# Using LLMs for Spoiler Detection

Anonymous ACL submission

## Abstract

Detecting spoilers in online book recommendation forums is essential for preserving users' reading experience. Traditional spoiler detection methods often rely on handcrafted features and domain-specific knowledge, limiting their scalability and adaptability. This study focuses on investigating Large Language Models (LLMs) for spoiler detection without relying on manual feature engineering or training neural networks from scratch. We frame the task as a binary classification problem and use a balanced subset of the GoodReads dataset. We experiment with zero-shot prompting, few-shot prompting, and fine-tuning approaches with the Llama3.1-Instruct model and compare the performance with existing non-neural methods including our History Model, Bag-Of-Words and TFIDF. Our findings demonstrate that LLMs are both more effective at learning from training data to perform spoiler detection as well as capable of detecting spoilers with no additional pre-training.

## 1 Introduction

The rise of online platforms for book discussions has led to an abundance of book reviews from people with various backgrounds. However, some reviews often contain spoilers, which diminishes the enjoyment of reading a book. Detecting spoilers using automatic approaches has been widely studied in Natural Language Processing (NLP) research. Early spoiler detection approaches heavily depend on handcrafted knowledge and manually engineered features, such as bag-of-words models augmented with linguistic cues extracted through dependency parsing (Guo and Ramakrishnan, 2010) and incorporating metadata and user behavior (Boyd-Graber et al., 2013). These techniques often require extensive feature engineering and domain-specific knowledge, limiting their generalizability and scalability. Machine learning methods such as fine-tuning transformer-based pre-trained models and training graphical networks (Bao et al., 2021; Chang et al., 2021; Wang et al., 2023) still require labeled datasets and task-specific adjustments.

With the emergence of Large Language Models (LLMs), there is potential to capture the complex linguistic patterns and context with LLMs that are trained on a vast amount of web data (Kojima et al., 2023). In this project, we focus on studying and analyzing a balanced subset of the GoodReads dataset (Wan et al., 2019) for spoiler detection. Framing spoiler detection as a binary classification problem, we examine the potential of pre-trained LLMs to detect spoilers without the need of manual feature selection using zero-shot prompting, few-shot prompting, and fine-tuning methods. We compare the results with existing state-of-the-art spoiler detection methods including Bag-Of-Words and TFIDF.

## 2 Dataset

We use a portion of the spoiler reviews in the GoodReads review dataset (Wan et al., 2019). The original dataset was sub-sampled to create a split containing a roughly equal number of reviews with and without spoilers. This was done due to the original dataset being heavily biased towards spoiler-free reviews, with spoiler-containing reviews consisting of only 89,627 samples out of 1,378,033 or roughly 6.5% of the total dataset, allowing for a naive model to obtain a high accuracy by naively guessing that the review is spoiler-free. Our sub-sampled dataset consists of 200,000 total reviews with the original 89,627 reviews with spoilers and 110,373 reviews without spoilers. We show metrics demonstrating that our sub-sampled dataset is representative of the dataset as a whole in Table 1.

We divide the subsampled data into training, validation and test splits in a 80:10:10 ratio. All reported results are on the test split. The original subsampled data was first divided into spoiler and non-spoiler reviewers before being split and rejoined such that the roughly 50-50 split of labels were preserved in all splits.

## 3 Predictive Task

Our task is classifying whether a review contains spoilers. The definition of what constitutes a

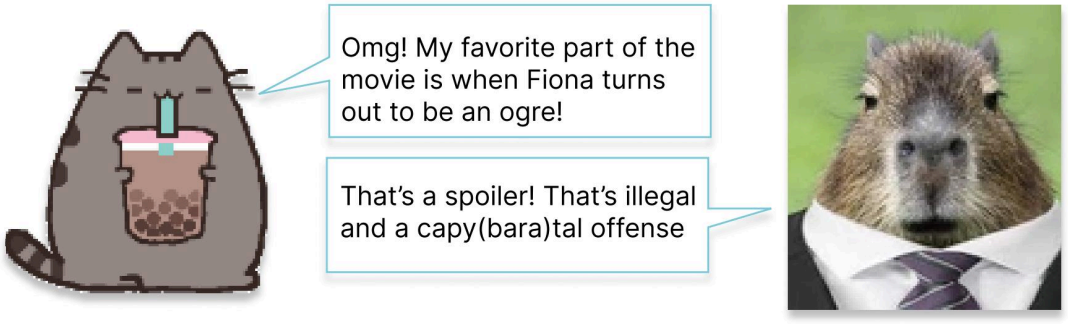


Figure 1: Motivation figure illustrating the necessity of spoiler detection in real world situations.

| Label | Avg Num of Words | Std Num of Words | Avg Num of Uniq Words | Avg Std of Uniq Words | Uniq Users | Uniq Books |
|---|---|---|---|---|---|---|
| **Full and Subsampled Dataset (Spoiler)** | | | | | | |
| Spoiler | **367.41** | **298.76** | **213.07** | **136.84** | **18892** | **25475** |
| **Full Dataset (No Spoilers)** | | | | | | |
| No Spoilers | **182.27** | **215.95** | **114.74** | **111.33** | **18612** | **25475** |
| **Subsampled Dataset (No Spoilers)** | | | | | | |
| No Spoilers | **173.32** | **202.84** | **110.02** | **104.97** | **14965** | **21231** |

Table 1: Comparison of Full and Subsampled Datasets. All reviews containing spoilers were used while only a portion of the reviews without spoilers were used. We see in terms of average review length and number of unique words, there is no statistically significant difference between the full dataset and the subsampled dataset. Despite only using 8% of the spoiler-free reviews, we still retain the majority of unique users and books.

spoiler can be highly subjective depending on the audience, ranging from explicit events, ie "One of the main characters dies in Act Three" to vague hints such as "the story has a tragic ending". However, defining exactly to what extent a comment must reveal something about a story to be considered a 'spoiler' is outside the scope of our work. The GoodReads Review dataset relies on the author themselves to tag whether or not a review contains a spoiler and we use this as our ground-truth.

In a recommendation setting, one framing of this task is as a binary recommendation task where given a user's past history of reviews, if their newest review also contains spoilers. We use this recommendation setting model along with a Bag-Of-Words approach as non-neural baselines against our LLM-based approaches. We use zero-shot, few-shot and fine-tuning approaches for the Llama3.1-Instruct family of models. Due to monetary, computation and time constraints, we use only Llama3.1-70B-Instruct for the zero and few-shot experiments and Llama-8B-Instruct for fine-tuning.

## 4 Model

We evaluate three non-neural recommendation-based models and three approaches involving LLMs. We detail each method below:

### 4.1 History Model

The History model uses the percentage of reviews the user gave that were or were not spoilers, to determine whether the current review contained a spoiler. This model was primary motivated by the theory that people will have a distinct reviewing style and so people who write more reviews containing spoilers than reviews without spoilers would likely prefer the former to the latter. While straightforward, we found that introducing additional complexity only decreased the performance of this model. Attempts included combinations of:

- Classifying based on the higher of the max/mean of the Jaccard similarities between all spoiler and non-spoiler reviews
- Weighting by the percentage of the user's spoiler and non-spoiler reviews.
- Using Cosine Similarity rather than Jaccard Similarity
- Classifying based on word count of the review.

| C | Accuracy | Precision | Recall | F1 | Balance Error Rate |
|---|---|---|---|---|---|
| 0.01 | 0.73 | 0.70 | 0.70 | 0.70 | 0.27 |
| 0.1 | 0.75 | 0.72 | 0.72 | 0.72 | 0.25 |
| 1 | 0.76 | 0.73 | 0.73 | 0.73 | 0.25 |
| 10 | 0.76 | 0.73 | 0.73 | 0.73 | 0.25 |
| 100 | 0.76 | 0.73 | 0.73 | 0.73 | 0.25 |

Table 2: Hyperparameter tuning for the value of C

### 4.2 Bag-Of-Words model

The Bag-Of-Words model represents text data by counting the frequency of each word in an document. We select the BOW model as the baseline because of its simplicity and ease of implementation. We transform the reviews into feature vectors based on word counts and use a logistic regression classifier to predict spoilers. This approach achieved a better performance compared to the History Model, likely due to the feature vectors being able to better represent common words that often signify a spoiler. However, this method lacks understanding of the word order and context, leading to many misclassifications.

### 4.3 TF-IDF model

The Term Frequency-Inverse Document Frequency (TF-IDF) model improves upon BOW model by weighting words based on their frequency in a document relative to their frequency across all documents. We apply TF-IDF vectorization to the reviews and input to logistic regression classifier. TF-IDF model addresses some limitations of the BOW model by reducing the impacts of common words, but it still lacks understanding of the semantic meanings. We hyperparameter tune the $C$ regularization coefficient and the max number of features on the validation set, finding that $C$ of 1 and a max number of features of 10000 performs best.

#### 4.3.1 Hyperparameter Tuning for TFIDF

We try 5 different values of $C$: $[0.01, 0.1, 1, 10, 100]$ and find that a $C$ of 1 performed the best. Results can be found in Table 2. Similarly, we ablate the max features elements to find that the number of max features that performs best is 10000. We report our results in Table 3

### 4.4 Pre-trained LLM for prompting

Recent studies have demonstrated pre-trained LLMs' abilities in adapting to new tasks with zero-shot and few-shot prompting methods. Leveraging the power of pre-trained models, we use the Llama-3.1-70B-Instruct model to assess whether it could successfully predict spoilers based solely on its pre-training data.

We chose the Llama3.1-Instruct family of models due to their state of the art performance. Our primary bottleneck for not testing across a wider range of models was monetary costs in the case of proprietary models (GPT-4, Claude) or inference-time costs in the case of other open-source models (Mistral, Olmo).

#### 4.4.1 Zero-shot

Our zero-shot prompt was optimized over a further subsampled pool of 5 reviews with spoilers and 5 reviews without spoilers from the validation set. We do not optimize the prompt over the entire validation set due to time constraints, with inference over the entire testing set taking roughly four hours. We found that including more descriptive identifiers of what constituted a spoiler such as 'mention of a character' or 'explicitly describing events that occur in the latter half of the book' decreased performance, likely due to the subjectivity of how the dataset was originally created. As stated before, concretely defining what constitutes a spoiler is outside the scope of this work.

#### 4.4.2 Few-shot

For our few-shot experiments, we kept the initial prompt largely the same as in the zero-shot experiments but also included several examples of reviews that did contain spoilers and reviews that did not contain spoilers, sampled from the valida-

| Max Features | Accuracy | Precision | Recall | F1 | Balanced Error Rate |
|---|---|---|---|---|---|
| 100 | 0.72 | 0.68 | 0.71 | 0.70 | 0.27 |
| 1000 | 0.75 | 0.73 | 0.73 | 0.73 | 0.25 |
| 10000 | 0.77 | 0.76 | 0.74 | 0.75 | 0.23 |
| 50000 | 0.77 | 0.76 | 0.74 | 0.75 | 0.23 |
| 100000 | 0.77 | 0.76 | 0.74 | 0.75 | 0.23 |

Table 3: Hyperparameter tuning for the max number of features. We find that a number of features for 10000, 50000, 100000 performs roughly equivalent so we select the smallest number of features to avoid overfitting. We perform hyperparameter tuning on the validation of the dataset.

| | Accuracy | Precision | Recall | F1 | Balanced Error Rate |
|---|---|---|---|---|---|
| History | 0.70 | 0.62 | 0.68 | 0.65 | 0.30 |
| BOW | 0.74 | 0.77 | 0.60 | 0.68 | 0.27 |
| TFIDF | 0.78 | 0.76 | 0.73 | 0.74 | 0.23 |
| Zero-shot llama3-70b | 0.71 | 0.81 | 0.46 | 0.59 | 0.31 |
| Few-shot llama3-70b | 0.70 | 0.80 | 0.45 | 0.58 | 0.32 |
| Zero-shot llama3-8b | 0.58 | 0.58 | 0.28 | 0.38 | 0.45 |
| Finetuned llama3-8b | 0.85 | 0.79 | 0.90 | 0.84 | 0.15 |

Table 4: Prediction results on test data using different methods. BOW stands for the Bag of Words method.

tion set. We found that including a larger number of samples did not significantly increase or decrease performance when testing on the LLM-evaluation subsample of the data. The key challenge for few-shot (or in-context) learning is the LLMs' sensitivity to different prompts. In addition, LLMs may produce inconsistent results given shuffled order of the few-shot examples, making it unstable in the accuracy score. We did not notice any significant difference in performance when using different few-shot examples, however the inference time was greatly increased versus the zero-shot experiments, a single iteration over the test set taking almost ten hours.

### 4.5 Pre-trained LLM for fine-tuning

Fine-tuning LLMs involves updating a pre-trained LLMs' weights on a downstream task dataset to improve the performance on the task. To unleash LLMs' power in adapting to new tasks with limited data, we fine-tune a Llama-8B-Instruct model on a subset of the GoodReads dataset with spoiler labels. We optimize the training with learning rate scheduling and batch size to prevent overfitting. Using a fine-tuned LLM achieves better performance by learning task-specific patterns and enables scalability on demonstration data. Despite the advantages of fine-tuning, we should be aware of the computational cost of training and inference.

## 5 Results

We report the results of our models on the down-sampled test dataset. The results can be found in Table 4. Hyperparameter tuning details for the TFIDF method can be found in Section 4.3.1.

### 5.1 Efficient Fine-Tuning with QLoRA using LLM

Fine-tuning the Llama-3-8B-Instruct model using QLoRA (Dettmers et al., 2023) significantly improved its performance on the spoiler detection task. Leveraging a balanced subset of 64,000 training examples, 8,000 validation examples, and 8,000 testing examples from the GoodReads dataset, the model was optimized for computational efficiency by using only a fraction of the available data to fit the constraints of a single A100 GPU. The fine-tuning process was conducted for one epoch, employing a 4-bit quantization for reduced memory usage. After fine-tuning, the model achieved an accuracy of 85%, a substantial improvement over the pre-finetuning accuracy of 58%. Precision improved from 58% to 79%, while recall exhibited a significant increase from 28% to 90%. The balanced error rate dropped from 45% to 15%, highlighting a notable reduction in classification errors. The superior performance of the fine-tuned Llama-3-8B-Instruct model underscores the efficacy of large language models in capturing complex linguistic patterns and contextual nuances, which are often missed by traditional methods. This advancement highlights the potential of fine-tuned LLMs to significantly enhance tasks such as spoiler detection, where understanding subtle contextual cues is crucial.

### 5.2 Analysis of Results

We see that when trained on a balanced subset of the data, the recommendation-based approaches are all roughly in the same range of performance with TDIDF performing the best. With LLMs, we see that fine-tuning drastically improves the performance of Llama3-8b with performance going from below the recommendation-based approaches to significantly above the performance of TDIDF. This suggests that the QLoRA-finetuning is able to allow the pre-trained LLM to better capture the nuance of what might constitute a spoiler. Notably, all methods outside of those involving Llama3.1-70 involved a significant amount of training data for the models to achieve their performance where as Llama3.1-70b was capable of classifying spoilers when provided only the review itself. However, we acknowledge the possibility that the Goodread Reviews were part of the dataset used the train the Llama models, albeit not for the specific task in this paper.

## References

Allen Bao, Marshall Ho, and Saarthak Sangamnerkar. 2021. Spoiler alert: Using natural lan-

guage processing to detect spoilers in book reviews. *Preprint*, arXiv:2102.03882.

Jordan L. Boyd-Graber, Kimberly Glasgow, and Jackie Sauter Zajac. 2013. Spoiler alert: Machine learning approaches to detect social media posts with revelatory information. In *ASIS&T Annual Meeting*.

Buru Chang, Inggeol Lee, Hyunjae Kim, and Jaewoo Kang. 2021. "killing me" is not a spoiler: Spoiler detection model using graph neural networks with dependency relation-aware attention mechanism. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 3613–3617, Online. Association for Computational Linguistics.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. Qlora: efficient finetuning of quantized llms (2023). *arXiv preprint arXiv:2305.14314*, 52:3982–3992.

Sheng Guo and Naren Ramakrishnan. 2010. Finding the storyteller: Automatic spoiler tagging using linguistic cues. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 412–420, Beijing, China. Coling 2010 Organizing Committee.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2023. Large language models are zero-shot reasoners. *Preprint*, arXiv:2205.11916.

Mengting Wan, Rishabh Misra, Ndapa Nakashole, and Julian McAuley. 2019. Fine-grained spoiler detection from large-scale review corpora. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2605–2610, Florence, Italy. Association for Computational Linguistics.

Heng Wang, Wenqian Zhang, Yuyang Bai, Zhaoxuan Tan, Shangbin Feng, Qinghua Zheng, and Minnan Luo. 2023. Detecting spoilers in movie reviews with external movie knowledge and user networks. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 16035–16050, Singapore. Association for Computational Linguistics.