CSE 158 Assignment 2: Goodreads Spoilers

Link to Google Collab

ABSTRACT

For this Assignment, we chose a project to predict whether a book review contains a spoiler or not based on Professor McAuley's dataset: Goodreads Spoilers [1].

1. EXPLORATORY ANALYSIS

For our dataset, we used Professors McAuley's Goodreads Spoilers [1]. This dataset contained 1,378,033 reviews from Goodreads and annotated whether the book review contained a spoiler or not. Due to Google Collab's limitation, we took a random subset of this dataset to focus our predictions on. We initially trained our models on a random training dataset of 130,914 entries. However, in our dataset investigation, one of our biggest discoveries was that the dataset was dominated with non-spoiler reviews. This can be examined in Figure 1. Around 93% of the dataset was non-spoiler reviews. We ended up creating a random subset of 40,000 reviews for our training dataset and 5,000 for our validation dataset. We ensured that these training dataset and the validation had an equal amount of reviews with spoilers and non-spoilers. In order to evaluate performance on a realistic set of data, we created another test dataset of 68,902 reviews, which were randomly selected from the full dataset. The test dataset was a better representation of what we would actually find on Goodreads. We wanted our models to make predictions on both the validation and test set.

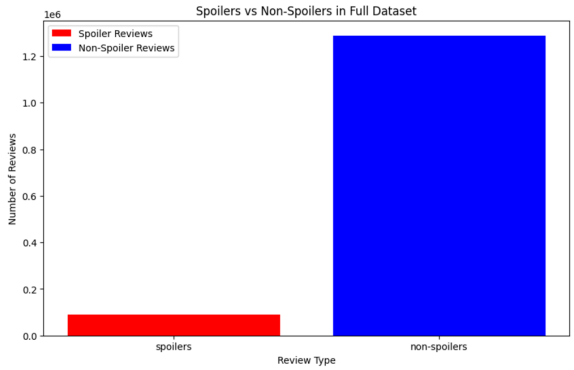

Figure 1:Spoiler vs Non-spoiler reviews for the full dataset.

Along with annotations of whether the review contained a spoiler or not, the dataset also included the rating the user gave the book. However, there did not appear to be a correlation between ratings and spoilers. The mean rating for reviews that contained spoilers was 3.58, while for non-spoilers was 3.70. There was only about a 3% difference. But there did seem to be a difference between the review length. The average length of spoiler reviews was 87.2 characters, while the average length of non-spoiler reviews was 78.48 characters. Here, there was about a 11% increase.

We also investigated the ratio of spoiler vs non-spoiler reviews per user. We found that on average, users tend to either have more reviews in one category or the other. As seen in Figure 2, this means that if a person has written a spoiler review in the past, there is a higher chance that

they will write a spoiler review again. If a user tends to write non-spoiler reviews, then they will usually write only non-spoiler reviews.

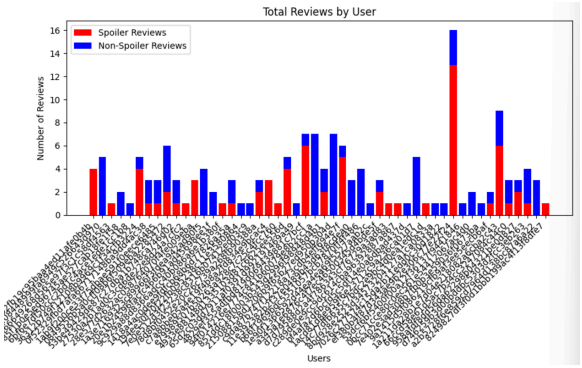

Figure 2: Random sampling of 50 users' total reviews from the training set.

Additionally, we examined the amount of spoilers per book. We wanted to see if some books had more spoiler reviews written about it compared to other books. As seen in Figure 3, it appears that some books are more spoiler prone than others. So if a book already has a spoiler written about it, there is a higher chance that the new review for the same book will also contain a spoiler.

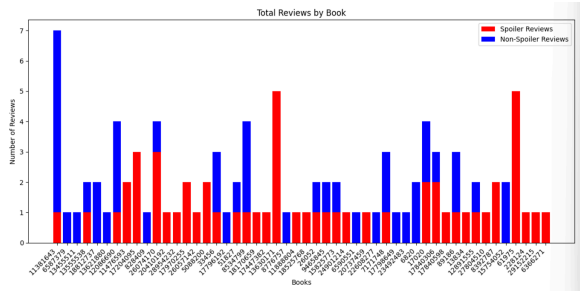

Figure 3: Random sampling of 50 books' total reviews from the training set.

2. PREDICTIVE TASK

2.1 Task

Our predictive task is to predict whether or not a Goodreads book review is a spoiler. A solid baseline we can begin with is a model that only predicts everything as a non-spoiler. Considering that our dataset is predominantly non-spoiler reviews, our baseline already does very well with a 93% accuracy. By creating this baseline and comparing it to the models we create, we will be able to more accurately evaluate the validity of our model to ensure the predictions are legitimate and classify spoiler reviews. If we are beating the baseline, then that indicates that we are correctly predicting spoiler reviews.

A model that will be appropriate for this kind of task would be a Word2Vec model. This model would indicate if a new review is more similar to a review with a spoiler rather than one without a spoiler. Another model that could be useful is taking into account a user's review history to indicate whether or not their review is going to be a spoiler, as we know from our data exploration that users who tend to post spoilers would be more likely to post more spoilers. We also look at a model that takes into account if the review is one from a book that already has a high number of spoiler reviews, as from our dataset exploration, we saw that certain books tend to have more spoiler prone reviews.

2.2 Relevant Features

From this dataset, there were 7 columns (user_id, timestamp, review_sentences, rating,

has_spoiler, book_id, review_id) that we could potentially work with.

A few of these columns stood out with relevant features for the prediction task of categorizing spoiler reviews. The columns included:

- 'has_spoilers':

  The 'has_spoilers' column is likely one of the most relevant features in this dataset. With this feature, we are able to create a supervised model that can accurately predict whether or not reviews are spoilers.

- 'book_id':

  The 'book_id' column is also another significant feature that can be implemented into our model as it serves to uniquely identify each book.

- 'user_id':

  The 'user_id' column is an important feature that may be implemented into a model as it uniquely identifies each user that gives reviews.

- 'review_sentences':

  The 'review_sentences' column could be another relevant feature to include in the model as the review itself should contain the spoiler.


```
{
  'user_id': '01ec1a320ffded6b2dd47833f2c8e4fb',
  'timestamp': '2013-12-28',
  'review_sentences': [[0, 'First, be aware that this book is not for
the faint of heart.'],
   [0, 'Human trafficking, drugs, kidnapping, abuse in all forms -
this story contains all of this and more.'],
   ...,
   [0, '(ARC provided by the author in return for an honest
review.)']],
  'rating': 5,
  'has_spoiler': False,
  'book_id': '18398089',
  'review_id': '4b3ffeaf14310ac6854f140188e191cd'
}
```


Figure 4: Example of the Spoiler Data

To obtain these features from the dataset, we utilized a pandas dataframe framework in order to retrieve individual columns and entries quickly and easily.

3. MODEL

3.1 Word2Vec

Our most accurate model was Word2Vec, which represented words as numerical vectors (word embeddings) based on their meanings and relationships to other words. We trained Word2Vec using the continuous bag of words (CBOW) approach, where each word was predicted using the context of surrounding words.

In our setup, words w were represented as 100-dimensional vectors V_w, with words of similar meanings having similar embeddings:

$$V_w = Word2Vec(w)$$

Each review was then represented as the average of all the word vectors within it, creating a sentence vector V_R. V_R was calculated as:

$$V_R = \frac{1}{n}\sum_{i=1}^{n} V_{w(i)}$$

Where n represents the number of total number of words in a review and $V_{w(i)}$ is the vector representation of the i-th word in the review. These sentence vectors served as input features for a random forest classifier. This classifier, composed of T decision trees $\{h_1,\ldots\ldots, h_T\}$ trained on random subsets of the data, combined the outputs of all the trees to provide robust and accurate predictions about whether a review was a spoiler (1) or not (0), based on patterns in the embedding vectors:

$$P(spoiler) = \frac{1}{T}\sum_{t=1}^{T} h_t(V_R)$$

We chose Word2Vec because predicting whether a review is a spoiler required numerical representations of text. Word2Vec effectively transformed reviews into dense numerical vectors while preserving their semantic meaning. It also proved highly flexible, as it could generalize unseen words as long as they were part of the English language and handle unseen users by focusing on the context of their reviews. However, Word2Vec has some limitations. It ignores the order of words, which can be problematic in sentences where swapping two words significantly changes the meaning. Additionally, Word2Vec may struggle with rare or previously unseen words, requiring substantial training data to generate meaningful embeddings. Fortunately, the large size of our dataset mitigated this drawback, allowing the model to perform effectively. Our random forest classifier also exhibited a notable drawback: the default setting of 100 decision trees resulted in prolonged computational time. Through experimentation, we determined that reducing the number of trees to 40 achieved an optimal balance, as increasing beyond this threshold significantly extended runtime without yielding a substantial improvement in accuracy.

Given the sparse representation of spoilers in the dataset (Figure 1), training a Word2Vec model on a dataset predominantly composed of non-spoilers could negatively impact the model's predictive performance. This imbalance would result in the model primarily learning from non-spoiler content, thereby limiting its ability to identify spoilers effectively. To address this issue and enhance the model's detection accuracy, we created a balanced subset of the training dataset, ensuring a 50/50 distribution between spoilers and non-spoilers. This balanced approach optimizes the model's training process, allowing it to better differentiate and identify spoiler content.

3.2 Other Models

In addition to a Word2Vec model, we attempted to implement a couple other models including a most-spoiled books based model, a per-user frequency based model, a per-book spoiler frequency based model, and a hybrid model that combined Word2Vec and per-book spoiler frequency.

The first model we implemented was similar to

the baseline model used in Assignment 1. We counted the number of spoilers for each book in our training dataset and ranked the books in descending order of number of spoilers. The books that accounted for half the spoilers in the dataset were considered to be the "most spoiled" books. Reviews for a book in the most-spoiled list was predicted to be a spoiler, and any review for a book not on the list was predicted to be a non-spoiler.

When analyzing our dataset, we found that users that spoil a book tend to do so not just once but fairly consistently, so it seemed reasonable to assume that users who have spoiled books before would do so again. We decided to make a per-user spoiler frequency based model that would make predictions based on how often a user has spoiled books. We calculated how often each user in the training dataset posted a spoiler using the equation:

$$Freq(u) = \frac{\#\ of\ spoilers\ by\ u}{total\ \#\ of\ reviews\ by\ u}$$

for each user u. During prediction, we predict true if we've seen the user before and the user had a spoiler frequency over 0.5, if the user's frequency is below 0.5 or we haven't seen the user before we predicted false. We experimented with other thresholds but found no value that provided significant improvement in accuracy.

We built our per-book frequency based model similarly. Our data showed that books that were spoiled also tended to be spoiled repeatedly. In fact, there appeared to be a stronger correlation than per-user. We found the spoiler frequency using an analogous formula:

$$Freq(b) = \frac{\#\ of\ spoilers\ for\ b}{total\ \#\ of\ reviews\ for\ b}$$

for each book b. For prediction, we predicted true if the book has been seen before and has a frequency above 0.5, and false if the frequency was below or if the book hasn't been seen before.

Additionally, we attempted to combine our per-user frequency model with our Word2Vec model to achieve a more robust result. We calculated per-user frequency and trained a Word2Vec model as described above. To predict we checked if we had seen the user before and used the per-user frequency and a 0.5 threshold to predict if we had; if we hadn't, we used the Word2Vec model to predict.

4. LITERATURE

We used an existing dataset from the paper *Fine-Grained Spoiler Detection from Large-Scale Review Copra* [1]. Similar to our predictive task, this paper also attempted to predict if a review contained a spoiler or not. They used SpoilerNet, which is an attention neural network based on heuristics from their own dataset exploration. Although they measure their accuracy using the area under the ROC curve (AUC), we find that the results are simultaneously similar and different compared to our. This paper uses previous studies as their baselines, the lowest being a standard SVM [3].

The comparison of the different models cited in the paper can be observed in Figure 5.

Compared to the other models and SpoilerNet, our Word2Vec model surprisingly does very well. Although it is difficult to compare accuracies as we measured them differently, SpoilerNet achieved an AUC of 0.919, while our Word2Vec achieved an accuracy of 0.975 (see Table 2).

We believe that the difference in accuracy is not due to the models themselves, as the SVM-BOW model is similar to what we are doing but achieved a 0.838 AUC [4], but rather on how we train our model. It seems that the other five models are trained on a random training set of about 20,000 reviews. In our case, we train our model on a 50/50 split of spoiler vs non-spoiler reviews, which substantially increased our accuracy rates. If we compare our model when trained on the actual dataset, we received a result of 0.730 (see Table 1), which is worse than the SVM's result.

	Goodreads	
	AUC	AUC(d.)
SVM	0.744	0.790
+ item-spec.	0.746 ↑	0.800 ↑
+ bias	0.864 ↑	0.793 ↑
SVM-BOW	0.692	0.729
+ item-spec.	0.693 ↑	0.734 ↑
+ bias	0.838 ↑	0.742 ↑
CNN	0.777	0.825
+ item-spec.	0.783 ↑	0.827 ↑
+ bias	0.812 ↑	0.822 ↓
- word attn.	0.898 ↓	0.880 ↓
- word init.	0.900 ↓	0.880 ↓
- sent. encoder	0.790 ↓	0.836 ↓
HAN	0.901	0.884
+ item-spec.	0.906 ↑	*0.889* ↑
+ bias	0.916 ↑	0.887 ↑
SpoilerNet	*0.919*	*0.889*

Figure 5: Table 1 from [1] showing spoiler detection results on Goodreads dataset

In the past two years (2023-2024), some other methods to solve this task have been to separate reviews into even smaller sections. For example, while not book related, *Spoiler Detection as Semantic Text Matching*, proposes spoiler matching and assigns spoilers to episode numbers [5]. This is a more specific group to look at compared to the TV show in general. Perhaps, in a book related concept, future work could split spoilers based on chapters.

Additionally, MMoE is a Multi-modal Information and Domain-aware Mixture-of-Experts that attempts to predict movie spoiler reviews [6]. This model uses a mix of multiple features for their predictor such

as multiple user perspectives, reviewers' preference, text features, and genre-specific spoilers. It essentially combines all the features and models that we were examining into one cohesive model.

5. CONCLUSION

5.1 Training and Test Datasets

Looking at our dataset (Figure 1), we found that the overwhelming majority of the reviews were non-spoilers and only a small proportion were spoilers.

In order for our models to be thoroughly trained on non-spoiler data, we constructed our training dataset to ensure that 50% of our training set was spoilers and 50% was non-spoilers.

We constructed a validation set in a similar manner, however we also included a test set that was a random sample of the dataset to evaluate performance on data that was representative of the actual dataset.

We used a 50/50 training set of 40,000 entries, a validation set of 5,000 entries, random training set of 130,914 entries, and a random test set of 68,902 entries.

5.2 Results

	Baseline	**Most Spoiled Books**	**Word2Vec**	**User Freq.**	**Book Freq.**	**Hybrid**
50/50 Validation	**0.5**	**0.5000**	**0.7512**	**0.5498**	**0.514**	**0.5522**
Actual dataset	**0.9346**	**0.9345**	**0.7301**	**0.9238**	**0.917**	**0.9222**

Table 1: Training on actual dataset(10 estimators for Word2Vec)

	Baseline	**Most Spoiled Books**	**Word2Vec**	**User Freq.**	**Book Freq.**	**Hybrid**
50/50 Validation	**0.5**	**0.4996**	**0.7293**	**0.6696**	**0.5240**	**0.547**
Actual dataset	**0.9346**	**0.9341**	**0.9577**	**0.7033**	**0.6175**	**0.9405**

Table 2: Training on 50/50 train set(10 estimators for Word2Vec)

Random Forest Estimators	**10**	**20**	**30**	**40**	**50**
Word2Vec Accuracy	**0.9577**	**0.9713**	**0.9740**	**0.9749**	**0.9747**

Table 3: Changing # of estimators for random forest classifier (tested after training on 50/50 train set)

5.3 Discussion

The original dataset (Figure 1) was heavily imbalanced, with a predominance of non-spoiler reviews, resulting in a baseline accuracy of about 93% when predicting "non-spoiler" for all reviews. This imbalance artificially inflated the performance of the user and book frequency models, as cases where a user's spoiler rating exceeded 50% or a book's spoiler percentage surpassed 50% were exceedingly rare. As a result, these models would mark the majority of user's reviews and books as non-spoilers which gives the illusion of good performance on our non-spoiler dominated dataset. However, testing on a balanced validation set with 50% spoilers and 50% non-spoilers revealed the limitations of the user and book frequency models, achieving accuracies of only 51.4% and 54.98% respectively. Even on the imbalanced test set, these models only achieved accuracies of 92.4% and 91.7%, failing to surpass the baseline accuracy of 93%.

In scenarios where the training dataset was overwhelmingly composed of non-spoiler reviews, Word2Vec underperformed compared to other models. This was primarily due to its inability to train on sufficient instances of spoiler-related language, which limited its capacity to capture meaningful semantic patterns. Additionally, non-spoiler reviews exhibited greater linguistic variability compared to spoiler reviews, further complicating Word2Vec's training process. However, when trained on a balanced 50/50 training set, Word2Vec had access to enough spoiler and non-spoiler examples to learn the relationships between words indicative of spoilers and non-spoilers. This contextual understanding enabled the model to generalize effectively and accurately predict unseen reviews, highlighting its strength in capturing semantic nuances when sufficient data diversity is present.

5.4 Error Analysis

From Table 3, we observed that increasing the number of estimators from 40 to 50 resulted in a decrease in accuracy. This decline may indicate the onset of overfitting. Recognizing that overfitting occurred at 50 estimators, we chose not to further increase the number of estimators. Such an increase would likely exacerbate overfitting, reduce accuracy, and significantly

prolong runtime without delivering meaningful performance improvements.

6. REFERENCES

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

[2] Bowman, M., Debray, S. K., and Peterson, L. L. 1993. Reasoning about naming systems. *ACM Trans. Program. Lang. Syst.* 15, 5 (Nov. 1993), 795-825. DOI= http://doi.acm.org/10.1145/161468.16147.

[3] 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

[4] Armand Joulin, Edouard Grave, Piotr Bojanowski, Matthijs Douze, Herve J ´ egou, and Tomas Mikolov. ´ 2016. Fasttext.zip: Compressing text classification models. CoRR, abs/1612.03651.

[5] Tran, Ryan. Spoiler Recognition as Semantic Text Matching. University of California, San Diego, 2023.

[6] Zeng, Zinan, et al. "MMoE: Robust Spoiler Detection with Multi-modal Information and Domain-aware Mixture-of-Experts." arXiv preprint arXiv:2403.05265 (2024).