Supervised Learning for Automated Spoiler Detection in Book Reviews

Connor Frank
University of California, San Diego
Computer Science & Engineering
cfrank@ucsd.edu

ABSTRACT

To protect prospective buyers browsing user-submitted book reviews from dreaded 'spoilers' - unwanted revelations of plot development - sites like Goodreads allow users to flag and censor spoilers in their reviews, allowing new readers to view spoilers at their discretion. However, users may fail to censor their own reviews. Automatic spoiler detection and flagging serves as extra security against losing buyers to spoilage frustration. Using a dataset of spoiler-flagged book reviews from Goodreads, I create an automatic classifier which recognizes text containing spoilers in reviews using supervised machine learning. I establish baseline performance using only lexical features, then significantly improve performance with the addition of derived metadata features.

EXPLORATORY ANALYSIS

In the Goodreads dataset, each datum includes the following features: reviewID, userID, bookID, timestamp, rating (0-5), review contains spoiler (yes/no), and review text. The review text is given as a list of sentences where each sentence is flagged as containing a spoiler or not. The timespan covered by the Goodreads dataset begins on December 7, 2016 and ends November 5, 2017. There are 1,378,033 total reviews in this dataset, representing 18,892 users and 25,475 books. Among all reviews, the average rating is 3.57 out of 5 (Fig 1).

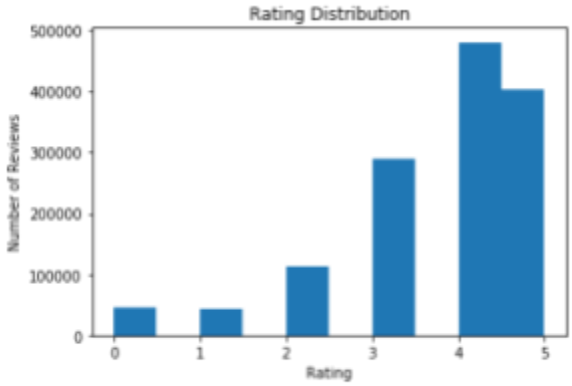

Fig. 1: Ratings distribution of all reviews

Out of all of the reviews, 89,627 reviews contain at least one spoiler (6.5% of total reviews are spoiled), representing 16,040 spoiler-users and 24,558 spoiled books. Among reviews which contain spoilers, the average rating is 3.69 out of 5 (Fig 2).

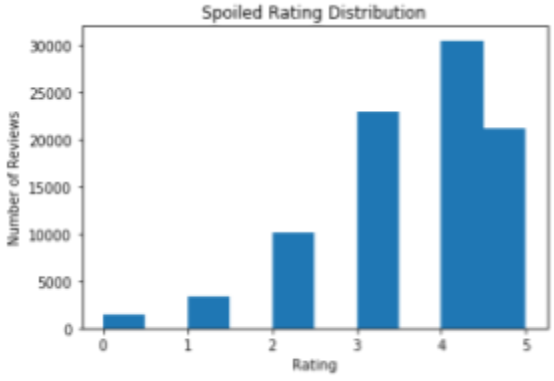

Fig. 2: Ratings distribution of only reviews containing spoilers

Across reviews containing spoilers, I calculated the "spoil percentage" - the proportion of sentences flagged as spoilers to sentences which are not spoilers - per

review. On average, 28.24% of spoiled reviews are spoiler-flagged (Fig 3).

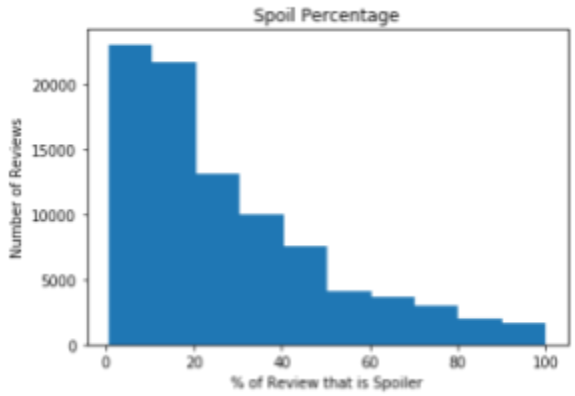

Fig. 3: Spoil percentage per spoiled review. Most reviews contain less than 20% spoiled content

PREDICTIVE TASK

The predictive task of interest on this dataset is spoiler detection - a binary classification task which labels reviews as either containing or not containing spoilers. Since most reviews contain a minority percentage of spoiler content, it makes more sense to classify individual sentences per review, rather than just predicting if a whole review contains a spoiler (a derivative metric, anyway). Moreover, atomic spoiler detection more closely matches the format of the Goodreads dataset, which flags and censors only spoiled content rather than entire reviews. Given that spoilers are lexical elements of reviews, a text-based classifier makes sense for a baseline model.

Since the ratings distributions between spoiled reviews and non-spoiled reviews are highly similar, I exclude rating as a relevant feature in my spoiler detection model. However, userID and bookID could yield relevant information given spoilage history in past reviews. For this purpose I am only interested in reviews which contain spoilers.

DATASET CONSTRUCTION

From the reviews dataset I extract only reviews which contain spoilers. I then generate a new dataset which separates reviews into individual labeled sentences. I create a new dataset of label-text pairs, where the text is a single sentence from a review and the label identifies whether it is a spoiler, plus the userID and bookID. To balance the new dataset, I populate it with an equal number of spoiler and non-spoiler text samples.

To encode user and book average spoilage information as features, first I must generate two dictionaries, *books_per_user* and *users_per_book*. For each user, I calculate their average spoil percentage among all books that user has reviewed. E.g. if a user has reviewed 10 books, and each of those reviews contained 50% spoilers, then that user has an *average spoil percentage* of 50%.

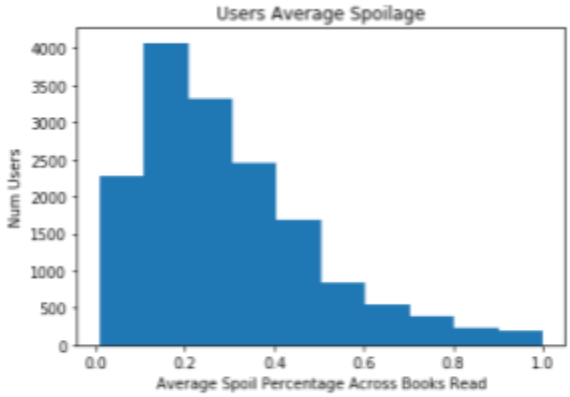

Fig. 5: users' average spoil percentage across all books read per user

The average user spoil percentage is 29.92% - that is, a typical user's reviews contain 29.92% spoilers across all books they have reviewed.

The second step is to calculate the average spoil percentage per book. For each book, I calculate the average spoil percentage

across all users that have reviewed that book. E.g. if a book has been reviewed by 10 users, and each of those users have average spoil percentages (from the first calculation) of 50%, then that book has an average spoil percentage of 50%.

Fig. 6: book average spoil percentage across all reviewing users

The average book spoil percentage is 28.24% - that is, a typical book's reviews contain 28.24% spoilers across all users that have reviewed it. Now each user and each book have an associated 'average spoilage' metric stored in a dictionary, accessible by userID or bookID.

PERFORMANCE EVALUATION

In this experiment I use 200,000 spoilers and 200,000 non-spoilers, for a total dataset size of 400,000 labeled text samples. I partition this dataset into training, validation, and test splits at 80-10-10% ratio. I use the training set for training the logistic regressor, the validation set to select a model, and the test set to test model generalization. I use the metrics of accuracy (Acc) and balanced error rate (BER) to evaluate performance during model validation and testing.

BASELINE MODEL DESIGN

I create the binary classifier for spoiler detection using logistic regression and a bag-of-words model. It stands to reason that spoilers exhibit similar lexical properties. For example, reviews containing words like 'ending,' 'secret,' or 'twist' may be more likely to spoil important plot developments. Additionally, a bag-of-words approach seems reasonable as previous experiments in spoiler detection used similar techniques (Boyd-Graber, *et. al.* 2013). Using the bag-of-words model, I created a dictionary of 2000 most common words (unigrams), removing punctuation, as a baseline. On the unigram bag-of-words model I also test stemming (using the nltk PorterStemmer library), as well as removal of stopwords. Then I test a bigram bag-of-words model, expanding the dictionary to 2000 most common unigrams + bigrams. Finally, having established baseline performance using only lexical features, I test the addition of derived metadata features: *user average spoilage* and *book average spoilage*.

RESULTS

Model	Acc (%)	BER
Unigram	65.82	-
Stopwords Removed	64.09	-
Stemmed Unigram	66.66	0.333
Bigram	64.45	0.356
Stemmed Unigram + Average Spoilage	77.63	0.224

Fig. 4: Model performance in terms of accuracy (Acc) and balanced error rate (BER)

Among models which examined only text features, the Stemmed Unigram model performed best. The stemmed unigram model slightly outperformed the unstemmed unigram model. The Bigram model and Stopwords-removed model each performed

worse than the baseline, and were discarded. The Stemmed Unigram model marks overall baseline performance at 66.66% accuracy and 0.333 BER. The addition of average spoiler metrics for users and books increases performance to 77.63% accuracy and 0.224 BER.

CONCLUSIONS

Incorporating derived metadata features substantially increased task performance. It makes sense intuitively that book average spoilage and user average spoilage are predictive features during classification. If a user has high average spoilage, then that users reviews on average contain more spoilers, so any given text written by this user is more likely to be a spoiler. Likewise, if a book has high average spoilage, its readers tend to spoil a lot in their reviews, so any text written about this book is more likely to be a spoiler. The book average spoilage metric may additionally be correlated (or even predictive) to the genre or popularity of a book. A popular mystery thriller may have more spoil-worthy and relevant content for reviewers to discuss than an obscure non-fiction book, for example. Therefore, extracting additional features such as genre and popularity of a book may further increase model performance for spoiler detection. A more robust dataset is required to test this hypothesis, though, given the Goodreads dataset used in this experiment has anonymized its books and users with ID numbers. Nevertheless, it is significant that we have shown external metadata features can be derived or approximated by manipulating data patterns and averages.

Regarding lexical features, stemming slightly improved task performance while removing stopwords and adding bigrams each slightly decreased task performance. Stemming allows the model to better capture words which may be diluted by changes in tense or use but essentially carry the same meaning, i.e. ‘end’ vs. ‘ends’ vs. ‘ending,’ so it makes sense that it improved performance. I had expected bigrams to increase performance, since n-gram models better capture contextual meaning in text which may be lost during unigram processing. One possible explanation for the bigram model’s poorer performance is that the bigram model largely increases the number of features but may only slightly increase the number of informative features.

RELATED LITERATURE

The *Goodreads* dataset was created and used in a previous experiment on fine-grained spoiler detection (Wan, *et. al* 2019.) Similar to my model’s average spoilage metrics, that experiment feature-encoded the percentage of reviews containing spoilers per item/user. Both approaches successfully captured average spoiler tendencies for books and users. They went beyond my model’s feature representation by also encoding average spoilers position within reviews, spoiler clustering tendencies, and book-specific terms characterized by document frequency and inverse item frequency (DF-IFF). The ‘SpoilerNet’ model proposed by that experiment incorporates cutting-edge machine learning techniques including SVM (BoydGraber et al., 2013; Jeon et al., 2013), SVM-BOW (Joulin et al., 2016), CNN (Kim, 2014), and HAN (Yang et al., 2016). The experiment concludes from its results that spoiler detection performance is generally improved by including item-specificity and

user/item bias features. These findings are consistent with my own conclusions, and suggest improvements for advancing my own model's performance.

References

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

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.

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, Herve J ´ egou, 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.

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