UNIVERSITY OF CALIFORNIA SAN DIEGO

Spoiler Recognition as Semantic Text Matching

A Thesis submitted in partial satisfaction of the requirements
for the degree Master of Science

in

Computer Science

by

Ryan Tran

Committee in charge:

Professor Julian McAuley, Chair
Professor Arun Kumar
Professor Jingbo Shang

2023

The Thesis of Ryan Tran is approved, and it is acceptable in quality and form for publication on microfilm and electronically.

University of California San Diego

2023

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

SVM	Support vector machine
CNN	Convolutional neural network
BiGRU	Bidirectional gated recurrent unit
IR	Information retrieval
ROC	Receiver operating characteristic
AUC	Area under the curve
MRR	Mean reciprocal rank

ACKNOWLEDGEMENTS

The thesis author would like to thank Canwen Xu and Julian McAuley for their guidance throughout the project and access to GPU resources.

Chapter 2 contains unpublished material coauthored with Canwen Xu and Julian McAuley. The thesis author was the primary author of this chapter.

Chapter 3 contains unpublished material coauthored with Canwen Xu and Julian McAuley. The thesis author was the primary author of this chapter.

ABSTRACT OF THE THESIS

Spoiler Recognition as Semantic Text Matching

by

Ryan Tran

Master of Science in Computer Science

University of California San Diego, 2023

Professor Julian McAuley, Chair

Engaging with a TV show in the age of the Internet often means avoiding show-related content for months out of fear of being spoiled. While spoiler detection research shows promising results for protecting viewers from generic spoilers, these approaches don't actually solve the problem of users avoiding show-related content during their watch. This is because what constitutes a spoiler is different depending on where a viewer is in the show, and spoiler detection on its own is too coarse to capture this complexity. Instead, we propose the task of spoiler recognition, which seeks to assign an episode number to a spoiler, given a show. We pose

this task as semantic text matching and present a dataset of comments and episode summaries for evaluating model performance. The dataset consists of ~3.1K and ~2.8K manually-labeled test and validation comments respectively, and over 200K auto-labeled comments for training. We experimentally demonstrate the utility of this training set and use it to benchmark the performance of BigBird, Nyströmformer, and Longformer on this task. Specifically, we cross-encode summaries with comments and examine the mean reciprocal rank scores. Our results find Longformer to be best suited for this task. We also perform an error analysis to shed some light on the kinds of challenges spoiler recognition poses. In total, we present this dataset and these results to facilitate future research into spoiler recognition.

INTRODUCTION

With the growth of Internet forums and online social platforms, further accelerated by the events of the COVID-19 pandemic, never has it been easier to engage with media as part of a virtual community. These communities can be found everywhere, from large community servers on Discord to the comment section under videos on popular streaming sites. Many of these communities are centered around a particular TV show. Because they contain passionate, like-minded individuals, these communities should in theory be the best places to engage should one want to say, discuss the events of an episode. However, these discussions are often rife with spoilers. Unfortunately, this risk often leads viewers to avoid these communities altogether until they have completely caught up with the show.

Some websites such as Reddit have built-in functionality allowing users to tag their content as spoiler. However, for long shows with a large number of episodes, this proves unsatisfactory: A user could be halfway through a show but will still be afraid to click on spoiler-tagged content for fear that it might contain spoilers for events later in the show when in reality, it might pertain to events that the viewer has already seen and with which can engage. Again, some websites allow users to tag spoilers with more granularity, but it is far from guaranteed that users will be both accurate and consistent in tagging their content. This highlights the need for automatic spoiler *recognition*. Unlike spoiler detection, spoiler recognition aims to match a given spoiler to an episode number. A spoiler recognition model working hand-in-hand with a spoiler detection model could provide much more fine-grained protection from spoilers, enabling users to engage with show-related discussions at their own speed while they are still watching the show.

In this work, we consider the setting where the show is known and we would like to determine the episode to which a comment is referring. Specifically, we pose the problem as a semantic text matching task between comments and episode summaries. The rest of this paper details the process by which we scrape the dataset, annotate it, and benchmark the performance of several language models as cross-encoders on this task. We make both our dataset[1] and source code[2] publicly available. To our knowledge, this is the first work to study the task of spoiler recognition.

[1]https://www.kaggle.com/datasets/bobotran/tv-show-spoiler-recognition
[2]https://github.com/bobotran/spoiler-recognition

CHAPTER 1 BACKGROUND

Unlike spoiler recognition, spoiler detection has been addressed by several works in the literature. Boyd-Graber et al. (2013) collected a dataset from the website TV Tropes where users tag spoiler content about TV shows. They experiment with fitting binary SVM classifiers to traditional lexical features as well as hand-designed features such as content length and genre. Incorporating genre information is a bit of a theme. Chang et al. (2018) designed a deep spoiler detection model using a CNN-based genre encoder, a BiGRU-based (Cho et al., 2014) sentence encoder, and a genre-informed attention mechanism. Wroblewska et al. (2021) experimented with concatenating genre information with the last BERT (Devlin et al., 2019) encoder state for the "[CLS]" token as well as simply appending the genre information to the input before feeding to BERT. Wroblewska et al. (2021) also investigated single-sentence classification versus sequential-sentence classification (Cohan et al., 2019), finding the additional context from classifying sentences in batch to be helpful. Similar to Boyd-Graber et al. (2013), Wan et al. (2019) collected a user-annotated spoiler dataset from the book review website Goodreads. They built a hierarchical attention network (Yang et al., 2016) for spoiler detection, incorporating additional hand-crafted features. The IMDb spoiler dataset (Misra 2022) is another publicly-available user-annotated spoiler dataset.

However, none of these datasets contain information matching a spoiler to an episode. Furthermore, the extra context incorporated by these models, such as genre, are insufficient for the much finer task of spoiler recognition. Instead, we feed an entire episode summary alongside the spoiler as a semantic text matching task.

Our setup shares strong similarities with IR systems. IR systems are typically characterized as retrieving relevant documents from a database given a query. Large-scale IR

systems commonly consist of a candidate generation step and a reranking step. The candidate generation step quickly procures a short list of documents likely to be relevant, often using a sparse retrieval method such as BM25 (Robertson et al., 2009). The re-ranking step then refines the ordering, often using a neural re-ranker such as BERT (Dai et al., 2019). For re-ranking, cross-encoders, which feed the query and document into the model together, are often used over bi-encoders, which calculate a similarity score between document and query embeddings computed independently, because cross-encoders are able to model interactions between the document and query. Some recent work has also suggested that cross-encoders generalize better to out-of-domain data (Zhan et al. 2022; Rosa et al. 2022).

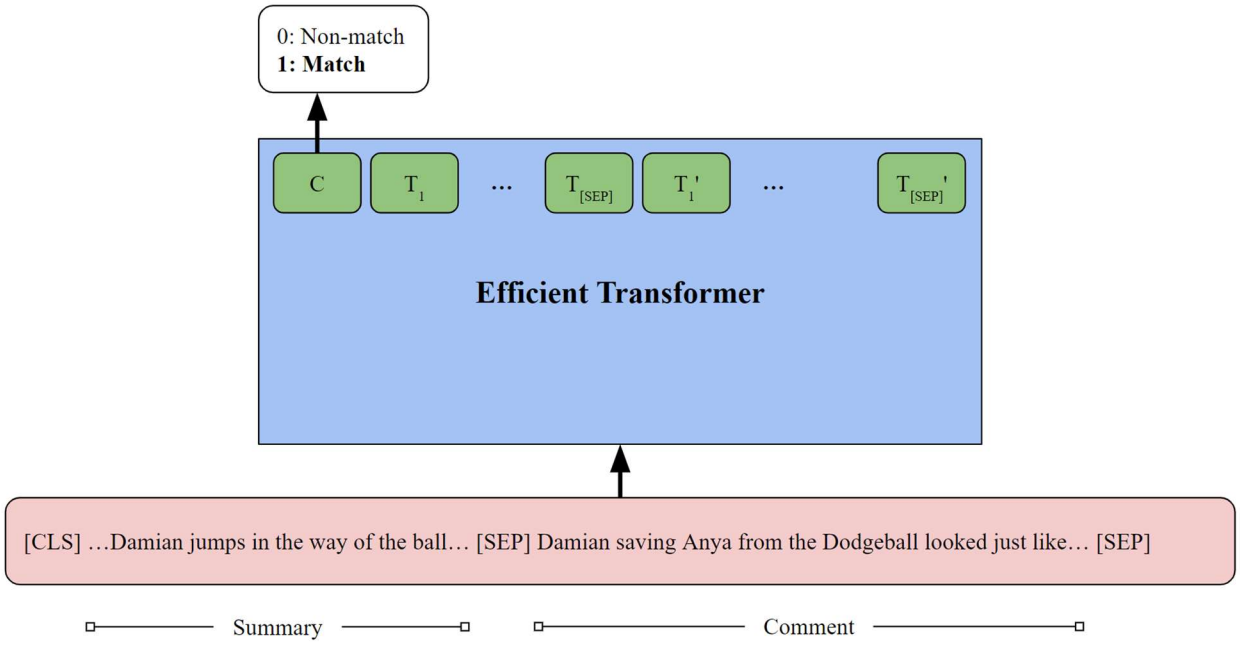

Figure 1: Our cross-encoding scheme. A summary is concatenated with a comment then passed to a Transformer architecture. The final hidden state corresponding to the [CLS] token is passed to a linear layer to predict *Match* or *Non-match*. During test time, each comment is paired with every summary that belongs to the same show.

In this work, we treat spoiler recognition as ranking episode summaries given a comment, and we evaluate the performance of different transformer architectures under the

cross-encoding scheme. Specifically, we concatenate a summary with a comment and train the model to predict *match* or *non-match* for each pair. Figure 1 illustrates this architecture. During test time, each comment is paired with every summary that belongs to the same show. We evaluate three efficient Transformer architectures, and one baseline architecture with this process.

Since its introduction, the Transformer architecture has found its way into state-of-the-art models for a wide array of different tasks. Due to the quadratic cost of its self-attention mechanism, a large number of variations on this architecture have been developed (Ziyaden 2021), and benchmarks evaluating performance on long sequences have been established (Tay et al., 2020). In particular, Tay et al. (2020) find that no one efficient Transformer design dominates but rather, the best Transformer architecture in each circumstance is task-specific. Because episode summaries are long, in this work, we focus on evaluating three efficient Transformer architectures on the spoiler recognition task: BigBird (Zaheer et al., 2021), Longformer (Beltagy et al., 2020), and Nyströmformer (Xiong et al., 2021). We choose these models because their pretrained checkpoints are publicly available so as to enable us to perform fine-tuning.

BigBird is a sparse-attention Transformer where each token attends to a fixed number of tokens on either side of it, to a fixed number of randomly selected tokens, and to a fixed set of global tokens (Zaheer et al., 2021). The authors also prove theoretically that BigBird is a universal approximator of sequence-to-sequence functions and is Turing complete. Longformer also uses sliding window attention, but eschews random attention and fixed global attention in favor of user-specified global attention. Furthermore, their global attention uses different key, query, and value matrices than their local attention. Taken together, this allows task-specific inductive bias to be added to the model (Beltagy et al., 2020). For our task, we designate all

tokens belonging to the comment as global tokens. Nyströmformer, on the other hand, uses the Nyström matrix approximation method to approximate the full self-attention matrix (Xiong et al., 2021).

As baselines, we also include evaluations for RoBERTa and BM25. RoBERTa is a checkpoint of BERT, carefully optimized with different hyperparameters and pretraining objectives (Liu et al., 2019). Crucially, however, it retains the same 512-token maximum sequence length limit, so we truncate the summaries to fit the input size. BM25 is a lexical matching approach to ranking documents by their relevance to a query (Robertson et al., 2009). It is a bag-of-words approach that does not require training but acts as a strong baseline regardless (Thakur et al., 2021).

CHAPTER 2 COLLECTING THE DATASET

Our dataset consists of 223,773 comments and 496 summaries across 13 TV shows. Each comment is labeled with a show name and episode number, and each summary is indexed with a show name and episode number. The summaries were scraped from their respective episode pages on the website fandom.com. The length of comments is a long-tailed distribution, with the median length at 19 tokens, as measured by BigBird's tokenizer, and the 98th percentile at 172 tokens. As such, comments are truncated at 172 tokens, and summaries over 3920 tokens are manually trimmed by throwing out less important details. This ensures that any concatenation of comment and summary will fit into the long-context Transformers we study.

Table 1: Dataset statistics. The length is measured after conversion to tokens via BigBird's tokenizer.

	Number	Median Length (in BigBird Tokens)
Summaries	496	1537.5
Comments	223,773	19

To prepare the comments, we first scrape 522,991 of them from discussion threads on Reddit. Threads are a hierarchical discussion format where users reply to each other's posts, and each reply increases the indentation level. Top-level comments are comments that are not made in reply to any other comment. Threads focused around discussion of a particular episode are hand-picked, and the top-level comments are scraped. We take only top-level comments to minimize collecting comments that are incomplete thoughts continued from another part of a conversation. We then remove links and markup elements and apply other cleaning preprocessing before proceeding.

Table 2: Example comments alongside an example summary. The first two comments are *relevant*; The first one corresponds to the same episode as the summary while the second does not, so they are examples of a positive and negative example during recognition training respectively. The third and fourth comments are irrelevant and were filtered out of the dataset during the auto-labeling step.

Summary	Comments
"…After thinking back to Yor's training, Anya uses her "killer move" and throws the ball at Bill. However, the ball hits the ground and bounces toward Bill, who throws the ball right back and hits her. Bill and his team were excited, thinking he was going to get a Stella Star. However, Henry informs them that they do not give out Stella Stars for a simple P.E. game..."	**Relevant - Same episode**: "Anya: 'Finisher strike: Star Catch Arrow!!' Ball: 'nah, i don't really feel like it' "
	Relevant - Different Episode: "The dog finally has a name. Borf!"
	Irrelevant: "This episode was fun. Just joy from start to end."
	Irrelevant: "Haven't been this hyped over a dodgeball game since Hunter x Hunter."

Comments are grouped by show name and episode number based on the discussion thread from which they were scraped, but we do not yet consider them labeled at this step. This is because we found that about half of the comments scraped this way are *irrelevant*. We take this time to note the subtle difference in labels at this step compared to spoiler detection. While spoiler detection classifies text as spoiler or non-spoiler, we at this step look to separate the irrelevant comments we have scraped from the relevant ones. We define a *relevant* comment as one that describes events relevant to the episode (discussion thread) from which it was scraped. Examples of irrelevant comments are discussions about music, acting quality, personal feelings about the episode, etc. Table 2 shows some examples. While the first irrelevant comment is straightforward, the second is a more nuanced example: The episode in question *is* about

dodgeball, but the comment does not discuss any events that occurred in the episode, so it is not considered relevant.

Because labeling all 522,991 comments as relevant/irrelevant would be extremely time-consuming, we took a semi-supervised approach: We labeled 11,032 comments and split them 70-20-10 into a small training, validation, and test set to train an auto-labeler. To maximize data efficiency with this small dataset, we performed prompt-based fine-tuning to train RoBERTa. Specifically, we followed the methods outlined by Gao et al. (2021). At a high level, Gao et al. (2021) described a set of techniques for few-shot fine-tuning of pre-trained language models, including prompt-based fine-tuning with demonstrations and automatic template generation. Please refer to their paper for details. It is important to note that at this step, summaries are not concatenated with comments before being fed to the auto-labeler; the auto-labeler predicts relevant/irrelevant based on the words in the comment alone. This is because the job of the auto-labeler is to filter out generic irrelevant comments. Unlike recognition, episode-specific context is not required to perform this task.

Table 3: Test set performance of the relevant/irrelevant auto-labeler. Relevant is labeled as 0 and irrelevant as 1. For each model, the threshold that resulted in the highest F1 score on the validation set was chosen and used to compute the test set F1, recall, and precision. The AUC is computed using the ROC curve.

Name	F1	Recall	Precision	AUC
Standard Fine-tuning	0.7975	0.7845	0.8109	0.9023
Prompt-based Fine-tuning	0.8109	0.8455	0.7790	0.9092
Prompt-based Fine-tuning + Demonstrations	0.8079	0.7825	0.8351	0.9097

Table 3 shows the auto-labeler's performance on the test set. For each model, the threshold that resulted in the highest F1 score on the validation set was selected as its operating

point and used to compute the test set scores. Although prompt-based fine-tuning with demonstrations achieved the highest AUC, prompt-based fine-tuning without demonstrations showed a higher F1 score at its operating point. Furthermore, prompt-based fine-tuning without demonstrations achieved a significantly higher recall. In this task, relevant is labeled as 0 and irrelevant as 1, therefore, the recall in this context estimates the probability that an irrelevant comment is caught. This is more important than precision for our application because we seek to produce a dataset of relevant comments for training that is as clean as possible, so irrelevant comments need to be reliably detected. Concurrently, because unlabeled comment data is plentiful, we suffer no real consequences from a lower-precision auto-labeler throwing out some relevant comments. Thus, we select the model trained using prompt-based fine-tuning without demonstrations as our auto-labeler.

These results somewhat differ from the findings of Gao et al. (2021), who found prompt-based fine-tuning with demonstrations and automatic template generation to be the most successful. In fact, we also found that handcrafted templates resulted in better accuracy than generated templates for our application. We theorize that this could be due to the fact that we trained our auto-labeler with significantly more data than Gao et al. (2021). Our setup is more a low-shot setting than a few-shot one, and importantly, Gao et al. (2021) found that the gains their methods offered diminished significantly as the amount of training data made available to the model increased, thus potentially explaining our results. Regardless, we found that prompt-based fine-tuning still provided a significant advantage over standard fine-tuning, so we selected the corresponding model as our auto-labeler.

Using our chosen auto-labeler, the 511,959 unlabeled comments were auto-labeled, separating 217,881 relevant comments from 294,078 irrelevant ones. Using our test set, we

estimate that of the 217,881 comments labeled as relevant, about 14% are actually irrelevant. As we will demonstrate later, this number is low enough that the auto-labeled comments still serve as an effective training set for fine-tuning a spoiler recognition model.

To recount, we have 511,959 auto-labeled comments and 11,032 hand-labeled comments. Among the auto-labeled comments, we have 217,881 relevant comments and among the hand-labeled we have 5,892. Relevant comments are converted to the spoiler recognition dataset format by assigning them the episode number of the discussion thread from which they were scraped. This results in a spoiler recognition dataset with 217,881 comments for training and 5,892 comments for validation and testing. To test the ability of the recognition models to generalize to unseen shows, the test set is constructed such that it contains 3,105 hand-labeled comments from 4 shows that are neither in the validation set nor the training set. The remaining 2,787 hand-labeled comments are used for validation.

Acknowledgements

Chapter 2 contains unpublished material coauthored with Canwen Xu and Julian McAuley. The thesis author was the primary author of this chapter.

CHAPTER 3 EXPERIMENTS

As we are treating spoiler recognition as a re-ranking task, we choose the mean reciprocal rank (Craswell 2009) as our metric of performance. Specifically, for each comment, its match score with every summary from the same show is computed. The reciprocal rank is the position of the correct summary in the descending sorted list of match scores. For this task, it is an appropriate metric since our labeling process assumes that a comment can be relevant to at most one episode. We discuss this assumption in more detail in the Error Analysis section.

Table 4: Experiment details. The dataset sizes are measured in number of comments. All models in Setup 2 were trained with the same learning rate and weight decay as Setup 1 except Nyströmformer. The validation interval specifies the frequency with which models were evaluated on the validation set during training.

	Setup 1	Setup 2
Training set size	2,229	217,881
Validation set size	558	2,787
Test set size	3,105	3,105
Learning rate	2e-5	2e-5 (Nyströmformer: 1e-5)
Weight decay	1e-2	1e-2 (Nyströmformer: 1e-3)
Validation interval	1 training epoch	576 training steps
Learning rate schedule	Reduce on plateau: Divide by 4 after 4 plateau validation epochs	
Optimizer	AdamW	
Batch size	32	

In Setup 1, we evaluate the performance of the models on hand-labeled data only. To achieve this, the original validation set (2787 comments) is divided 80-20 into a new training set

and validation set respectively, and the models are fine-tuned on these sets then evaluated on the original test set.

All models were trained using the AdamW optimizer (Loshchilov et al., 2017) with learning rate 2e-5, weight decay 1e-2, and batch size 32, with evaluation on the validation set occurring at the end of each epoch in Setup 1. Training presented an equal ratio of positive and negative examples to the models. Negatives were sampled randomly from the set of all summaries with the same show as the comment; we did briefly experiment with mining hard negatives using BM25 but found that this did not improve performance. We theorize that sampling negatives from summaries of the same show provides negatives that are already hard enough. Furthermore, should a mislabeled comment actually be relevant to multiple episodes, it's likely that the hardest negatives returned by BM25 for a comment should actually be positives, and this would deteriorate the quality of the training. Therefore, for the experiments reported in Table 5, we stick to sampling negatives randomly from within the same show. The learning rate was divided by 4 if 4 validation epochs went by without the validation MRR increasing. BM25 was computed with word stemming, stopword removal, and lowercasing. For all learned models, we use the implementations and pretrained weights made available by Huggingface Transformers (Wolf et al., 2019). All models were fine-tuned on a single NVIDIA Geforce RTX 3090, with FP16 automatic mixed precision to save compute. Table 4 summarizes these experiment details.

In Setup 2, we evaluate the performance of the models fitted to auto-labeled training data and hand-labeled validation data. With the exception of Nyströmformer, which used a learning rate of 1e-5 and weight decay 1e-3, all other hyperparameters are the same as in Setup 1. In

Setup 2, validation was performed every 576 training steps. Table 5 shows the results on the test set.

Table 5: Test set MRR on spoiler recognition dataset. In parentheses is the validation set MRR.

	Setup 1	Setup 2
BM25	0.3680 (0.3890)	0.3680 (0.4067)
RoBERTa	0.3103 (0.5463)	0.3499 (0.4514)
BigBird	0.3328 (0.6275)	0.4035 (0.5682)
Nyströmformer	0.3383 (0.5674)	0.4635 (0.5286)
Longformer	0.4271 (0.6557)	**0.5820** (0.6365)

Looking at Setup 1 in Table 5, we can see that BM25 provides a strong baseline that almost no learned model can beat when trained on the manually-labeled data alone. Contrast this with Setup 2 where all but one learned model, RoBERTa, outperforms BM25. This is to be expected since the median summary length is about 1500 tokens, but RoBERTa can see at most 512 tokens, so it is missing a great deal of crucial context that the other models have. In fact, all learned models perform better when trained on the auto-labeled set, with Longformer being the most capable of taking advantage of the extra training data. Another noteworthy result is the difference in generalization gap between Setup 1 and Setup 2: The drop from the validation MRR to the test MRR is much steeper in Setup 1. Granted, the validation scores are not directly comparable across the two setups because they do not use the same validation set, but the gap difference still suggests that training on the auto-labeled data results in better generalization. Taken together, these results make a strong case for the utility of the auto-labeled data, despite its noisy labels.

Comparing the efficient Transformers, we can see that on both setups, Longformer outperforms Nyströmformer which outperforms BigBird. Interestingly, BigBird achieves a higher validation score than Nyströmformer in both cases but a lower test score, implying that BigBird's attention model does not generalize as well on this dataset. Most striking, however, is the large gap between Longformer and the rest of the models. We theorize that the consistent global attention to all of the comment tokens (but not the summary tokens) allows the model to effectively sift through the summary to find the pertinent sections, making Longformer uniquely suited to this spoiler recognition task.

Acknowledgements

Chapter 3 contains unpublished material coauthored with Canwen Xu and Julian McAuley. The thesis author was the primary author of this chapter.

CHAPTER 4 ERROR ANALYSIS

This chapter analyzes comments discussing various episodes from two shows in the validation set, Dr. Stone and Spy x Family. The author would like to warn the reader to proceed with caution if they have not finished these shows but intend to watch them.

Table 6 lists several challenging comments along with Longformer's output on them. The first example refers to a scene where Santa very briefly flies across the sky in the background. It is treated as unimportant, fantastical garnish on an event from the episode: None of the characters acknowledge it, so it is understood to have not actually occurred. Thus, the episode 21 summary does not mention it at all. This represents a class of comments that references relevant but obscure events, which only a recent viewer of the episode might remember. Interestingly, the score is relatively low, suggesting that the model understands that it is outputting a low-quality prediction.

The second comment describes an event from the show but references a character from an entirely different show; it is drawing a comparison between two characters, one of them in the show, based on physical likeness. The reference is fairly well-known within the community, but without additional outside information, it would be difficult for the model to understand this comment beyond just context clues.

The third comment is challenging because it concerns predictions. Comments on ongoing shows often contain predictions and, if they happen to be right, will likely match better lexically to the future episode when the events occur than the current episode when they are foreshadowed/predicted. This is not necessarily a bad thing for the end user, but it poses a challenge for training and evaluating our models. For this example, it is visually hinted in episode 11 that the dog has the ability to see the future but not confirmed until episode 13, so the

summary for episode 11 does not mention this ability explicitly but the summary for episode 13 does, posing a possible explanation for the model's behavior.

The last example exposes some of the limitations with our labeling scheme. While the assessment "Loid is adjusting to be a father" is very relevant to the events of episode 7, it is not unique to episode 7; it represents a long-running arc for Loid. Therefore, during the labeling process, marking this comment as "Relevant" was correct but that label is less meaningful in the context of semantic text matching. Defining a precise way to distinguish "Relevant" from "Relevant and Unique" is a potential topic for future experiments and works.

Table 6: Longformer output on several validation set examples. For the given comment, the first column represents the rank of the correct episode, the second is the correct episode number, the third is the episode with the highest score, the fourth is this highest score, and the fifth is the text of the comment. The score is the positive-class confidence after softmax.

Rank	Correct	Prediction	Score	Comment
23	21	1	0.1031	"Great santa still alive"
1	2	1	0.5822	"Kars the ultimate lifeform is released from his stone inprisonment !!!!"
1	11	13	0.8801	"I really really want Anya to have a dog she can communicate with. If she has a pupner with the ability to predict the future....."
1	7	12	0.8128	"I like that Loid is adjusting to be a father. It makes it feel more realistic."

Taken together, these examples give a glimpse into the challenges posed by spoiler recognition. The hope is that this analysis motivates new lines of work into the study.

CONCLUSION

In this work, we defined the task of spoiler recognition and formulated it as a semantic text matching problem. We scraped a dataset of summaries and comments, and we annotated a small portion for testing and validation. We used the manually-labeled data to train an effective auto-labeler that allowed us to procure a medium-sized training set. We demonstrated the utility of this dataset and benchmarked the performance of three efficient Transformer architectures as cross-encoders, finding that Longformer is the best-suited for this task. Finally, we performed an error analysis to identify the challenges posed by this task. We present this dataset and these results in the hopes that they will drive further research in the study of spoiler recognition.

REFERENCES

Beltagy, I., Peters, M. E., & Cohan, A. (2020). Longformer: The long-document transformer. *arXiv:2004.05150v2.* https://doi.org/10.48550/arXiv.2004.05150

Cho, K., van Merrienboer, B., Bahdanau, D., & Bengio, Y. (2014). On the properties of neural machine translation: Encoder-decoder approaches. *arXiv:1409.1259v2*. https://doi.org/10.48550/arXiv.1409.1259

Cohan, A., Beltagy, I., King, D., Dalvi, B., & Weld, D. (2019). Pretrained language models for sequential sentence classification. *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 3693–3699. http://dx.doi.org/10.18653/v1/D19-1383

Craswell, N. (2009). Mean reciprocal rank. *Encyclopedia of Database Systems*, 1703–1703. https://doi.org/10.1007/978-0-387-39940-9_488

Dai, Z., & Callan, J. (2019). Deeper text understanding for IR with contextual neural language modeling. *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 985-988. https://doi.org/10.1145/3331184.3331303

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2018). BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv:1810.04805v2*. https://doi.org/10.48550/arXiv.1810.04805

Gao, T., Fisch, A., & Chen, D. (2021). Making pre-trained language models better few-shot learners. *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, 3816–3830. http://dx.doi.org/10.18653/v1/2021.acl-long.295

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy., O., Lewis, M., Zettlemoyer, L., & Stoyanov, V. (2019). RoBERTa: A robustly optimized BERT pretraining approach. *arXiv:1907.11692v1*. https://doi.org/10.48550/arXiv.1907.11692

Loshchilov, I., & Hutter, F. (2017). Decoupled weight decay regularization. *arXiv:1711.05101v3*. https://doi.org/10.48550/arXiv.1711.05101

Misra, R. (2022). IMDB spoiler dataset. *arXiv:2212.06034v1*. https://doi.org/10.48550/arXiv.2212.06034

Robertson, S., & Zaragoza, H. (2009). The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends in Information Retrieval, 3*(4), 333–389. https://doi.org/10.1561/1500000019

Rosa, G., Bonifacio, L., Jeronymo, V., Abonizio, H., Fadaee, M., Lotufo, R., & Nogueira, R. (2022). In defense of cross-encoders for zero-shot retrieval. *arXiv:2212.06121v1*. https://doi.org/10.48550/arXiv.2212.06121

Thakur, N., Reimers, N., Rücklé, A., Srivastava, A., & Gurevych, I. (2021). BEIR: A heterogenous benchmark for zero-shot evaluation of information retrieval models. *arXiv:2104.08663v4*. https://doi.org/10.48550/arXiv.2104.08663

Tay, Y., Dehghani, M., Abnar, S., Shen, Y., Bahri, D., Pham, P., Rao, J., Yang, L., Ruder, S., & Metzler, D. (2020). Long range arena: A benchmark for efficient transformers. *arXiv:2011.04006v1*. https://doi.org/10.48550/arXiv.2011.04006

Wan, M., Misra, R., Nakashole, N., & McAuley, J. (2019). Fine-grained spoiler detection from large-scale review corpora. *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2605–2610. http://dx.doi.org/10.18653/v1/P19-1248

Wroblewska, A., Rzepinski, P., & Sysko-Romanczuk, S. (2021). Spoiler in a textstack: How much can transformers help? *arXiv:2112.12913v1.* https://doi.org/10.48550/arXiv.2112.12913

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., von Platen, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Scao, T. L., Gugger, S., Drame, M., Lhoest, Q., & Rush, A. M. (2019). HuggingFace's transformers: State-of-the-art natural language processing. *arXiv:1910.03771v5*. https://doi.org/10.48550/arXiv.1910.03771

Xiong, Y., Zeng, Z., Chakraborty, R., Tan, M., Fung, G., Li, Y., & Singh, V. (2021). Nyströmformer: A nyström-based algorithm for approximating self-attention. *arXiv:2102.03902v3*. https://doi.org/10.48550/arXiv.2102.03902

Yang, Z., Yang, D., Dyer, C., He, X., Smola, A., & Hovy, E. (2016). Hierarchical attention networks for document classification. *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 1480–1489. http://dx.doi.org/10.18653/v1/N16-1174

Zaheer, M., Guruganesh, G., Dubey, A., Ainslie, J., Alberti, C., Ontanon, S., Pham, P., Ravula, A., Wang, Q., Yang, L., & Ahmed, A. (2020). Big bird: Transformers for longer sequences. *arXiv:2007.14062v2*. https://doi.org/10.48550/arXiv.2007.14062

Zhan, J., Xie, X., Mao, J., Liu, Y., Guo, J., Zhang, M., & Ma, S. (2022). Evaluating interpolation and extrapolation performance of neural retrieval models. *arXiv:2204.11447v2*. https://doi.org/10.48550/arXiv.2204.11447

Ziyaden, A., Yelenov, A., & Pak, A. (2021). Long-context transformers: A survey. *2021 5th Scientific School Dynamics of Complex Networks and Their Applications (DCNA)*, 215–218. https://doi.org/10.1109/DCNA53427.2021.9587279