Detecting Spoilers from Goodreads Book Reviews

Udayan Joshi
A53319215

Amogha Sekar
A53301791

Akshaya Raju
A53301016

Abstract

Due to the heavy use of internet, it has become impossible to prevent oneself from spoilers posted in popular social networks. The aim of this study is to develop an effective machine learning model to detect spoilers in book reviews present in Goodreads. This paper focus on understanding words and sentences in a review text and categorizing it as a spoiler or non-spoiler review. We carefully analyzed the Goodreads spoiler dataset that includes fine-grained spoiler annotations at the sentence-level, as well as book and (anonymized) user information. We understood that spoiler language common across books and users contains syntactically related word pairs. At a granular level, spoilers for a specific book contain certain words, distributed across the corpus. Using these observations, we developed a machine learning pipeline to detect spoiler sentences in the review corpus. Quantitative results prove that the proposed method outperforms existing baselines.

1 Introduction

Spoiler is a description of a key plot information in a book. Due to the early disclosure, it may ruin the mood of a first-time reader. To provide a good user experience, it is essential for reviewers to consider this and provide reviews without revealing any significant detail of the plot. However, not always self-annotations are correct. In order to aid reviews to be user-friendly, we rely on machine learning techniques to automatically detect spoilers from review texts.

2 Literature Review

Spoiler detection is a relatively unexplored domain. In pursuit of information and knowledge on existing work in this field, we came across the following work:

Guo and Ramakrishnan proposed a topic model [1] based on LDA for detecting spoilers [2]. They experimented their model on the reviews of four movies collected from the IMDb. They presented the problem as a ranking problem. Reviews that contain more spoiler materials are targeted to rank higher than the others. They also used dependency parsers to extract features instead of the traditional bag-of-words(BOW). One of the major drawbacks of using LDA is that, it requires setting a number of topics in advance. In spoiler detection, setting the number of topics is irrelevant to the problem. Another drawback of this LDA-based ranking model is setting the number of reviews considered to be spoilers. The performance matrices of their dataset varied significantly based on this number.

Maeda et al.[3] conducted a study on the location to which the content of the spoilers correspond in the story documents. They hypothesized that spoilers occur in the latter half of a story document. They experimented their study on five novels and reviews collected from Japanese online review site, Booklog. Their results are quite inconclusive as the words of high frequency from latter portion of a story both appear in spoiler and non-spoiler reviews. The proposed method also cannot support words that are not directly used in the story documents.

Preprint. Work in progress.

For detecting spoilers, Boyd-Graber first proposed a linear-kernal SVM using some metadata based features [4]. These features are Genre, Length, First Aired, Country and Episodes. These are proposed along with the traditional baseline features such as BOW and n-grams. Their results showed that these additional features increase the overall accuracy. The major drawback of their proposed method is that the additional meta-information were collected manually from different sites, such as IMDb, episode guides and TV tropes. In 2016, four distinguished features, Named Entity, Frequently Used Verb, Objectivity and Url, and Main Tense of Tweet, are proposed to detect spoilers in tweets [5]. However, these features come with their own limitations. Firstly, the frequently used verb list was set based on a manual study. Moreover, the context of a reality show or any sports is also quite straight forward. On the other hand, the contexts of books are quite distinct from each other. In these cases, the proposed features will not perform as well as in their experiment.

Deep learning methods were first applied to this task by a recent study [6], where the focus is modeling external genre information. Due to the lack of data with complete review documents and the associated user and book information, the user/item spoiler bias, as well as the sentence semantics under different item contexts, have never been studied in this domain.

The Goodreads spoiler dataset used for this project contain reviews from the Goodreads book review website, along with annotated spoiler information from each review[7]. Sentences are annotated as "1" if the sentence contains a spoiler, "0" otherwise. It is to be noted that there aren't significant historical work using this dataset. Past work with this dataset involves an end-to-end neural network architecture to detect spoiler sentences. However,since this model uses only semantic information of the review, it was prone to errors due to revelatory terms and surrounding sentences.

Recently, Ueno et al. in 2019[8], proposed a method for classifying between spoiler and non-spoiler sentences in Japanese-written reviews of products with stories.This method performs two kinds of learning: word vector learning using fastText and classification learning using a Long Short-Term Memory network. Their method worked well with Japanese-written reviews, which often included slang, new words, and subject-omitted sentences. This idea can be adopted for English reviews as well to test whether it handles slang and new words, since subject-omitted sentences don't constitute much in the English language.

3 Proposed Work

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 ($y_s = 0$). In order to this, we have separated the Goodreads spoilers dataset into 60% for training, 20% for validation and 20% for testing. Since the dataset is quite huge and owing to lack of high computation power, we have chosen 100,000 reviews randomly to perform our study.

3.1 The Goodreads Spoilers Dataset

The Goodreads book reviews dataset annotated with spoiler(1 for spoiler and 0 for non-spoiler) scraped by Wan et. al is being used for this study. Totally, there are 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[7].

By intuition, we believed that review length might help in predicting spoilers, since people tend to write longer reviews when they have more to say about the plot. We plotted histogram of length of reviews for spoiler and non-spoiler reviews to explore on this area. The observations were made are as per Figure 1. From the plot it is evident that there are distinct peaks for the two classes. As per our expectation, reviews with larger word count are more likely to contain spoilers. Another interesting observation is that most of the reviews with no spoilers have word count between 0 and 500. This further supports our intuition that review length is correlated with whether the review contains a spoiler. Further, we tried to understand what words appear frequently in spoiler reviews. In order to visually represent it and get to know better, we adopted the Wordcloud technique. A word cloud is a visual representation of text data, typically used to depict keyword metadata. The importance

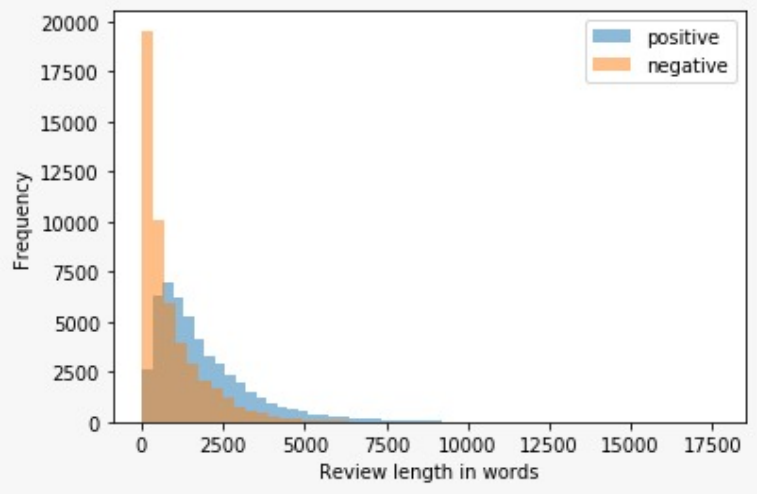

Figure 1: Plot of Review Length vs Frequency

of each word is shown with font size or color. This format is useful for quickly perceiving the most prominent terms to determine its relative prominence[10]. As per Figures [2] [3], we observe that some words are more common in one class than the other. Words like 'end' and 'still' are much more common in reviews with spoilers, while words like 'love' are common in reviews without spoilers.

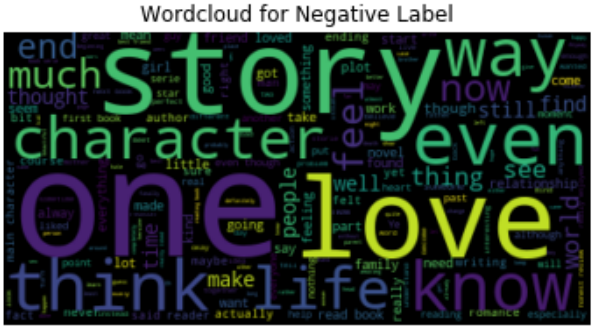

Figure 2: Wordcloud of non-spoiler reviews

3.2 Baseline Model

We use words as features for identifying sentences as plot descriptions or not. This method judges each sentence of new user reviews using a learned classifier as a plot description or not. Sentence p is represented in a bag-of-words for conducting machine learning as

$$p = w_1, w_2, ..., w_M(1) \tag{1}$$

where, w_m, stands for a word. The number of occurrences x_n,m of each word w_m in sentence p_n is also recorded. For creating bag-of-words model, we removed stop words. The Porter stemming algorithm is conducted to remove morphological and inflectional endings from words. Using this as features, we trained the following classification models, chosen carefully from previous studies and proven to work well for binary classification problems:

Figure 3: Wordcloud of spoiler reviews

1. Balanced Logistic Regressor
2. Naive Bayes Classifier
3. Decision-tree based Classifier

Specifically, for Logistic Regression, we applied regularization for all values of C= 0.00001 to C= 10000. We found that for our unbalanced BOW model, C= 0.001 works the best.

For Naive Bayes, we applied 4 forms of it, namely, Gaussian Naive Bayes, Bernoulli Naive Bayes, Multinomial Naive Bayes and Complement Naive Bayes. We found that for our unbalanced BOW model, Complement Naive Bayes worked the best.

3.2.1 Evaluation:

Due to the possible bias of users' self-reported spoiler tags, we regard the area under the ROC curve (AUC) as our primary evaluation metric. Imbalanced class distribution exists and so F1-score is also a better metric to evaluate our model on.

Model	Accuracy	Precision	Recall	F_1 score	AUC score
Logistic Regression with C= 0.001	80.65096	0.18617	0.63136	0.28755	0.72471
Complement Naive Bayes	71.52142	0.13444	0.66289	0.22355	0.69077
Decision Tree	89.46052	0.18095	0.19967	0.18985	0.57004

Table 1: Evaluation metrics of baseline model

As the labels are highly imbalanced, accuracy is not a good metric to evaluate our model. We should compare metrics like F_1 score or AUC score. We observe that Logistic Regression with C=0.001(This is tuned using validation set) performs best compared to other models. Another interesting observation is that Decision tree classifier performs best with respect to accuracy metric, but has very poor AUC score. This suggests that this model is no better than a trivial classifier.

3.3 Improvisations on the Baseline Model

Based on the evaluation, we observe that the Bag-of-words feature doesn't necessarily perform well. In order to improve our baseline model, we consider the following methods:

3.3.1 Adding more relevant data

From the exploratory analysis, it is observed that only 6% of the reviews contains spoilers. This Imbalance can be reduced to a great extent by Under-sampling the majority class to drive the variance down. AUC does not place more emphasis on one class over the other, so it does not reflect the minority class well. Random under-sampling for the majority class is equipped here. At the end of this step, we will have equal number of positive and negative smaples in the dataset. Now, as the dataset is balanced, accuracy becomes a good metric on which we can evaluate out models.

Model	Accuracy	Precision	Recall	F1 score	AUC score
Logistic Regression with C= 1	75.575%	0.79366	0.686473	0.73618	0.75526
Multinomial Naive Bayes	68.74%	0.68670	0.68103	0.68385	0.68735
Decision Tree	66.305%	0.66209	0.65615	0.65911	0.66300

Table 2: Evaluation metrics after under-sampling

We observe that the Logistic Regression model with C=1, performs best according to the all the metrics.

3.3.2 Adding relevant new features

Adding new features increases model flexibility and decreases bias. The below meaningful feature is extracted and the baseline algorithms are run again:

- Item-Specificity[7]:

  Since book-specific terms could be reveal plot information[9], we develop an effective method to identify the specificity of tokens regarding each item (i.e., each book).

  1. Popularity - For word w, item i, we calculate the item-wise document frequency (DF) as
  $$DF_w, i = |D_w, i|/|D_i| \tag{2}$$
  2. Uniqueness - For each word w, we calculate its inverse item frequency (IIF) as
  $$IIF_w = (log|I| + \epsilon)/(I_w + \epsilon) \tag{3}$$
  3. For each word w in item i, we calculate the DF-IIF as $DF_w, i*IIF_w$.

Model	Accuracy	Precision	Recall	F1 score	AUC score
Logistic Regression with C= 1	76.43%	0.76431	0.75939	0.76184	0.76426
Multinomial Naive Bayes	70.16%	0.68845	0.72867	0.70799	0.70179
Decision Tree	65.29%	0.65288	0.64236	0.64757	0.65282

Table 3: Evaluation metrics after adding TF-IDF features

We observe that the Logistic Regression with C=1 performs best according to all metrics even in this case. We also improve upon the simple Bag of Words model by almost 1% by using TF-IDF feature vectors.

- Named Entity Score[9]:

  The names of the central characters in a book play an important role in the spoilers. If we only use frequency as a score of named entity, its influence can be changed as a function of the data size. Thus, we can divide the sum of the total frequency and the frequency of the named entity. This can be tried in the future and get added to the baseline model to improve.

3.3.3 Trying advanced classification techniques

Random Forest classifiers work well with sparse, non-linearly-separable data sets with lots of correlations. This can also be tried with enough computation power on the large dataset in the future.

4 Analysis of Observations

As seen from Table [4], as we incorporate more features and balance the dataset, we can see the following observations:

- Accuracy:

  When it comes to considering accuracy as a metric, the baseline model performed better than the balanced TF-IDF model. The decision tree model provides the best accuracy, with a value 80.65096%. However, accuracy is not really a good metric to evaluate our model on, in this case, since our dataset was highly imbalanced and later on due to undersampling we balanced it.

- F1 Score:

  The TF-IDF model shows a significant improvement over the baseline model when F1 score is considered as a metric. Specifically, the Logistic Regression model with C= 1 performs the best, with value 0.76184.

- AUC score:

  The TF-IDF model shows a significant improvement over the baseline model when AUC score is considered as a metric. Specifically, the Logistic Regression model with C= 1 performs the best, with value 0.76426.

Model	Metric	Logistic Regression	Naive Bayes	Decision Tree
Baseline	Accuracy	80.65096%	71.52142%	89.46052%
	F1 score	0.28755	0.22355	0.18985
	AUC score	0.74721	0.69077	0.57004
Improved model	Accuracy	76.43%	70.16%	65.29%
	F1 score	0.76184	0.70799	0.64757
	AUC score	0.76426	0.70179	0.65282

Table 4: Overall comparison

Therefore, the model which works best on the Goodreads Spoiler dataset is the logistic regression model with regularization parameter set to 1, taking F1 score and AUC score as the performance metrics.

5 Challenges, Future Work and Conclusion

Since spoiler detection is a relatively unexplored domain and the Goodreads dataset is also quite new, there is a lot of scope to do further research.

1. Based on the previous study and our own understanding, we believe that using new features such as Genre of the book and the Year published can help in classifying spoilers and non-spoilers effectively[4]. This is because some genres are more likely to have spoilers (Ex:“mystery”) than others (Ex: “family”). And there are high chances that old books might have more spoilers since classics tends to get more reviewed in depth. However, in order to try out these features, we don’t have the meta data available in the existing Goodreads spoilers dataset.
2. Syntax information can be understood from the reviews to predict whether they are spoilers or not. A typed dependency parser could be implemented which represents dependencies between individual words and also labels dependencies with grammatical relations such as subject or indirect object. The motivation behind using syntactically related word pairs instead of n-grams is to capture the concept of the text more effectively, and to introduce minimum noise. These features can be extracted once we have sufficient computing power and NLP domain knowledge.

3. In order to try advanced deep learning methods such as RNNs and LSTMs, we require large computation power which is currently unavailable.
4. Much of the common spoiler language appears to be strong sentiment words. We need to explore more on sentiment analysis and fairly predict whether a review contains spoilers or not based on the sentiment, rather than having to learn the details of every individual book.

References

[1] S. Guo and N. Ramakrishnan, "Finding the storyteller: automatic spoiler tagging using linguistic cues," in Proceedings of the 23rd International Conference on Computational Linguistics, pp. 412–420, 2010.

[2]H. M. Wallach, "Topic modeling: Beyond bag-of-words," in Proceedings of the 23rd International Conference on Machine Learning, pp. 977–984, 2006.

[3] K. Maeda, Y. Hijikata, and S. Nakamura, "A basic study on spoiler detection from review comments using story documents," in International Conference on Web Intelligence, pp. 572–577, 2016.

[4] J. Boyd-Graber, K. Glasgow, and J. S. Zajac, "Spoiler alert: Machine learning approaches to detect social media posts with revelatory information," in Proceedings of the 76th ASIST Annual Meeting: Beyond the Cloud: Rethinking Information Boundaries, ASIST '13, pp. 1–9, 2013.

[5] S. Jeon, S. Kim, and H. Yu, "Spoiler detection in TV program tweets," Information Science, vol. 329, pp. 220–235, 2016.

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

[7] Fine-grained spoiler detection from large-scale review corpora Mengting Wan, Rishabh Misra, Ndapa Nakashole, Julian McAuley ACL, 2019.

[8] Atsushi Ueno, Yuu Kamoda and Tomohito Takubo, "A Spoiler Detection Method for Japanese-written reviews of stories", International Journal of Innovative Computing, Information and Control", Volume 15, Number 1, pp. 189-198, February 2019.

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

[10] Hassan-Montero, Y., Herrero-Solana, V. Improving Tag-Clouds as Visual Information Retrieval Interfaces Archived 2006-08-13 at the Wayback Machine. InSciT 2006: Mérida, Spain. October 25–28, 2006.