University of California, San Diego

CSE 158 Fall 2020

Assignment 2 Report

Yanxun Li

Xinyue “Ellie” Yu

Chongbin “Bob” Zhang

Zhenxuan “Paul” Zhu

Abstract

Nowadays, increasing amounts of spoilers are coming at us and could be devastating to our experience when reading a book. So we want to study the behavior of the spoilers and train a predictive model to detect spoilers. Among all 3 models we experimented, Random Forest performed the best, which is the one we recommend for future use.

Dataset

We chose to do the "Goodreads Spoilers" dataset to explore for this assignment. The dataset comes with 25,475 books, 18,892 users, and 1,378,033 reviews in total. The reviews within are structured as the following:

- review_sentences: list of sentences with each sentence labeled 1 if it is a spoiler and 0 otherwise.
- has_spoiler: boolean that indicates whether this review contains spoilers or not.
- user_id
- book_id
- review_id
- rating
- timestamp

While the whole dataset contains 1,378,033 reviews and 17,672,655 review sentences, there are only 89,627 spoiler reviews and 569,724 spoiler review sentences. Spoiler reviews take 3.22% of whole reviews and spoiler sentences take 6.50% of the whole review sentences. In one word, spoilers only take a small portion of the whole dataset.

We've also found that the percentage of spoiler sentences in terms of the total number of sentences in a review varies a lot across all the reviews, as demonstrated in Figure 1 as a boxplot. This means that some spoiler reviews have only very few, if not zero, spoilers, while other reviews have several spoilers.

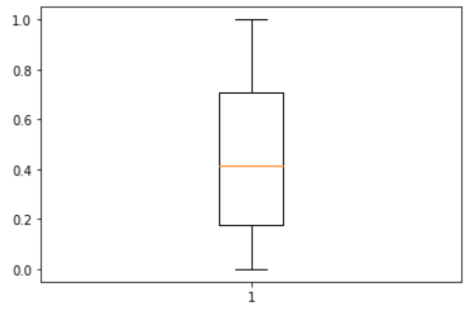

Figure 1. the percentage of spoiler sentences in terms of the total number of sentences in a review

To explore the relationship between ratings and the percentage of spoiler reviews, we collected all reviews' ratings and counted how many reviews per rating are spoilers. The above exploration gave us Figure 2 (on the next page). The graph suggests that reviews that have high ratings (4-5) are less likely to have spoilers. People who give low ratings (1-3) tend to post a spoiler review, but people who give 0 tend not to post a spoiler review.

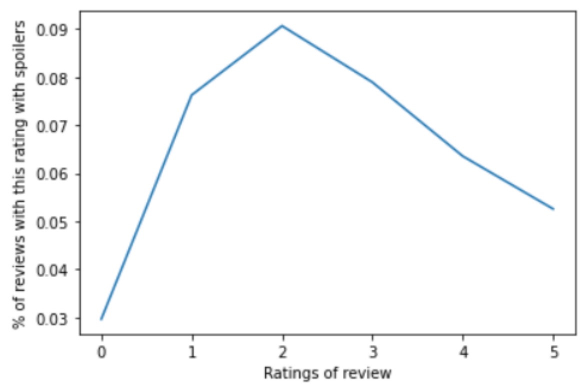

Figure 2. the relationship between ratings and the percentage of spoiler reviews

We were also curious if lengths of sentences are correlated to the likeliness of having spoilers, and if so, how. In Figure 3, we plotted the relationship between length of each individual sentence and percentage of sentences shorter than X many words with spoilers, with X being the numbers on the x-axis. It is true that longer sentences are more likely to contain a spoiler when the length of sentences are less than 100 words, but the adverse if true when sentences are longer than 140 words - turn out those sentences are less likely to contain a spoiler!

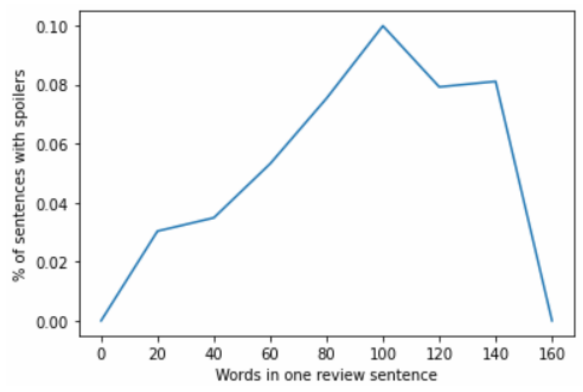

Figure 3. the relationship between sentence length and the percentage of spoiler reviews

Similar to Figure 3, we studied the relationship between length of each review (by summing up lengths of all sentences within) and the likelihood of having spoilers, demonstrated in figure 4. We noticed a similar "turning point" pattern: before the length reviews reach 2800 words in length, the longer they are, the more likely for them to contain a spoiler, while after 2800 words the adverse is true.

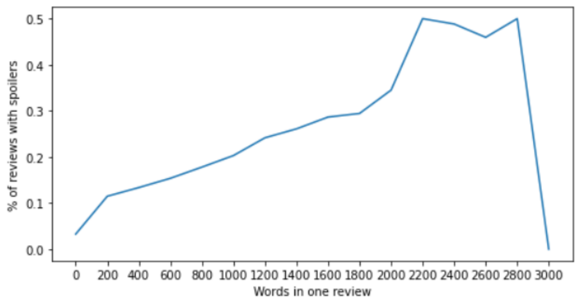

Figure 4. the relationship between review length and the percentage of spoiler reviews

There is a possibility that some reviewers prefer to leave spoiler reviews and some books tend to be spoiled more frequently than other books. In Figure 5 (on the next page), we plotted the percentage of spoiler reviews of the top 15 reviewers who post the most reviews, and in Figure 6 (on the next page as well) the percentage of spoiler reviews of the top 15 books that have the most reviews. It turns out that people who post the most reviews are very likely to post spoiler reviews. And books that have the most reviews do not have a high percentage of spoiler reviews.

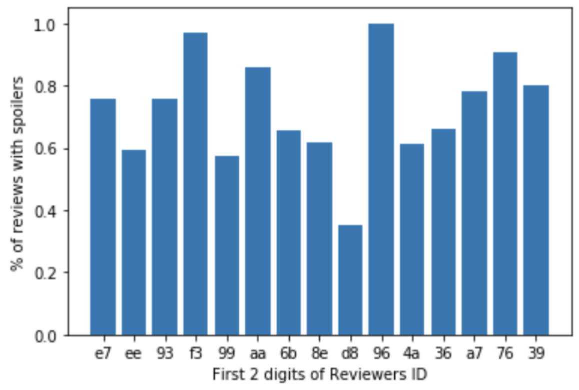

Figure 5. the percentage of spoiler reviews of the top 15 reviewers who post the most reviews

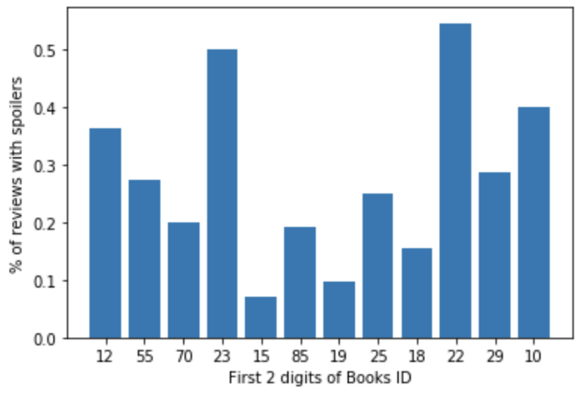

Figure 6. the percentage of spoiler reviews of the top 15 books that have the most reviews

Lastly, to confirm if reviews in recent years are more likely to have spoilers compared to earlier years, we plotted the relationship between the number of spoiler reviews and time in Figure 7. It confirms that reviews in recent years (which come later in timestamp) have a higher percentage of spoilers.

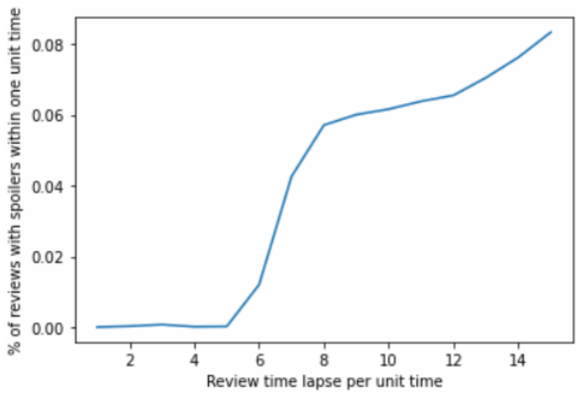

Figure 7. the relationship between the number of spoiler reviews and time from Dec. 7th, 2006 to Nov. 5th, 2017. One unit time is approximately 0.7 years, or 8.5 months.

It is likely that users explicitly mention "spoiler" or "spoil" when they are writing a spoiler review. So we drew a sample of 88000 data and calculated the explicit "spoil" words in spoiler reviews. It turns out that for all reviews that contain the stemmed word "spoil", 73% of the reviews are spoiler reviews. So a review that contains words "spoiler" or "spoil" is very likely to be a spoiler review. In our dataset, only 14% of reviews that contain spoilers explicitly mention the word "spoiler" as an alert for potential readers. Hence, it's common that readers come across spoilers unexpectedly, so we wish to train a model that predicts if a review contains spoilers or not.

Predictive Task

Our predictive task is a binary classification problem of whether a review contains spoilers or not.

We came up with the following baseline: use a bag of words of the top 1000 of most frequent words across all the review sentences to create feature vectors for each sentence, and build a logistic regression model on randomly sampled 80,000 training data. The accuracy of this model on our validation set consisting of randomly sampled 20,000 data is 92.8% and the corresponding balanced error rate is 48%. The sensitivity is 4.6%, defined as:

$$\frac{True\ Positive}{True\ Positive + False\ Negative}$$

, and the specificity is 99.3%, defined as:

$$\frac{True\ Negative}{True\ Negative + False\ Positive}$$

For this predictive task, although it is important to keep a high accuracy, we also need to take the balanced error rate into consideration. More specifically, we would like to maximize the sensitivity. This is because even if our model accidentally marked a non-spoiler review as a spoiler review, there are a lot more non-spoiler reviews for the users to read, so it would not hurt user experience if our model had a slightly higher false positive error rate. However, if our sensitivity is low and has a high false negative error rate (marked spoiler reviews as non-spoilers), users are going to read those reviews, which is exactly what we want to avoid by doing this prediction task.

Therefore, to assess the accuracy/validity of our model, we will consider both accuracy, balanced error rate, specificity and sensitivity. Out of them, sensitivity is the most important metric to measure. In case of the same accuracy, a higher sensitivity is better than a higher specificity.

Features we used in this prediction task are the following:
1) A bag of unigrams and bigrams extracted from spoiler reviews, because some words or phrases probably appear more in spoiler reviews than non-spoiler reviews,
2) The percentage of spoiler reviews over all reviews for each user, because users who spoil in most of their posts tend to spoil in other posts while users who don't often spoil tend to be careful of not spoiling,
3) The percentage of spoiler reviews over all reviews for each book, because books being spoiled in most of its posts tend to be spoiled in its other posts,
4) The rating of the review, because rating is associated with the percentage of spoiler reviews,
5) Whether the word 'spoiler' appears in the review, because ~73% reviews with the word 'spoiler' actually contains spoilers,
6) The length of the review, because longer reviews tend to contain spoilers, and lastly
7) The inverse document frequency (IDF) of each word in the review, because spoiler reviews usually contain some unusual words.

For features relating to text-based information, we process the text by the following methods: removing punctuation, converting them to lowercase, splitting them into lists of individual words, removing stopwords since they are not predictive, and stemming the words to remove the effect of tense.

We also noticed that our dataset is imbalanced - only ~7% of the data contains spoilers. To tackle this problem, we built a dataset of 50% positive data and 50%

negative data, and split them into training - validation - test set with a percentage of 50% - 25% - 25% specifically.

Since our predictive task is a binary classification problem, the models we chose are: Logistic Regression model, Decision Tree model, and Random Forest model. Logistic Regression is a general model for binary classification problems, while Decision Tree is able to explain non-linearity in our feature space and Random Forest model is an improvement of Decision Tree model. Detailed information will be explained in the next section.

Model

We tried fitting our features with three different types of commonly used machine learning algorithms - Logistic Regression, Decision Tree, and Random Forest.

Logistic regression is used in the first place to detect whether there is a general correlation between the features we chose and the predicted label. Specifically, we fit logistic regression on the following features - bag of unigrams + bigrams, rating of the review, whether the word "*spoiler*" is in the review, and length of the review - and find correlation between the label to predict and each of them. We also exclude features that do not show correlation with the label when trained with Logistic Regression, such as the percentage of spoiler reviews among all reviews of a book. However, when we combine all features we chose, the Logistic Regression model appears to be underfitting the input features since the accuracy does not improve much when we add additional features to the model. In the meantime, increasing C to 1000 to reduce regularization strength gives us the best accuracy and further increasing C-value has no effect on the accuracy, which also indicates the model is underfitting. Also, the inclusion of a bag of words causes each of our samples to have 500 - 1000 columns, leading to the result that we have to use a large `max_iter` hyperparameter to account for the slow convergence of the model.

Therefore, we switched to a decision tree to account for the non-linearity in our feature space. However, if we do not limit the `max_depth` parameter of the decision tree, it exhibits a strong pattern of overfitting. We fine-tune the model by setting a `max_depth` to counter overfitting. Empirically, a `max_depth` of 10 works best on both the training set and the validation set, and we witness an increase of accuracy by 4-5% compared to the one given by Logistic Regression.

Then we switched to Random Forest, which is an improvement of the decision tree algorithm and is also known as a more powerful model. Random forest includes a group of decision trees that have low correlation with each other, and predict the label by counting the vote of each tree. Random forest can easily overfit on the training sample as it also happens in ours, but it also exhibits better ability to generalize by producing a higher accuracy on the validation set and test set than decision tree. Therefore, we didn't limit the `max_depth` of our random forest model, instead we used Gini impurity as the criterion of the model. By switching to Random Forest, our accuracy on the validation set improved by approximately

2% compared to using a Decision Tree (which is a 7% improvement from logistic regression).

Out of all the features we used, the most successful one is “the number of words with a IDF score greater than a certain threshold in a review”. We determined the threshold by using a percentile of idf scores of all words in the training set. We tested percentiles from 75 to 40 and empirically, the model gives the best result when we set the threshold to 50 percentile (the median of IDF score of all words).

We encountered the issue of slow training when we sample a larger bag of words. We initially chose a bag-size of 1000, which forced us to increase the maximum number of iterations of the Logistic Regression model and the training is dramatically slowed. Through experiments, we discovered that reducing bag size from 1000 to 500 only decreases the training accuracy by ~0.3%. Further reducing bag size to 250 reduced the accuracy by ~0.9%. On the other hand, increasing the size of the bag to 2000 only improves the accuracy by 0.5% but the run time is extremely long. So eventually we decided to use 500 as our bag size for a faster model with the tradeoff of slightly lower accuracy. Using a smaller bag size significantly improves the speed of Logistic Regression and Random Forest.

Bag size	250	500	1000	2000
Accuracy (%)	71.95	72.87	73.20	73.69
Time (sec)	53.0	112.2	458.8	1001.7

Figure 8. the relationship between size of the word bag, accuracy and run time

We also encountered the issue of overfitting when including the feature “percentage of spoilers by user” in the training set. This is probably because we arbitrarily sampled an equal number of positive sample and negative sample in the training set, making the percentage of spoiler of each user to be much higher than the real percentage in the population. We then changed this feature to the raw count of the spoiler posted by a user in the training set, which alleviated the problem of overfitting.

When comparing the performance of different models we used, we found that Logistic Regression is relatively less prone to overfitting and is better at generalizing for the validation set and test set. However, it’s not complex enough to fit a large number of models and discover the non-linear pattern in the feature space. Compared to Logistic Regression, Decision Tree is less vulnerable to outliers, but is more vulnerable to overfitting than logistic regression especially when the training samples are sparsely distributed. Random Forest has similar properties as Decision Tree, but even though it overfits on the training set, it still performs better on the validation set and test set than the other two models, and therefore we stick with Random Forest as the model to fit our features.

Literature

We used an existing dataset from the course website. The advantage of this dataset is that it labeled every sentence of reviews to indicate whether it is a spoiler or not. This enables our model to learn the spoiler information at the sentence level, which could outperform the model learning at the entire review level.

We referred to *MOVIE REVIEWS: TO READ OR NOT TO READ! Spoiler Detection with Applied Machine Learning*, a movie review spoiler detection paper. The dataset they used is similar to our dataset with a few differences. Their dataset does not have labels associated with each review sentence, but it has movie plot summary and synopsis. That paper used state-of-the-art methods and achieved 85.6% accuracy. In their model, they took advantage of the movie plot description. Their approach was to score the similarity between the plot and the review by using 5-grams and extracting the top 20 of the most important 5-grams based on their positions and frequencies. The reviews were then scored based on how the review matched with the selected 5-grams from the plot. They also used a network map to calculate the centrality score - the average degree to which a word is related to other words in the document. Based on the centrality score and the frequency of each review, they chose 10 words with highest centrality score and 10 with lowest centrality score. Then they used these 20 words to compare with the top 20 5-grams from the plot and got a matching score, which was used to determine whether the review is a spoiler or not. Their conclusion is that their model is effective in predicting spoiler reviews. And in their model, position and frequency of words play important roles. These are the features we would like to incorporate into our model if possible.

We also referred to *Fine-grained spoiler detection from large-scale review corpora*, which analyzes the same dataset as we do. They used SpoilerNet, a deep learning approach to process the dataset and do the predictive analysis, such as passing words through bidirectional recurrent neural networks (bi-RNN). SpoilerNet proved to be very effective and could achieve 90% accuracy. Insights from that paper helped us develop our feature vectors by pointing out that:
1) spoiler sentences generally appear together in the latter part of the review,
2) TF-IDF could be useful to distinguish spoiler reviews, and
3) self-reporting spoilers across users and items can be helpful to predict spoiler reviews.

Our conclusion is very similar to the conclusions from the papers. They concluded that detecting review semantics and syntax information are essential in their models. Similarly, we found that incorporating frequency and uniqueness of words into feature vectors is important to our model to beat the baseline. Hence, the more text-based information we can extract from the review, the higher accuracy we can achieve.

Results

The model we chose in the end was Random Forest. It outperformed Decision Tree as the algorithm of Random Forest is an improvement from Decision Tree. The model also outperformed Logistic Regression since Logistic Regression isn’t complex enough to fit features we selected demonstrated by patterns of underfitting. Our model achieved a high sensitivity, meaning that if a review contains spoilers, our model can accurately detect it, to prevent spoilers from spoiling the reading experience of a reader.

We evaluated the performance of our best model, Random Forest, with various metrics mentioned in the *Predictive Task* section.

To measure the reliability of our model, we calculated the sensitivity of the model. The sensitivity of our model on the test set is 0.864. Meaning whenever our model predicts a review as a spoiler, there is 86.4% of probability that the review really contains a spoiler, and it indicates that our model has a decent reliability. As a comparison, the baseline model only has a sensitivity of 0.046. We also measure the specificity of our model. The specificity of our model on the test set is 0.695, meaning that whenever our model predicts a sample as not containing a spoiler, 69.5% of the predictions are correct. This is relatively low compared to sensitivity, but given our predictive task, our primary goal is to accurately detect those reviews that actually have a spoiler. Lastly, we calculated the BER of our model on a randomly sampled set from the entire dataset to measure how well our model performs on an imbalanced dataset (the entire dataset has 93% of negative samples and 7% of positive samples). The randomly sampled set has a size of 130000 data points and our model gives a BER of 0.22 on the set by correctly finding out most of the positive samples. As a comparison, the baseline model has a specificity of 0.99 as the training set in the baseline model is not balanced.

The feature that resulted in the largest improvement in accuracy was the modified IDF score. IDF measures the “uniqueness” of a word in the entire document, and in reviews containing spoilers usually feature high “uniqueness” in its wording by mentioning terms such as characters’ name, locations’ name, terminologies in the book. These unique words are usually related to the plot. We modify the calculation of IDF by removing the logarithm to allow highly unique words to be more differentiable from those that are not. The eventual usage is that for each review, we calculate the number of words that have a greater IDF score than the median (50 percentile) IDF score of all words in the document, and put this number as a column in the feature matrix.

As for features that negatively affected our model, aside from the percentage of spoiler reviews per one specific user/book mentioned above, we also noticed that combining TF with IDF performs less ideally than using solely IDF. While we never digged into the reason of this, our guess is that when TF (text frequency) is included, it multiplies the term frequency with the IDF, which is not useful in our case since we are training on only parts of the dataset instead of the whole thing, therefore TF might not represent the true frequency (occurrence) of a specific word.

As for parameters, the size of the bag of words has an influence on the accuracy of our model. Larger size can increase the accuracy, though at cost of the training time. This agrees with the assumption that a larger bag of words containing more text-based information can achieve higher accuracy. The value of `C` also made a difference when training the logistic regression since it directly affects the strength of regularization. Since data are relatively close to each other in the set, it'd make sense for us to decrease the regularization strength, which is why our logistic regressor reached its peak performance at `C=1000`.

In conclusion, our model has a decent performance on imbalance dataset and on finding out the spoiler reviews but may sometimes wrongly predict non-spoiler reviews as a spoiler review. The improvement of our model from the baseline is indicated by the fact that the BER has increased from 0.48 to 0.22.

Reference

Sreejita Biswas, Goutam Chakraborty, *MOVIE REVIEWS: TO READ OR NOT TO READ! Spoiler Detection with Applied Machine Learning*. SAS Global Forum, 2020. Available from here.

Mengting Wan, Rishabh Misra, Ndapa Nakashole, Julian McAuley. *Fine-grained spoiler detection from large-scale review corpora*. ACL, 2019. Available from here.