –instructions_dir	Directory containing used models data. Should contain subdirectories /TextClassification and /QAwith models data for subtasks 1 and 2 respectively.	optional	any valid path	"./instructions_local"
-p, –preprocess_mode	"0": preprocess X_train, X_dev and X_test (if provided), aim: for initial training (and prediction); "1": preprocess only X_test (if provided), aim: for prediction after training; "2": no preprocessing, aim: for evaluation and tests. NB! Preprocess in the context means reading and processing raw data; no preprocessing refers to reading preprocessed previously and saved datasets.	optional	one of "0", "1", "2"	"1"
-m, –mode	"train" forces fine-tuning where applicable, whereas "test" skips it and goes directly to prediction.	optional	one of "train", "test"	"test"
-s, –summarize	Whether to use summarized texts for subtask 1 (spoiler classification); otherwise titles are used.	optional	one of "True", "False"	"False"
-oc, –summarize_only_on_cuda	Whether to allow summarization only on CUDA. Ignored if summarize=="False".	optional	one of "True", "False"	"True"
-save, –save_datasets	Whether to save datasets after preprocessing. Ignored if summarize=="False".	optional	one of "True", "False"	"False"
-save_dir, –saved_datasets_dir	Directory to save datasets after preprocessing to. Ignored if save_datasets=="False"or summarize=="False".	optional	any valid path	"./webis22_summarized"

Table 4: WebSemble Arguments Reference.