The Frankenstein Model: Spoiler-Detection using Text and Metadata Features

Mira Behar
Vassar College

Francesca Lucchetti
Vassar College

Abstract

This study tackles the Spoiler Detection Task using binary classifiers along with text and metadata features. We found that metadata did not significantly improve the performance of the classifier, while text features performed worse than the bag-of-words baseline. However, we found that restricting the dataset to specific genres improved the performance of the classifier; further adding data on the reviewer's user ID as well the location of the spoiler within the review improved the F_1 scores for the positive label.

1 Problem Statement

Our study is a replication study where we tackle the spoiler detection task: given a sentence from a review of a book, predict if it contains a spoiler. A spoiler is defined as a piece of information that reveals important plot developments which, if previously unknown, may ruin the reading experience for new readers. The most popular state-of-the-art models in spoiler detection are based on neural networks. For simplicity, in this study we utilized classifiers in lieu of neural networks. We investigated which text or metadata features may improve a bag-of-words baseline in a binary classification model.

2 Related Work

Spoiler detection has achieved great results in recent years with attention-based neural networks. Neural networks like the hierarchical attention network (HAN) (Yang et al., 2016) have had success in modeling sequential dependency among sentences in a text (Wan et al., 2019), which is useful in identifying the location of a spoiler sentence within the text. The best performing NN model is SDGNN (Chang et al., 2021), which uses syntax-aware Graph Neural Networks (GNNs) to weigh the relative importance of dependency relations between context words to fully capture semantics. This model also leverages genre-aware pooling to solve the common problem of distraction due to genre-specific revelatory words, which are erroneously flagged as spoilers (eg. "murder" in murder-mystery novels) (Chang et al., 2018).

Non-neural models have achieved moderate success in spoiler-detection with machine learning algorithms. Guo and Ramakrishnan 2010 proposed a solution which leveraged LDA (Latent Dirichlet Allocation) topic modeling alongside sequential dependency data. To capture sequential dependency without the use of attention networks, the authors used the Stanford typed dependency parser (de Marneffe et al., 2006) in the pre-processing phase and stored dependency information as bigrams. Another contribution of this model is the use of LDA-based predictive perplexity (PP) to compute the similarity of item synopsis with the spoiler text. The intuition behind this is that words which appear in the item's synopsis cannot be classified as spoilers in the candidate text.

Other machine learning models used in spoiler detection are SVMs (Support-vector machines). SVMs have generally been used to examine lexical elements of text such as unigrams, stems of unigrams or bigrams (Boyd-Graber et al., 2013). Other common features are Named Entities (NE), frequently used verbs, presence of URLs as a measure of objectivity, and the main tense of a text (Jeon et al., 2013). The problem with using lexical features is that unigrams can sometimes be distracting (for example "father" was found to be a misleading word because many spoilers are related to family member reveals (Boyd-Graber et al., 2013)). For this reason, some studies have used stepwise regression to evaluate usefulness of proposed features (Jeon et al., 2013). Other studies found that the incorporation of metadata around the item's genre, the date the item was published or reviewer data (Boyd-Graber et al., 2013; Wan et al., 2019) have been effective in improving precision.

A problem which is ubiquitously cited among spoiler detection studies is annotator error due to the subjectivity of spoilers. The existence of this bias may affect what results are achievable with non-neural models. We expect the classifiers to achieve an F_1 score of less than 0.75, as per the results in Jeon et al. 2013.

3 Methods

For this study we will use a combination of the Goodreads Reviews dataset[1], Goodreads Books and Goodreads Book Genres[2] datasets (Wan et al., 2019) containing book reviews scraped from the Goodreads website in JSON format. For each review, the Reviews dataset provides the item (book) ID, the reviewer's user ID, the review text, as well as a flag "has_spoiler" which was set by annotators for each sentence in the review. The Book and Genre datasets can be indexed using the book ID and have information on the book's genres and author ID.

Since we are combining two datasets, in our data preprocessing stage we created a CSV file to collect all our target data. Each row in the CSV corresponds to one sentence in a review, along with the review ID, reviewer ID, book ID, publication date, book author ID, book genre, reviewer's rating for the book, the sentence's spoiler label (0 or 1) as well as the location of the sentence in the review given by an index.

We made sure that the new CSV dataset was balanced. In the original Reviews dataset, spoiler sentences made up just 3% of all 14 million sentences. In our balanced CSV, we undersampled the sentences with no spoilers and created a dataset with around 1 million sentences and an even split between the positive and negative spoiler flag. To maintain balance during test/train splitting of the dataset, we used the Sklearn.metrics `train_test_split` function which balances the split and randomizes the sentences. Without a balanced dataset, F1 scores were very low (< 0.1) due to the model having an oversample of negative spoiler.

[1]https://sites.google.com/eng.ucsd.edu/ucsdbookgraph/reviews

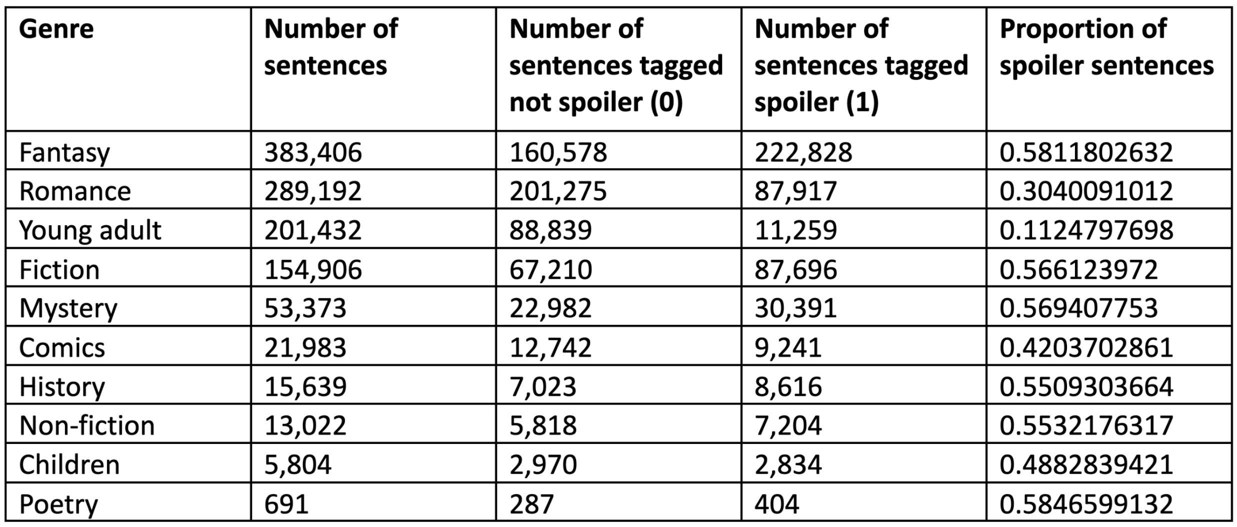
[2]https://sites.google.com/eng.ucsd.edu/ucsdbookgraph/books

Our study used three main classifiers. The first is the NLTK NaiveBayes classifier (`classify_bow_NB`) which is a bag-of-words classifier using presence of words in a document instead of counts. The BOWs that were fed to this classifier were unigrams, bigrams, and Named Entity unigrams (following the example in Jeon et al. 2013). This classifier was useful to investigate the most predictive features for our task.

The second classifier and our baseline is the Sklearn MultinomialNB classifier (`classify_bow_counts`) which is a bag-of-words classifier with counts. We also tested this method on the LinearSVC classifier, but found that it performed worse both on F_1 scores and time constraints. Along with bag-of-words unigrams, we also passed bigrams, trigrams and four-grams to this model to investigate how the F_1 scores would improve as word relation information is added to the prediction model. Another interesting application of this classifier is that it can be used to predict unseen sentences (`predict_unseen`). We supplied it with made-up sentences (eg. plausible spoilers or deceptive non-spoilers) and observed what its predictions were after training on the CSV dataset.

The third and main classifier of this study is also an Sklearn MultinomialNB classifier (`classify_many_feats`) with the difference that it uses a Scipy HStack of matrices (CSR) to combine multiple features. For example, along with bag-of-word unigrams, this classifier can be fed metadata such as the book genre or the user's ID, or the index of the sentence within the review. The intuition behind this is that certain users and genres may be prone to more spoilers, and spoilers may tend to appear toward the end of reviews.

This classifier also has the option of swapping the simple bag-of-words unigrams with other text-based features, like POS bigrams or only NE unigrams. POS bigrams are a way of approximating dependency parsing by creating bigrams containing nouns and verbs (eg."NNP-VB" $\rightarrow$ "Harry-kills"). POS bigrams can also we used with placeholder substitutions, where the proper noun is substituted with a placeholder to avoid overfitting to a specific book (eg. 'Harry-kills" $\rightarrow$ "NNP-kills"). The idea of this classifier is that we could create a Frankenstein classifier which uses all our best scoring features (text or metadata) to achieve the highest score possible.



Genre	Number of sentences	Number of sentences tagged not spoiler (0)	Number of sentences tagged spoiler (1)	Proportion of spoiler sentences
Fantasy	383,406	160,578	222,828	0.5811802632
Romance	289,192	201,275	87,917	0.3040091012
Young adult	201,432	88,839	11,259	0.1124797698
Fiction	154,906	67,210	87,696	0.566123972
Mystery	53,373	22,982	30,391	0.569407753
Comics	21,983	12,742	9,241	0.4203702861
History	15,639	7,023	8,616	0.5509303664
Non-fiction	13,022	5,818	7,204	0.5532176317
Children	5,804	2,970	2,834	0.4882839421
Poetry	691	287	404	0.5846599132

Figure 1: Statistics for the Goodreads Review Dataset

For evaluation we collected data on recall, precision and F_1 measures. We explored Area Under Curve (AUC) measure suggested by Wan et al. 2019 but found that it did not provide additional information because we used balanced dataset.

4 Results

We compared the performance of the MultinomialNB and LinearSVC classifiers when trained with bag-of-words n-grams, starting with unigrams and successively including larger n-grams (Fig. 2 and 3). The MultinomialNB classifier produced better F_1 results across all features and performed better as we added more n-gram features, but declined slightly when we added four-grams (due to four-grams becoming less informative as sentence length grows). Including trigram features when training the MultinomialNB classifier produced the best F_1 score of 0.73. The LinearSVC model F_1 results were fairly consistent at 0.70 across the tested n-gram combinations, with recall increasing slightly from 0.69 to 0.71 after including trigrams. We established the MultinomialNB model trained with unigrams as our baseline, with an F_1 score of 0.71.

The performance of the NLTK NB classifier improved slightly when using bigrams as opposed to unigrams, and decreased when only named entity unigrams were used (Fig 4). The NLTK NB classifier given only bigrams had the best performance, with a F_1 score of 0.74.

Beyond n-grams, we experimented with feeding the MultinomialNB model several different text and metadata features, and found that none of these features were able to improve upon our baseline. Of the text features, using only NE unigrams produced a poor F_1 value of 0.64, and the performance with POS bigrams was particularly bad, with F_1 scores below 0.5 (Fig 7). The metadata features were slightly more informative. UserID and Genre produced the highest F_1 scores of 0.71 when fed to the MultinomialNB classifier, but these are still lower than the baseline (Fig 5). We then tried feeding the classifier multiple metadata features, starting with the ones that performed the best when used individually and found that as we incorporated more metadata features into the model, the F_1 score declined from 0.71 to 0.54 (Fig 6).

Based on these results we designed our Frankenstein model, combining informative text and metadata features (Fig 8). We were able to improve slightly upon the baseline by feeding the MultinomialNB classifier genre metadata with n-grams, excluding 4-grams, which yielded an F_1 score of 0.72. We repeated this experiment with user ID metadata instead of genre, and obtained the same results.

To determine if thematic differences between genres would impact the way spoiler sentences were tagged, we tried running the Multinomial NB classifier on smaller datasets organized by genre. For accuracy, we used the fantasy and romance datasets, since they contained the most sentences (Fig 1). The baseline performance of the classifier on the fantasy dataset, given only unigrams, was the same as the baseline for the whole dataset, but the F_1 score increased to 0.75 when we fed the classifier the same information as our Frankenstein model, with the addition of the spoiler sentence location metadata (Fig 10). The baseline classifier for the romance dataset achieved the same F_1 score, and the improved model for the romance genre performed even better, with an F_1 score of 0.77 (Fig 9).

We further tested our Frankenstein trigram and user ID model on unseen input consisting of made-up sentences.

Predicted 1 (Spoilers):

```
(1) H kills V at the end.
(2) I never expected it to end
with a big wedding...
(3) ajvhsdbnmu Wooooooow fs qon
vjk vkuhg dkj!!!!
(4) I don't believe it!!
(5) Really I am trying to trick
you, why did it have to happen
```

Predicted 0 (Not spoilers):

```
(6) What a cliffhanger!
(7) ajvhsdbnmu fs qon vjk vkuhg
dkj
(8) Another one for the hell of
it.
(9) This isn't a spoiler.
```

The model successfully predicts sentences 6 and 9 as not-spoilers despite the trigger words "spoiler" and "cliffhanger". The model however seems to be misled by exclamations like "Woooooow" (sentence 3) which when removed solves the mislabeling (sentence 7). Since the Goodreads Reviews Dataset contains reviews from an online source, it makes sense why the model would be biased toward exclamatory words like the aforementioned "Wow".

We also found that the Naive Bayes without counts model was able to pick up Goodreads-specific information in its most predictive features. For example, among the top 30 most predictive bigrams were the following words: "ow-om", "h-was", "h-is". This may seem like nonsense at first glance, but actually "ow-om" or "other-women/other-men" is a Goodreads specific category in the Romance genre which is prone to spoilers because of its subject (adultery romance books). Similarly the model is picking up on a Goodreads slang for hero/heroine, "h".

By training the MultinomialNB classifier with a limited number of metadata and text features, we were able to create models that were more successful at predicting spoiler sentences than our baseline, especially when we narrowed our dataset to book reviews from specific genres. However, even our best results are not comparable to the performance of neural network models for spoiler detection, demonstrating that this spoiler detection task is difficult to implement with simplistic classifiers. The spoiler detection task is especially tricky because of the subjectivity of the annotations. Furthermore it is hard to guess which features will be predictive of spoilers: genre ID and user ID were our best guesses and did improve scores, but similar metadata like author ID performed badly. Likewise, POS bigrams did not perform as well as we thought. This is because it is hard to know which POS will be indicative of spoilers. We selected noun-verb pairings (Guo and Ramakrishnan, 2010) because these could indicate the presence of character names and their actions, but there could be other better predictors.

5 Limitations and Future Work

More tests need to be run in order to understand what criteria are a significant improvement upon the bag-of-words baseline, especially considering the randomization of the test/train set and the number of possibilities in combining all of our various features.

While working on this project we made plans for certain features which we could not implement since they were beyond the scope of the project. For future implementations, it would be interesting to observe the effect of a DF-IFF transformer (Wan et al., 2019); DF-IIF is a measure of the importance of a word for a specific item (book). By using DF-IIF a model could score which words are significant predictors for certain books and which are not (eg. "murder" is a good spoiler predictor for a fantasy book, but not a mystery). This feature was not implemented because of how computationally expensive it is. Similarly, instead of using POS bigrams, which did not perform as well as bag-of-words, it may be worthwhile to implement dependency parsing with tools like the Stanford Dependency Parser. Again, we chose to approximate word dependency through POS tagging because of computational limitations.

References

Jordan Boyd-Graber, Kimberly Glasgow, and Jackie Sauter Zajac. 2013. Spoiler alert: Machine learning approaches to detect social media posts with revelatory information. *Proceedings of the American Society for Information Science and Technology*, 50(1):1–9.

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 *Advances in Knowledge Discovery and Data Mining*, pages 183–195, Cham. Springer International Publishing.

Buru Chang, Inggeol Lee, Hyunjae Kim, and Jaewoo Kang. 2021. "killing me" is not a spoiler: Spoiler detection model using graph neural networks with dependency relation-aware attention mechanism. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 3613–3617, Online. Association for Computational Linguistics.

Sheng Guo and Naren Ramakrishnan. 2010. Finding the storyteller: automatic spoiler tagging using linguistic cues. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 412–420.

Sungho Jeon, Sungchul Kim, and Hwanjo Yu. 2013. Don't be spoiled by your friends: spoiler detection in tv program tweets. In *Proceedings of the International AAAI Conference on Web and Social Media*, volume 7, pages 681–684.

Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC'06)*, Genoa, Italy. European Language Resources Association (ELRA).

Mengting Wan, Rishabh Misra, Ndapa Nakashole, and Julian McAuley. 2019. Fine-grained spoiler detection from large-scale review corpora. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2605–2610, Florence, Italy. Association for Computational Linguistics.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489, San Diego, California. Association for Computational Linguistics.

Entered Feature(s)	Precision	Recall	F1 Score
Unigrams	0.70	0.72	0.71
+Bigrams	0.71	0.73	0.72
+Trigrams	0.72	0.74	0.73
+4-grams	0.72	0.73	0.72

Figure 2: Label-1 scores for baseline model. Multinomial Naive Bayes with N-grams.

Entered Feature(s)	Precision	Recall	F1 Score
Unigrams	0.71	0.69	0.70
+Bigrams	0.70	0.69	0.69
+Trigrams	0.70	0.71	0.70
+4-grams	0.70	0.71	0.70

Figure 3: Label-1 scores for baseline model. Support Vector Classification with N-grams.

Entered Feature	Precision	Recall	F1 Score
Unigrams	0.6242481526	0.8870883338	0.7328126576
Bigrams	0.6586445589	0.859017426	0.7456036446
Named entity (NE) unigrams	0.6083869795	0.8268216535	0.7009817483

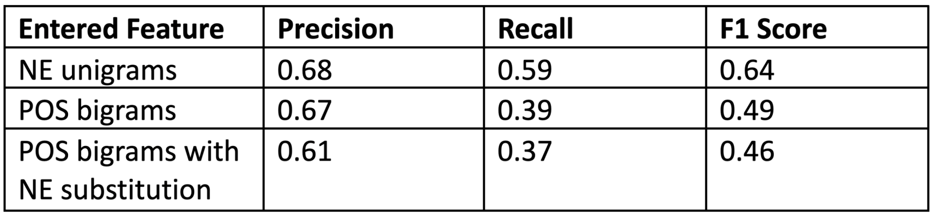
Figure 4: Label-1 scores for Naive Bayes Model without counts.

Entered Feature	Precision	Recall	F1 Score
User ID	0.70	0.71	0.71
Author ID	0.63	0.63	0.63
Genre	0.71	0.71	0.71
Rating	0.71	0.67	0.69
Book ID	0.68	0.60	0.63
Spoiler sentence location	0.68	0.69	0.68

Figure 5: Label-1 scores for Multinomial Naive Bayes Model with individual metadata features.

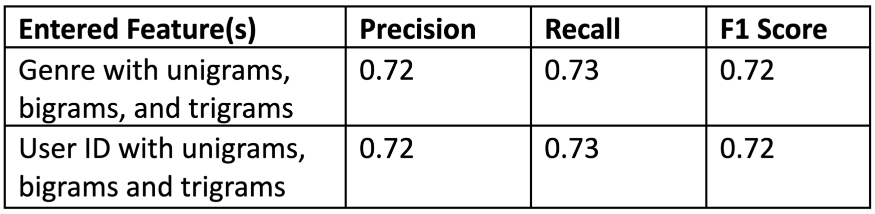
Entered Feature(s)	Precision	Recall	F1 Score
Genre	0.71	0.71	0.71
+User ID	0.70	0.70	0.70
+Rating	0.71	0.66	0.68
+Spoiler sentence location	0.70	0.66	0.68
+Book ID	0.66	0.60	0.63
+Author ID	0.56	0.53	0.54

Figure 6: Label-1 scores for Multinomial Naive Bayes Model with combined metadata features.



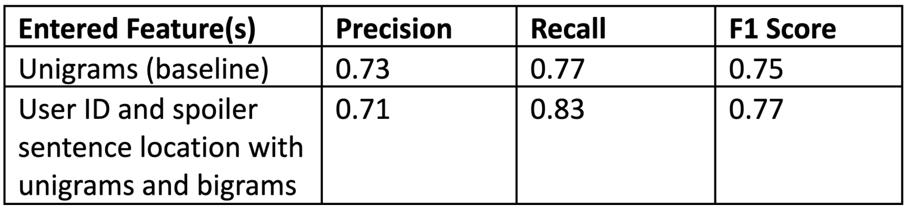
Entered Feature	Precision	Recall	F1 Score
NE unigrams	0.68	0.59	0.64
POS bigrams	0.67	0.39	0.49
POS bigrams with NE substitution	0.61	0.37	0.46

Figure 7: Label-1 scores for Multinomial Naive Bayes Model with text features.



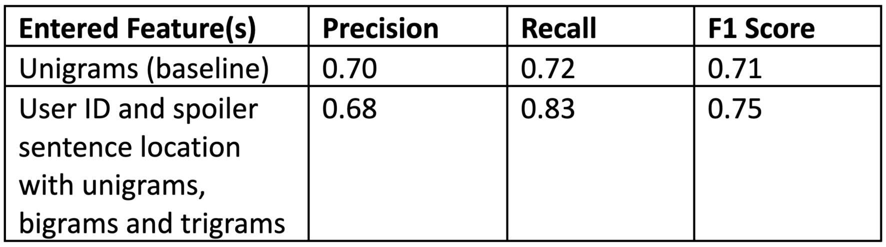
Entered Feature(s)	Precision	Recall	F1 Score
Genre with unigrams, bigrams, and trigrams	0.72	0.73	0.72
User ID with unigrams, bigrams and trigrams	0.72	0.73	0.72

Figure 8: Label-1 scores for best model, Frankenstein, using Genre and User ID metadata with trigrams.



Entered Feature(s)	Precision	Recall	F1 Score
Unigrams (baseline)	0.73	0.77	0.75
User ID and spoiler sentence location with unigrams and bigrams	0.71	0.83	0.77

Figure 9: Label-1 scores for best model, Frankenstein, on Romance genre dataset.



Entered Feature(s)	Precision	Recall	F1 Score
Unigrams (baseline)	0.70	0.72	0.71
User ID and spoiler sentence location with unigrams, bigrams and trigrams	0.68	0.83	0.75

Figure 10: Label-1 scores for best model, Frankenstein, on Fantasy genre dataset.