# Fine-Grained Spoiler Detection from Large-Scale Review Corpora

**Mengting Wan**[*], **Rishabh Misra**[†], **Ndapa Nakashole**[*], **Julian McAuley**[*]
[*]University of California, San Diego, [†]Amazon.com, Inc
{m5wan, r1misra, nnakashole, jmcauley}@ucsd.edu

## Abstract

This paper presents computational approaches for automatically detecting critical plot twists in reviews of media products. First, we created a large-scale book review dataset that includes fine-grained spoiler annotations at the sentence-level, as well as book and (anonymized) user information. Second, we carefully analyzed this dataset, and found that: spoiler language tends to be book-specific; spoiler distributions vary greatly across books and review authors; and spoiler sentences tend to jointly appear in the latter part of reviews. Third, inspired by these findings, we developed an end-to-end neural network architecture to detect spoiler sentences in review corpora. Quantitative and qualitative results demonstrate that the proposed method substantially outperforms existing baselines.

## 1 Introduction

'Spoilers' on review websites can be a concern for consumers who want to fully experience the excitement that arises from the pleasurable uncertainty and suspense of media consumption (Loewenstein, 1994). Certain review websites allow reviewers to tag whether their review (or sentences in their reviews) contain spoilers. However, we observe that in reality only a few users utilize this feature. Thus, requiring sentence-level spoiler annotations from users is not a successful approach to comprehensive fine-grained spoiler annotation. One possible solution is crowdsourcing: whereby consumers can report reviews that reveal critical plot details. This is complementary to the self-reporting approach, but may have scalability issues as it is relatively difficult to engage sufficient consumers in a timely fashion. Therefore, we seek to address the lack of completeness exhibited by self-reporting and crowdsourcing. We instead focus on developing machine learning techniques to automatically detect spoiler sentences from review documents.

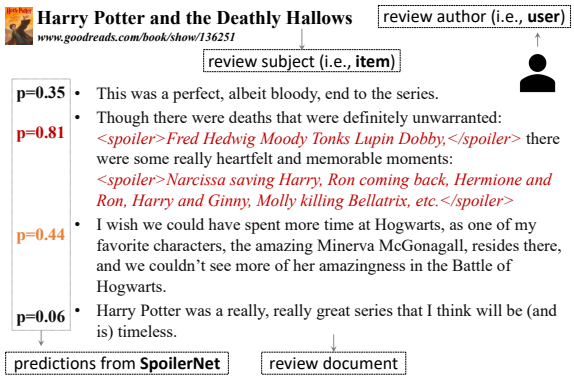
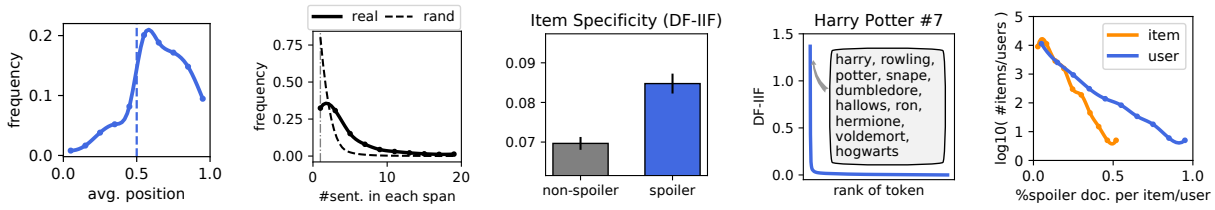



Figure 1: An example review from ***Goodreads***, where spoiler tags and the predicted spoiler probabilities from **SpoilerNet** are provided.

**Related Work.** Surprisingly, we find that spoiler analysis and detection is a relatively unexplored topic; previous work focuses on leveraging simple topic models (Guo and Ramakrishnan, 2010), or incorporating lexical features (e.g. unigrams) (Boyd-Graber et al., 2013; Iwai et al., 2014), frequent verbs and named entities (Jeon et al., 2013), and external meta-data of the review subjects (e.g. genres) (Boyd-Graber et al., 2013) in a standard classifier such as a Support Vector Machine. Deep learning methods were first applied to this task by a recent study (Chang et al., 2018), where the focus is modeling external genre information. Possibly due to the lack of data with complete review documents and the associated user (i.e., the review author) and item (i.e., the subject to review) ids, issues such as the dependency among sentences, the user/item spoiler bias, as well as the sentence semantics under different item contexts, have never been studied in this domain.

Neural network approaches have achieved great success on sentence/document classification tasks, including **CNN**-based approaches (Kim, 2014),




(a) (b) (c) (d) (e)

Figure 2: Distributions of **(a)** average spoiler sentence position; **(b)** the length of each spoiler span; **(c)** item-specificity of non-spoiler and spoiler sentences (sample means and 95% confidence intervals); **(d) DF-IIF** of each term and top ranked item-specific terms for an example book; **(e)** the percentage of spoiler reviews per book/user.

**RNN**-based approaches (Yang et al., 2016), and self-attention-based approaches (Devlin et al., 2018). In this study, we cast the spoiler sentence detection task as a special sentence classification problem, but focus on modeling domain-specific language patterns.

**Contributions.** To address real-world, large-scale application scenarios and to facilitate the possibility of adopting modern 'data-hungry' language models in this domain, we collect a new large-scale book review dataset from *goodreads.com*. Spoiler tags in this dataset are self-reported by the review authors and are sentence-specific, which makes it an ideal platform for us to build supervised models. Motivated by the results from preliminary analysis on ***Goodreads***, we propose a new model **SpoilerNet** for the spoiler sentence detection task. Using the new ***Goodreads*** dataset and an existing small-scale ***TV Tropes*** dataset (Boyd-Graber et al., 2013), we demonstrate the effectiveness of the proposed techniques.

## 2 The *Goodreads* Book Review Dataset

We scraped 1,378,033 English book reviews, across 25,475 books and 18,892 users from *goodreads.com*, where each book/user has at least one associated spoiler review. These reviews include 17,672,655 sentences, 3.22% of which are labeled as 'spoiler sentences.' To our knowledge, this is the first dataset with fine-grained spoiler annotations at this scale. This dataset is available at `https://github.com/MengtingWan/goodreads`.

**Appearance of Spoiler Sentences.** We first analyze the appearance of spoiler sentences in reviews by evaluating 1) the average position of spoiler sentences in a review document and 2) the average number of sentences in a spoiler span (a series of consecutive spoiler sentences). We present the first evaluation in Figure 2a. Compared with the expected average position of randomly sampled sentences (0.5), we observe that spoiler contents tend to appear later in a review document. For the second evaluation, we create a benchmark distribution by randomly sampling sentences within reviews and averaging the length of each span formed by those sentences. From Figure 2b, compared with this random benchmark, we notice that real-world spoiler sentences tend to be 'clumped' (i.e., more sentences in each span).

**Item-Specificity.** As book-specific terms such as locations or characters' names could be informative to reveal plot information (Jeon et al., 2013), we develop an effective method to identify the specificity of tokens regarding each item (i.e., each book) as follows:[1]

- (**Popularity**) For word $w$, item $i$, we calculate the item-wise document frequency (**DF**) as $DF_{w,i} = \frac{|\mathcal{D}_{w,i}|}{|\mathcal{D}_i|}$;
- (**Uniqueness**) For each word $w$, we calculate its inverse item frequency (**IIF**) as $IIF_w = \log \frac{|\mathcal{I}|+\epsilon}{|\mathcal{I}_w|+\epsilon}$;
- Then for each term $w$, item $i$, we are able to obtain the **DF-IIF** as $DF_{w,i} \times IIF_w$.

We show the distributions of the average **DF-IIF** values of randomly sampled non-spoiler and spoiler sentences in Figure 2c, where we find spoilers are likely to be more book-specific. The ranking of terms for the book ***Harry Potter #7*** is presented in Figure 2d, where we find that all of the top 10 terms refer to the character/author names and important plot points.

**Item/User Spoilers and Self-Reporting Bias.** We further investigate the fraction of reviews containing spoiler content per item/user to analyze the spoiler appearance tendencies for each item and

[1] $|\mathcal{D}_i|$: #reviews associated with $i$; $|\mathcal{D}_{w,i}|$: #reviews containing word $w$; $|\mathcal{I}_w|$: #items containing $w$; $|\mathcal{I}|$: the total number of items. $\epsilon = 1$ is a smoothing term.

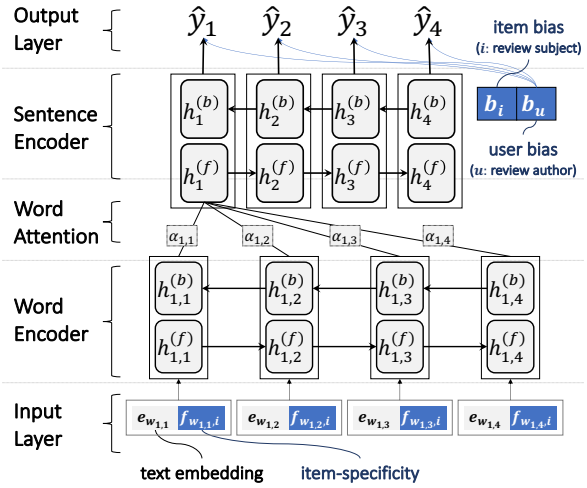



Figure 3: Model architecture of **SpoilerNet**

user (Figure 2e). We notice that the distributions are highly skewed indicating significantly different spoiler tendencies across users and items.

**Summary of Insights.** We summarize the obtained insights as follows: 1) Spoiler sentences generally tend to appear together in the latter part of a review document, which indicates the dependency among sentences and motivates us to consider encoding such information in a spoiler detection model; 2) Item-specificity could be useful to distinguish spoiler contents; 3) Distributions of self-reported spoiler labels are dramatically different across users and items, which motivates us to explicitly calibrate them in the model design.

## 3 The Proposed Approach: SpoilerNet

We formulate the predictive task as a binary classification problem: given a sentence $s$ in a review document, we aim to predict if it contains spoilers ($y_s = 1$) or not ($y_s = 0$).

We introduce **SpoilerNet**, which extends the hierarchical attention network (**HAN**) (Yang et al., 2016) by incorporating the above insights. We use the sentence encoder in **HAN** to model the sequential dependency among sentences. We incorporate the item-specificity information in the word embedding layer to enhance word representations based on different item (e.g. book) contexts. Item and user bias terms are included in the output layer to further alleviate the disparity of spoiler distributions. Figure 3 shows the overall architecture of our proposed **SpoilerNet**. We briefly describe each layer of this network as follows.

**Input Layer.** For each word $w$, we introduce a $K$-dimensional text embedding $\boldsymbol{e}_w$ to represent its lexical information, which is shared across the corpus. For each word in each sentence, we calculate its corresponding *item specificity features*: $\boldsymbol{f}_{w,i} = [DF_{w,i}, IIF_w, DF_{w,i} \times IIF_w]$. We expect this component could help distinguish different word semantics under different contexts (e.g., 'Green' indicates a character's name with high item-specificity while it represents a color otherwise). The concatenated vector $[\boldsymbol{e}_w; \boldsymbol{f}_{i,w}]$ is used as the input word embedding of word $w$ in sentence $s$.

**Word Encoder, Word Attention, and Sentence Encoder.** Next we pass words through bidirectional recurrent neural networks (**bi-RNN**) with Gated Recurrent Units (**GRU**) (Cho et al., 2014). **GRUs** accept a sequence of input embedding vectors $\boldsymbol{x}_t$ and recursively encode them into hidden states $\boldsymbol{h}_t$. Words are fed sequentially through a **GRU** and in reverse order through another **GRU**. Then we use the concatenation of these forward and backward hidden state vectors $\boldsymbol{h}_w = [\boldsymbol{h}_w^{(f)}; \boldsymbol{h}_w^{(b)}]$ to represent a word $w$ in a sentence $s$.

Then we introduce a word attention mechanism to focus on revelatory words (e.g., 'kill', 'die'), which yields

$$\boldsymbol{\mu}_w = tanh(\boldsymbol{W}_a \boldsymbol{h}_w + \boldsymbol{b}_a),$$
$$\alpha_w = \frac{\exp(\boldsymbol{\nu}^T \boldsymbol{\mu}_w)}{\sum_{w' \in s}, \exp(\boldsymbol{\nu}^T \boldsymbol{\mu}_{w'})}, \quad \boldsymbol{v}_s = \sum_{w \in s} \alpha_w \boldsymbol{h}_w,$$

where $\boldsymbol{W}_a$, $\boldsymbol{b}_a$ and $\boldsymbol{\nu}$ are model parameters. The weighted sums $\boldsymbol{v}_s$ are used as an input vector to represent sentence $s$ in the following sentence-level model.

Within each review, we pass the sentence input vectors $\{\boldsymbol{v}_s\}$ to another **bi-RNN** with **GRU** to encode the sequential dependency among sentences. We concatenate the resulting forward and backward hidden states to get the final representation of a sentence, i.e., $\boldsymbol{h}_s = [\boldsymbol{h}_s^{(f)}; \boldsymbol{h}_s^{(b)}]$.

**Output Layer.** The spoiler probability of a sentence $s$ can be calculated as

$$p_s = \sigma(\boldsymbol{w}_o^T \boldsymbol{h}_s + b_i + b_u + b).$$

Here for each item $i$ and each user $u$, we introduce learnable parameters $b_i, b_u$ to model the *item and user biases* which can not be explained by the language model. Then we consider minimizing the following training loss

$$\mathcal{L} = \sum \left( y_s \log p_s + \eta (1 - y_s) \log(1 - p_s) \right),$$

where $\eta$ is a hyper-parameter used to balance positive and negative labels in the training data.

## 4 Experiments

We consider the following two datasets:

***Goodreads*.** We use the top 20,000 frequent unigrams as our vocabulary. We randomly select 20% of the reviews for testing. Among the remaining 80%, we separate 10,000 reviews for validation and use all other reviews for training. As the distribution of spoiler labels is severely imbalanced, we decrease the weight of negative labels to $\eta = 0.05$, which yields best results among $\{0.05, 0.1, 0.2, 0.5\}$ on the validation set.

***TV Tropes*** is a small-scale benchmark dataset collected from *tvtropes.org* (Boyd-Graber et al., 2013). This dataset contains 16,261 *single-sentence* comments about 884 TV programs, which have been partitioned into 70/10/20 training/validation/test splits. All unigrams are kept in the vocabulary. As it is a balanced dataset (52.72% of the sentences are spoilers), we set $\eta = 1$.

We use the ADAM optimizer (Kingma and Ba, 2014) with a learning rate of 0.001, a fixed batch size (64) and dropout (0.5) in the fully connected output layer. The dimensionalities of all hidden states and the context attention vector $\boldsymbol{\nu}$ are set to 50. Word embeddings are initialized with pre-trained **fasttext** word vectors (Joulin et al., 2016).

**Baselines.** We consider the following baselines:

- **SVM.** Similar to previous studies (Boyd-Graber et al., 2013; Jeon et al., 2013), we apply **SVM** with a linear kernel where counts of words are used as features.
- **SVM-BOW.** Weighted averages of **fasttext** word embeddings (Joulin et al., 2016) are used as sentence features, where the weights are **Tf-Idf**s.
- **CNN.** **textCNN** (Kim, 2014) is applied where we use filter sizes 3,4, and 5, each with 50 filters.
- **HAN.** The item-specificity features and the item/user bias terms are removed from **SpoilerNet**. This can be regarded as a variant of **HAN** (Yang et al., 2016).

We add the item-specificity features and the item/user bias respectively on the above baselines to evaluate their effectiveness. We remove each of the word attention module, the pre-trained word embedding initialization, and the sentence encoder from **HAN** to evaluate their performance.

**Evaluation.** Due to the possible *subjectivity* of users' self-reported spoiler tags (i.e., different users may maintain different standards for various review subjects), we regard the area under the ROC curve (**AUC**) as our primary evaluation metric, i.e., we expect a positive spoiler sentence is ranked higher than a negative non-spoiler sentence based on $p_s$. For ***Goodreads***, we also calculate the sentence ranking **AUC** within each review document and report the average across reviews. Note this averaged document **AUC** is invariant of item/user self-reporting bias, thus the language model can be evaluated exclusively. We also report **accuracy** on ***TV Tropes*** so that our results can be fairly compared with existing studies (Boyd-Graber et al., 2013; Chang et al., 2018).

| | ***Goodreads*** | | ***TV Tropes*** | |
|---|---|---|---|---|
| | AUC | AUC(d.) | AUC | Acc. |
| **SVM** | 0.744 | 0.790 | 0.730 | 0.657 |
| + item-spec. | 0.746 ↑ | 0.800 ↑ | 0.747 ↑ | 0.653 ↓ |
| + bias | 0.864 ↑ | 0.793 ↑ | 0.722 ↓ | 0.536 ↓ |
| **SVM-BOW** | 0.692 | 0.729 | 0.756 | 0.702 |
| + item-spec. | 0.693 ↑ | 0.734 ↑ | 0.774 ↑ | 0.710 ↑ |
| + bias | 0.838 ↑ | 0.742 ↑ | 0.753 ↓ | 0.704 ↑ |
| **CNN** | 0.777 | 0.825 | 0.774 | 0.709 |
| + item-spec. | 0.783 ↑ | 0.827 ↑ | 0.790 ↑ | 0.723 ↑ |
| + bias | 0.812 ↑ | 0.822 ↓ | 0.781 ↑ | 0.711 ↑ |
| - word attn. | 0.898 ↓ | 0.880 ↓ | 0.760 ↓ | 0.695 ↓ |
| - word init. | 0.900 ↓ | 0.880 ↓ | 0.702 ↓ | 0.652 ↓ |
| - sent. encoder | 0.790 ↓ | 0.836 ↓ | - | - |
| **HAN** | 0.901 | 0.884 | 0.783 | 0.720 |
| + item-spec. | 0.906 ↑ | *0.889* ↑ | *0.803* ↑ | 0.733 ↑ |
| + bias | 0.916 ↑ | 0.887 ↑ | 0.789 ↑ | 0.729 ↑ |
| **SpoilerNet** | *0.919* | *0.889* | *0.803* | *0.737* |

Table 1: Spoiler sentence detection results on ***Goodreads*** and ***TV Tropes***, where arrows indicate the performance boost (↑) or drop (↓) compared with the base model in each group. Best results are *highlighed*.

**Results.** Spoiler detection results are presented in Table 1, where the complete **SpoilerNet** model consistently and substantially outperform baselines on both datasets. The accuracy that **SpoilerNet** achieved on ***TV Tropes*** beats the highest one among existing methods without using external item genre information (0.723), but is slightly lower than the best published result (0.756) where a genre encoder is applied (Chang et al., 2018). We notice adding the item-specificity and user/item bias generally improves the performance of most baselines except **SVM** on ***TV Tropes***. We find the

pre-trained word embedding initialization is particularly important on ***TV Tropes***. One possible reason could be that the model capacity is too large compared with this dataset so that it easily overfits without proper initialization. Note that a substantial performance drop can be observed by removing the sentence encoder on ***Goodreads***, which validates the importance of modeling sentence dependency in this task.

## 5 Error Analysis

We provide case studies to understand the limitations of the proposed model. We show review examples for three popular books ***Murder on the Orient Express***, ***The Fault in Our Stars***, and ***The Hunger Games*** respectively. For each example, we provide the review text, the groudtruth spoiler tags (i.e., if a sentence contains spoilers or not) and the predicted spoiler probabilities from **SpoilerNet**.

**Distracted by Revelatory Terms.** We find the majority of false positively predicted sentences from **SpoilerNet** can be found in this category. As shown in Table 2, the proposed network could be easily distracted by revelatory terms (e.g. 'murder', 'killed'). This leads to a potential direction for improvement: emphasizing 'difficult' negative sentences with revelatory terms during training (e.g. by 'hot' negative sampling) such that the semantic nuances can be addressed.

| Prob. | Label | Review Text |
|---|---|---|
| 0.35 | False | Language: Low (one/two usages of d*mn) |
| 0.32 | False | Religion: None |
| 0.39 | False | Romance: None |
| 0.59 | False | Violence: Low (It's a murder mystery! Someone is killed, but it is only ever talked about.) |

Table 2: An example review for the book ***Murder on the Orient Express***.

**Distracted by Surrounding Sentences.** Although the model is able to capture the 'coagulation' of spoilers (i.e., spoiler sentences tend to appear together), it can be distracted by such a property as well. As presented in Table 3, the third sentence was mistakenly predicted possibly because it immediately follows a spoiler sentence and contains an item-specific revelatory term (the character name 'Hazel'). This indicates the current model still needs to comprehend fine-grained sentence dependencies, so that it can decide whether to propagate or ignore the surrounding spoiler signals under different contexts.

| Prob. | Label | Review Text |
|---|---|---|
| 0.08 | False | This is not your typical teenage love story. |
| 0.86 | True | In fact it doesn't even have a happy ending. |
| 0.70 | False | I have to say Hazel with all her pragmatism and intelligence has won me over. |
| 0.43 | False | She is on the exact opposite side of the spectrum than characters like the hideous Bella Swan. |

Table 3: An example review for the book ***The Fault in Our Stars***.

**Inconsistent Standards of Spoiler Tags.** We find some self-reported labels are relatively controversial, which also verifies our suspicion regarding the subjectivity of spoiler tags. As shown in Table 4, the last sentence was classified as 'non-spoiler' by the language model, while reported by the review author as the opposite, probably due to its close connection to the previous spoiler sentence. Note that such an example is difficult to justify even by human annotators. This motivates us to consider spoiler detection as a ranking task instead of conventional binary classification. In this way sentences can be legitimately evaluated in the same context (e.g. the same review document) regardless of absolute thresholds. Besides the evaluation metrics, ranking losses can also be considered in future studies.

| Prob. | Label | Review Text |
|---|---|---|
| 0.01 | False | The writing is simplistic, a little more so than befits even the 1st-person narrative of a 16-year-old. |
| 0.50 | True | One of things I liked best about this is having a heroine who in addition to acting for the cameras, also has to fake her affection to someone who reciprocates far more than she feels. |
| 0.15 | True | I found it very relatable. |

Table 4: An example review for the book ***The Hunger Games***.

## 6 Conclusions and Future Work

Our new dataset, analysis of spoiler language, and positive results facilitate several directions for future work. For example, revising spoiler contents in a 'non-spoiler' way would be an interesting language generation task. In addition to review semantics, syntax information could be incorporated in a spoiler language model. The ***Goodreads*** dataset may also serve as a powerful spoiler source corpus. Models and knowledge learned on this dataset could be transferred to other corpora where spoiler annotations are limited or unavailable (e.g. detecting spoilers from tweets).

## References


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, Hyunjae Kim, Raehyun Kim, Deahan Kim, and Jaewoo Kang. 2018. A deep neural spoiler detection model using a genre-aware attention mechanism. In *PAKDD*.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *EMNLP*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*.

Sheng Guo and Naren Ramakrishnan. 2010. Finding the storyteller: Automatic spoiler tagging using linguistic cues. In *COLING*.

Hidenari Iwai, Yoshinori Hijikata, Kaori Ikeda, and Shogo Nishida. 2014. Sentence-based plot classification for online review comments. In *WI-IAT*.

Sungho Jeon, Sungchul Kim, and Hwanjo Yu. 2013. Don't be spoiled by your friends: Spoiler detection in TV program tweets. In *ICWSM*.

Armand Joulin, Edouard Grave, Piotr Bojanowski, Matthijs Douze, Hervé Jégou, and Tomas Mikolov. 2016. Fasttext.zip: Compressing text classification models. *CoRR*, abs/1612.03651.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *EMNLP*.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

George Loewenstein. 1994. The psychology of curiosity: A review and reinterpretation. *Psychological bulletin*, 116(1):75.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alexander J. Smola, and Eduard H. Hovy. 2016. Hierarchical attention networks for document classification. In *NAACL*.