|  |
| --- |
| **Machine Learning Approaches to Detect Spoiler Reviews** |
|  |
| **Jue Wang**  Northwestern University |
| juewang2020@u.northwestern.edu |
| Git Repo: <https://github.com/im-daniel-wang/Spoiler_Detection> |
|  |
|  |

Abstract

|  |
| --- |
| 000  001  002  003  004  005  006  007  008  009  010  011  012  013  014  015  016  017  018  019  020  021  022  023  024  025  026  027  028  029  030  031  032  033  034  035  036  037  038  039  040  041  042  043  044  045  046  047  048  049 |

Spoilers are important plot information of books/movies that can ruin user experience on review websites when they are looking for plot summary or user reviews/ratings. This project aims to predict sentence-level spoiler probability based on machine learning models and compare performance metrics among different models including Support Vector Machine (SVM), Logistic Regression, and Convolutional Neural Networks (CNN).

Introduction

Book content and reviews websites such as *goodreads.com* provide users with ratings and reviews. The reviews on the website can sometimes contain ‘spoilers’, which are review or summary texts that reveal elements of the plots and thus have the potential to kill the joy of audience who have not seen the book. Major movie/book content websites have put out guidelines that discourage spoiler contents, but more work can be done to ‘mask’ the content that have a high probability of containing spoilers, such that users can have a more confident browsing experience on the website. This project is designed to predict spoiler content using machine learning models in order to achieve this goal.

1. Related Work

There have been numerous approaches to text classification using machine learning and deep learning models. These models have been used extensively for sentiment analysis, classifying news stories into categories, etc.

Support Vector Machine (SVM) was introduced by V. Vapnik et al., and in 1998 was recognized by Joachims to be especially suited for text categorization tasks for its ability to deal with high dimensional input space and sparse document vectors. It was also pointed out that most text categorization tasks are linearly separable, and SVM performs well in these tasks since its idea is to find such linear separators between the classes.

|  |
| --- |
| 050  051  052  053  054  055  056  057  058  059  060  061  062  063  064  065  066  067  068  069  070  071  072  073  074  075  076  077  078  079  080  081  082  083  084  085  086  087  088  089  090  091  092  093  094  095  096  097  098  099 |

Neural network approaches including CNN-based (Kim, 2014) and RNN-based (Yang, 2016) have also achieved success on sentence classification.

Although text classification has seen substantial progress throughout the years, spoiler detection is still rather unexplored. Only a few labeled datasets containing sentence-level label for spoilers are publicly available online. Previous attempts include leveraging user/item spoiler bias to create additional features, using different models to account for different sentence semantics across genre contexts for better generalizations. Hierarchical attention network (HAN) was recently applied on the task to account for the dependency among the sentences in a review, since human pay different attention to different parts of a review (Wan, et al., 2019)

Dataset

The dataset used is from UCSD Book Graph’s Goodreads Datasets which were collected in late 2017’s from *goodreads.com*. This project uses the spoiler subset where each book/user has at least one associated spoiler review. The corpus contains 1,378,033 review documents and 17,672,655 sentences, out of which 3.1% are sentences that have spoilers. For each review, information on the user identifier, time of review, each sentence in the review along with the label, and the book identifier is provided. The labels for each sentence in each review are the response variable used in this project.

Chart

Description automatically generated Chart, line chart

Description automatically generated

|  |
| --- |
| 150  151  152  153  154  155  156  157  158  159  160  161  162  163  164  165  166  167  168  169  170  171  172  173  174  175  176  177  178  179  180  181  182  183  184  185  186  187  188  189  190  191  192  193  194  195  196  197  198  199 |

1. (b)

Chart, bar chart

Description automatically generatedChart, bar chart

Description automatically generated

1. (d)

Figure 1: (a) Average number of words per review sentence for spoiler/non-spoiler reviews (b) Density plot of proportion of spoiler sentences in a review (c) Proportion of occurrences of “die” in spoiler and non-spoiler review corpus (d) Proportion of occurrences of “survive” in spoiler and non-spoiler review corpus

From the visualizations (a) to (d) it can be seen that spoiler reviews tend to have more slightly more sentences than non-spoiler reviews. The two classes of texts have drastically different characteristics in terms of the occurrences of words such as “die” and “survive” that are closely related to the plot information. It is then worthwhile to discover manually extracted signals of metadata from the sentences.

To address the problem of imbalanced classes, the spoiler sentences are kept, and the non-spoiler sentences are down sampled to have equal amount of data (both containing 569,724 rows) during model training.

Method

Two options are available since we can both choose to predict whether a review will contain at least one spoiler sentence or predict whether a sentence is a spoiler. The latter one was chosen to be more appropriate since sentence-level data contains less noise, and when we deploy the model in production, we ideally want to mask the exact sentences that are spoilers. Some experiments were done on review-level classification, but they yielded very low recall score.

* 1. Data Cleaning and Feature Extraction

Data cleaning is performed prior to model fitting. All texts are preprocessed with tokenization, normalization, stop-words removal and lemmatization.

Features of text are extracted and review sentences are converted to TF-IDF scoring matrices which reflect the importance words in a review given the corpus context. The TF-IDF scores are used as predictors for the classical machine learning models (logistic regression and SVM), and both 1-gram and 2-gram bag-of-words (BOW) are used and their performance metrics compared to see which one gives the highest predictability.

* 1. Logistic Regression

Logistic regression is used as a benchmark model for the text classification problem. It’s parametric nature allow easy interpretation of model result and feature importance.

|  |
| --- |
| 100  101  102  103  104  105  106  107  108  109  110  111  112  113  114  115  116  117  118  119  120  121  122  123  124  125  126  127  128  129  130  131  132  133  134  135  136  137  138  139  140  141  142  143  144  145  146  147  148  149 |

Some hyperparameter tuning is performed to decide the best combination of parameters that gives the best prediction outcome. The cost parameter C, as well as 1-gram vs. 2-gram in TF-IDF vectorization are tuned. C is the inverse of regularization strenght, and smaller values indicate stronger regularization. Using bi-gram in TF-IDF preserves more temporal traits of the review sentence, and it is thus interesting to see whether including more temporal information would improve the prediction performance.

|  |
| --- |
| 150  151  152  153  154  155  156  157  158  159  0  161  162  163  164  165  166  167  168  169  170  171  172  173  174  175  176  177  178  179  180  181  182  183  184  185  186  187  188  189  190  191  192  193  194  195  196  197  198  199 |

|  |  |  |
| --- | --- | --- |
|  | F1 | AUC |
| bi-gram, C=5, l1 | 0.6939 | 0.6888 |
| bi-gram, C=15, l1 | 0.6795 | 0.6751 |
| bi-gram, C=5, l2 | 0.7060 | 0.6998 |
| bi-gram, C=15, l2 | 0.6962 | 0.6909 |

The experiments show that logistic regression using bi-gram BOW, regularization parameter C=5, l2-regularization performs best with the highest F1 score and AUC score.

* 1. Support Vector Machine (SVM)

Support vector machine (SVM) is then used for its advantages in text classification. More specifically, texts of different classes are often linear separable, and text contains a lot of features. Using linear kernel in SVM is suited for this type of task since it is fast and mapping the data to a higher dimensional space does not really improve performance. Hyperparameter tuning is done to determine the best combination of 1-gram/2-gram and regularization parameter c. F1 and AUC scores are recorded for each experiment.

|  |  |  |
| --- | --- | --- |
|  | F1 | AUC |
| 1-gram, C=1.0 | 0.7023 | 0.6936 |
| 1-gram, C=15.0 | 0.6932 | 0.6881 |
| 2-gram, C=1.0 | 0.7074 | 0.6999 |
| 2-gram, C=15.0 | 0.6650 | 0.6615 |

Using 2-gram BOW with C=1.0 regularization parameter works best for SVM according to the experiments. The performances across different combinations are similar to the ones produced by logistic regression without significant lifts. It seems that using bi-gram features allow the system to discover more temporal structure of the dataset and discover instances of absolute markers like “the end” and “end of” (Jordan et al, 2013).

* 1. Derived Features for Spoiler Detection

Apart from lexical features generated using TF-IDF, additional metadata of each review sentence can potentially contribute to classifying whether a sentence is a spoiler or not. In this section, we describe features that are useful for this specific classification task.

**Relative Position of Sentence in Review**

With some exploratory data analysis, we found that spoilers tend to occur in the latter half of the review text, meaning that the position of the sentence within the review text has a strong implication of whether the sentence contains spoilers. To account for such behavior, for each sentence the position of the sentence in percentile is calculated. For each sentence:

|  |
| --- |
| 250  251  252  253  254  255  256  257  258  259  260  261  262  263  264  265  266  267  268  269  270  271  272  273  274  275  276  277  278  279  280  281  282  283  284  285  286  287  288  289  290  291  292  293  294  295  296  297  298  299 |

|  |
| --- |
| 200  201  202  203  204  205  206  207  208  209  210  211  212  213  214  215  216  217  218  219  220  221  222  223  224  225  226  227  228  229  230  231  232  233  234  235  236  237  238  239  240  241  242  243  244  245  246  247  248  249 |

*Percentile = Sentence Index / # Sentences*

These two features are incorporated into the modelling process in addition to the TF-IDF scores and performance metrics are compared to see whether they introduce lift. We select the best-performing logistic regression and SVM model to do the comparison. From the below chart, red represent the density of percentile indices for spoiler sentences, while blue represents the same for non-spoiler. The two densities are clearly different and worth investigating.

Chart, histogram

Description automatically generated

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | TF-IDF | | Additional Feature | |
|  | F1 | AUC | F1 | AUC |
| LR | 0.7023 | 0.6936 | **0.7214** | **0.7149** |
| SVM | 0.7074 | 0.6999 | **0.7243** | **0.7158** |

The above chart shows that with the additional feature of index percentile, both SVM and LR model saw significant improvement in F1 and AUC scores. The additional feature does provide additional predictability of spoilers.

* 1. Convolutional Neural Network (CNN)

It was shown by Kim that a simple CNN with little hyperparameter tuning and static vectors achieves excellent results on multiple benchmarks (Kim, 2014). Convolutional neural networks (CNN) utilize layers with convolving filters that are applied to local features (LeCun et al., 1998).

For this project only a subset of 150,000 samples is used to train the neural network. The network is trained in mini batches of 150 and 10 epochs. Convolution filter of size 10\*10 is used to generate features from the texts, and dropout layer is used for regularization and reducing overfitting. 10-fold cross-validation is used to compute the F1 and AUC scores. Relu and Sigmoid activations are tried and their respective performance metrics compared.

|  |  |  |
| --- | --- | --- |
|  | F1 | AUC |
| Relu | 0.6504 | 0.8305 |
| Sigmoid | 0.6431 | 0.7849 |

The experiments show that ReLu activation is more suited for this task with higher F1 score and AUC score. The property of ReLU is better suited for this task possibly due to its mechanism to combat exploding/vanishing gradient. The CNN approach achieves higher AUC score at the expense of lower F1 score. This could imply that the model does well in terms of ranking the data based on classes, but still has problem of a low recall value, indicating that a lot of the spoiler sentences failed to get recognized.

Results

After each experiment for each model specifications, F1 and AUC scores are recorded and compared in the table below.

|  |  |  |
| --- | --- | --- |
|  | F1 | AUC |
| Logistic Regression | 0.7023 | 0.6936 |
| SVM | 0.7074 | 0.6999 |
| CNN | 0.6678 | 0.8312 |
| SVM with Rank Feature | 0.7243 | 0.7158 |

In terms of F1 score, SVM with added rank feature performs best with F1 = 0.7243. In terms of AUC score, convolutional neural network (CNN) performs best with AUC = 0.7158.

SVM with Rank Feature is selected to be productized for its balanced performance across F1 and AUC score. Since CNN is trained on a small subset of data and may have overfitting, we decided to not use it although it did achieve a higher AUC score. CNN also has a significantly lower F1 score which indicate that the classification threshold might not be well-chosen.

**Example Output**

Graphical user interface, application

Description automatically generated

Discussion

The CNN approach can be further fine-tuned by adding layers that contain specific lexical information. Due to run-time constraint only a small subset of data is used to train the network for 10 epochs, possibly leading to some overfitting. As a next step, it is interesting to see the performance of a fully trained CNN.

For the classical machine learning models, more feature engineering can be done to extract more metadata from the review sentences. The main tense of a sentence can be hugely important since past tensed sentence have a higher likelihood of containing information from the plot (as an example, “He died.” is more likely to be a spoiler than “I hope he will live.”) The tense detector is computationally expensive and was hence not implemented in this paper but is worth investigating in the future.

|  |
| --- |
| 300  301  302  303  304  305  306  307  308  309  310  311  312  313  314  315  316  317  318  319  320  321  322  323  324  325  326  327  328  329  330  331  332  333  334  335  336  337  338  339  340  341  342  343  344  345  346  347  348  349 |

References

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.

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.

Mengting Wan, Rishabh Misra, Ndapa Nakashole, and Julian McAuley. 2019. Fine-Grained Spoiler Detection from Large-Scale Review Corpora. In ACL

Thorsten Joachims. 1998. Text Categorization with Supp ort Vector Machines Learning with Many Features.