Predicting Age-Appropriateness of Fan Fiction Short Stories - Winter 2019

Oishi Banerjee, Minh Nguyen, Christine Phan
Stanford University
`oishib@stanford.edu, minh084@stanford.edu, cxphan@stanford.edu`

Abstract

We use RNN and CNN models for a Natural Language Processing (NLP) supervised learning task, predicting the age-appropriateness of fan fiction stories by using author-given labels as the outcome of interest.

1 Introduction

Fan fiction (unofficial written adaptations of popular media) is a large but understudied field. While writers typically label their fan fiction for sensitive content (i.e. drug use, violence, sexually explicit content), not all stories are fully labeled. Audience classification of a fiction work allows readers to know the targeted audience of the stories so that they can decide what to read or not.

Our goal is to build a model that will label stories by age-appropriateness in order to help readers choose appropriate reading material. We used fan fiction stories from the website *archiveofourown.org*. We scraped short stories of 100-200 words, along with their summaries and metadata, including their author-given age-appropriateness tags, and processed them with GloVe embeddings. Multiple architectures were explored, including a baseline logistic regression model, RNN and CNN models using story text and metadata to predict the author-given age-appropriateness as outcome labels.

2 Related work

- **Word embedding** is a method of representing words that allows words with similar meaning to have a similar representation. Words are represented by vectors that capture information in a context for used in machine learning tasks such as improving performance of text classifiers. Two most popular methods are Word2Vec (Mikolov et al., 2013) and GloVe (Pennington et al., 2014). A recently released tool is fastText, a library created by Facebook research, aims to do efficient learning of word representations and sentence classification, keeping only important features in the trained model and performing quantization of the weight matrices with hashing (Joulin 2016).
- **Recurrent Neural Networks (RNN)** is probably the most popular architecture used in Natural Language Processsing (NLP) tasks such as text classification (Zhou, 2016). A variant of RNN, the Long Short Term Memory networks (LSTM) was first introduced by Hochreiter & Schmidhuber in 1997. It is capable of handling sequences of any length and learning long-term dependencies while avoiding the problem of vanishing gradient (Graves, 2012). Other variants of RNN have been proposed in the literature such as Gated Recurrent Units (GRU) (Cho, 2014 and Chung 2014), LSTM uniform-layer, coupled-layer, and shared-layer (bidirectional) for text classification with multi-task learning (Liu, 2016). Tai et al. 2015 introduced the tree-structured LSTM networks.

- **Convolutional Neural Network (CNN)**: Although commonly used in computer vision, CNNs, mostly with 1D convolution coupled with pooling layers have become another mainstream architecture for NLP. It has been shown to be effective for various NLP tasks with promising results in text classification. A CNN was applied on top of pre-trained word vectors for sentence-level classification was shown to have great results (Kim, 2014). Kalchbrenner et al., 2014 introduced the dynamic CNN (DCNN) for modeling sentences. Wang 2016 proposed a unified framework to expand short texts based on word embedding clustering and CNN. Conneau 2017 used up to 29 convolution layers in their Very Deep CNN architecture.
- **Others:** An attention-based bidirectional LSTM model to automatically select influential features was introduced by Zhou in 2016, and a hierarchical network with word and sentence attention mechanisms for document classification was introduced by Yang et al. in 2016.

These strategies have also been used in combination with one another. From these general concepts in related work, text classification fields have developed further: Many combine both RNN and CNN architectures. For examples, Wang 2015 presented a jointed CNN and RNN architecture that takes the local features extracted by CNN as input to RNN for sentiment analysis of short texts. Zhou 2015 proposed a novel and unified model called C-LSTM whereas Lai 2015 and Lee 2016 proposed recurrent CNN or LSTM-CNN models.

Many of these models are considered state of the art models. The Tree-LSTM: a generalization of LSTMs to tree structured network topologies, introduced by Tai et al., 2015, where each LSTM unit gains information from its children units, seems to be the most clever. Recently, Google BERT (Bidirectional Encoder Representations from Transformers) has gained a lot of attention, claiming to have an F1 score of 93.2%, surpassing the previous state-of-the-art score of 91.6% and human-level score of 91.2% (Devlin, 2018).

3 Dataset and Features

There are 13,242 total stories, scraped from archiveofourown.org and split almost evenly among the four categories. When collecting data, we stored story texts, summaries, and tags (such as the author-given age-appropriateness label), as well as the number of likes or "kudos" a story obtained. The data is quite balanced with 3142 - 3496 stories for each of 4 classes. The only non-text (numerical) feature to be explored is kudos. Kudos range from 0 - 745, with an average of 10 and median of 20.

Here is an example data point:

Text: Giddy with excitement, Stiles climbed up onto a table in the middle of the cafeteria and shouted, “Can I have your attention please?!” When the cafeteria was quiet except for a few mutterings, Stiles shouted again, “I just wanted to announce that Derek Hale, captain of the basketball team, one of the most popular dudes at school, and the hottest guy ever in the history of guys, has asked me out! And I have accepted!”

Summary: Stiles makes an important announcement.

Tags: No Archive Warnings Apply, M/M, Teen Wolf (TV), Derek Hale/Stiles Stilinski, Stiles Stilinski, Derek Hale, Alternate Universe - High School, Fluff, Announcements, Jock Derek Hale, Drabble

Kudos: 116

We used a 80-10-10 split to divide our data, with shuffling, into the training, validation and test sets. This split is typically acceptable given the size of our data. We also explore a 70-15-15 split, but it did not work as well as the 80-10-10 split. We tokenized and pad story texts, summaries and tags and processed them with 100d Glove embedding.

4 Methods

Summary: First, all story text samples are converted into sequences of word indices. An embedding matrix, which contains at index i the embedding vector for the word of index i, was done using GloVe. Different neural network layers are then built on top of it, and end with softmax output layer over 4 outcome categories. The models explored were a logistic regression, a convolutional neural network, a recurrent neural network with various features for each story.

4.1 Word Embedding

All models used Global Vectors for Word Representation (GloVe) on story text to obtain vector representations for words. The pre-train 6B vector 100-dimension was downloaded from *https://nlp.stanford.edu/projects/glove/*. Individual words are represented as real-valued vectors via word embedding, and these embedding vectors are then used in the neural network models. When adding additional features such as summary, using embedding based on both story text and summary corpus decreased the model performance greatly. This can be due to many non-sensible characters in summaries that are not meaningful, written intentionally by the authors. As a result, the embedding for all models was done based on story text vocabulary only.

4.2 Models

Baseline Model: Logistic Regression: was chosen because it appears to be the most popular and standard baseline model. It yielded the lowest accuracy of 0.27. whereas the other 3 models yielded above 0.45 in accuracy prior to tuning. The logistic regression was done with one fully connected layer perceptron using a linear activation and an output softmax layer for multi-class classification.

R-CNN: The combined R-CNN model that we adapted from a paper did not perform better than our CNN and RNN models, so it was abandoned[1] (Wang, 2016)

Convolutional Neural Network: The CNN model: consists of 3 of the 1D convolutional layers each followed by a max pooling layer. These 2 operations are sometimes thought as feature extraction of the input. The output was then flattened and connected to a fully connected (FC) layer. We also tried adding a LSTM layer prior to FC, but it did not help with the performance. We used 256 filters of size 5, pooling windows of size 2, stride of 1, valid padding, Xavier normal initializer, ReLu activation, and softmax for multi-class classification.
The CNN is somewhat easier to train and have fewer parameters than FC networks with the same number of hidden units.

Recurrent Neural Network Although our CNN performs consistently equivalent with the RNN, we decided to choose RNN as the main architecture for further modeling with added features because our main input is text, which has a sequential structure with variable length. From this decision, we developed the LSTM-RNN models.
The LSTM-RNN model: operates over a sequence of inputs and produce an output for each step, enables the propagation of context information through the sequence and captures long-term dependencies. Our model consists of a bidirectional LSTM layer of story text. We used the default choices of tanh activation, hard sigmoid as the recurrent activation, Xavier normal for kernal initializer, and orthogonal for recurrent initializer.

4.3 Variations on LSTM-RNN Features

We also tested some variants of the model by adding one extra input (summary, tags, kudos) at a time and then combining some of them to assess how these features either individually or together affect the model performance besides the main input of story text. When adding summary or tags as additional inputs, the outputs from each bidirectional LSTM layer were concatenated prior to the FC layer. We used 256 hidden units, ReLu activation, and softmax for multi-class classification. A drop-out rate of 0.5 was used for regularization as our training accuracy can be very high.

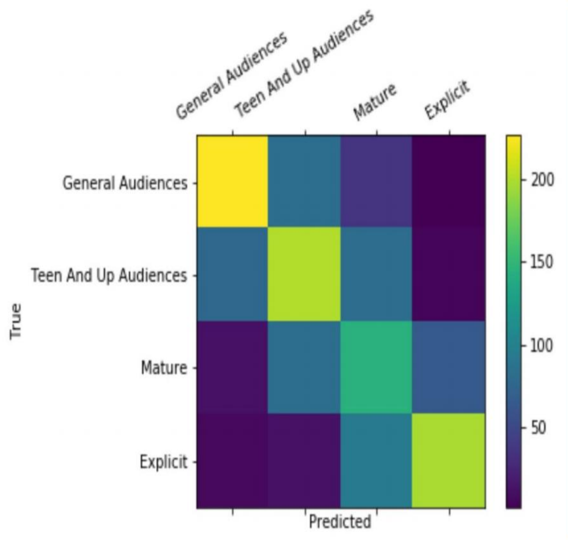
Our parameter choices were driven by literature search and model performance after a few runs on a smaller subset of the data. The standard categorical cross entropy loss function was used for 4-class classification task, although we also attempted a weighted version of this loss function as well. Model building was done using Keras.

[1]https://github.com/ultimate010/crnn/blob/master/sst2_cnn_rnn_kera1.py

5 Experiments/Results/Discussion

Hyperparameter Tuning: Aim to improve performance of the models, we attempted to adjust the number of epochs up to 100. However, our validation set accuracy started to go down after about 6-7 epochs. With batch size, we tested common size of 32, 62, 128 batches while considering the size of our data. Although none of these choices always does better than others, models with a batch size of 128 seem to have the most consistent results. We also tried to lower the learning rate from a default of 0.001 to 0.0001 with the intention to improve reliability of the models while trading off running time. A decay equals to learning-rate/sqrt(n-epochs) was also added to the optimizer as suggested in the Adam paper. However, it was suggested that the benefits of decaying the learning rate can be achieved by instead increasing the batch size during training (Smith, 2018).
Again, all these adjustments did not lead to large variations in the result within each model. However, we noticed that the CNN performs generally better with the customized Adam optimizer, whereas the RNN performs better with the default setting and Adam optimizer. Our final best performing model used a batch size of 128, learning rate of 0.001, and decay of 0 using 7 epochs.
For all classification models, the categorical cross entropy loss for multiple classes was used. Since our models did better with 2 end classes (General Audiences and Explicit), but more ambiguous with the 2 middle classes (Teen & Up and Mature), we also attempted a weighted loss function which penalizes for the class with lower accuracy. We implement the weights of [0.2, 0.3, 0.3, 0.2] for 4 classes, and this did not help with our model performance.

Results from Classification Task: With multiple classes, accuracy is used as the main evaluation metrics when comparing across different models. With the best performing model based on accuracy, we looked further into other evaluation metrics such as precision, recall, and F1-score. Below is our report for accuracy for the CNN and some RNNs. The baseline model logistic regression only achieved a 0.27 accuracy, which is slightly better than a random guess. Although we did not implement a more complex state-of-the-art models, reports from papers have shown achievements of high accuracy above 90% for text classification, and an F1 score of 93.2% for BERT. The confusion matrix and detailed evaluation metrics of the best performing model was reported, with precision, recall, and F1-score. The model does worse overall in the mature category. It is understandable the most difficult class to predict because often, it is not a clear cut between mature and explicit.



EVALUATION METRICS FOR THE BEST PERFORMING MODEL

(LSTM-RNN with story text and tags)

Class	Accuracy	Precision	Recall	F1-score
General Audience	0.66	0.74	0.64	0.69
Teen and Up	0.55	0.70	0.66	0.68
Mature	0.48	0.41	0.48	0.44
Explicit	0.64	0.53	0.55	0.54
Avg		**0.60**	**0.58**	**0.59**

Model	Accuracy	Model	Accuracy
CNN w/ text	0.51	RNN w/ text & kudos	0.50
RNN w/ text	0.49	RNN w/ text, summary, tags	0.57
RNN w/ text & summary	0.53	RNN w/ text, tags, kudos	0.57
RNN w/ text & tags	**0.59**	RNN w/ text, summary, tags, kudos	0.57

Discussion: Although our model outperformed the baseline (logistic regression), the result is not very encouraging. However, considering the consistency across 3 architectures that we implemented (CNN, RNN, and combined R-CNN), we have some confidence in our model performance.
Even after extensive tuning and adjusting some architectures, our models still have much higher training accuracy compared to validation and test set accuracy as the number of epochs increases. Due to over-fitting on the training data, the model is not able to generalize well on the validation and test sets. We worked with 2 different splits: 80-10-10 and 70-15-15. The latter of which did not do better than the earlier split. We used a high drop-out rate of 0.5 to add regularization.

Error Analysis: An extensive detailed error analysis was conducted to determine potential issues in the data and to gain more insights on why our models did not perform very well. Upon further examination of our model's predictions, we found some common sources of errors and problems.

- Author-given labels were not always to the same standard and therefore difficult to match. In one example, an author labeled a story with explicit sexual material "Teen and Up Audiences"; our model predicted a higher rating.
- In some instances, our model misunderstood the subtext or context of a story. For example, one story repeatedly used the word "sexuality," and the model predicted that it was for mature audiences. However, the story was about a LGBTQ+ character coming out and was therefore appropriate for teen audiences.
- There are also unexpected patterns that need extensive cleaning and adjustment of data parsing. For instance, there are stories with random non-linguistic characters. These are much more difficult to be cleaned out automatically as they are texts with alphabetic characters but not in any known languages.

6 Conclusion and Future Work

Conclusion: We first explored multiple architectures such as LSTM, CNN, and a combined RCNN combined and compared with a logistic baseline model, using only story text with Glove embedding. The baseline model performed poorly, just slightly better than a random guess. The combined R-CNN model that we adapted from a paper did not perform better than our CNN and RNN models, so it was not considered later for tuning.
After hyperparameter tuning and further adjustments in the modeling process, the CNN and RNN models' performance had an increase in about 3% of accuracy. Both of these architectures perform equivalently on story text. Because of the consistent success of LSTM models in text classification, we then chose the bidirectional LSTM model to explore further by adding more features.
We added features such as summary, tags, number of likes (kudos) on top of story text. Tags was the one feature that consistently boosted the model performance from 8% - 10%. The best performing model was the bidirectional LSTM with two features: story text and tags, with an overall accuracy of 0.59, a higher precision than recall, and a more consistent performance across 4 classes. Our error analysis give insights to why our model performance was much lower than state-of-the-art models.
Future Work: with larger data set, we can experiment with Facebook "fastText", which is a library for efficient learning of word representations and sentence classification. This might help with rare words and words not in the training corpus. We can also expand on the architecture search to more complex models and the use of transformers such as Google BERT, a new technique for NLP pre-training.
Currently, our model the overall accuracy varies about 1-2% between different runs. We would consider doing cross validation to achieve more stable results, prevent over-fitting, and to validate our model performance when generalize to new data. Other options include experimenting with different methods of regularization such as early stopping, L1, L2 or combination of both. We could scrap more data to reduce this high variance problem.
We can do a more extensive systematic search for hyperparameters and weighted loss functions. For comparison, expand our baseline models to Support Vector Machines and Naive Bayes as these are often seen as baseline models in addition the popular logistic regression.
Lastly but maybe most importantly, as we discussed in our error analysis, getting high quality labeled data would be crucial to our modeling process and to be able to validate our model performance.

7 Contributions

Oishi: identified the original task and data source, scraped the data from archiveofourown.org and manually cleaned out hundreds of mislabeled non-English stories. She contributed heavily to building the RNN model and tuning the hyperparameters. She also performed an in-depth error analysis and helped with the project write-ups and implementation search/literature review.

Minh: literature review of word representation methods and models for text classification, built and compared CNN, RCNN, and multiple RNN models with different combinations of features, tuned hyperparameters, customized the optimizer and loss function for classification tasks, wrote report (parts 2,4,5,6), and helped with some poster content.

Christine: Wrote parsing tool for interpreting raw text and extracting features from original data source (post-scraper) and cleaned additional data. Contributed to background readings/lit review on project proposal, milestone, and images for report/poster. Helped with writing and formatting the final report and wrote and designed poster.

References

[1] Pennington, J., Socher, R., Manning, C. (2014). Glove: Global vectors for word representation. In Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP) (pp. 1532-1543).

[2] Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In Advances in neural information processing systems (pp. 3111-3119).

[3] Hochreiter, S., Schmidhuber, J. (1997). LSTM can solve hard long time lag problems. In Advances in neural information processing systems (pp. 473-479).

[4] Graves, A. (2012). Long short-term memory. In Supervised sequence labelling with recurrent neural networks (pp. 37-45). Springer, Berlin, Heidelberg.

[5] Cho, K., Van Merriënboer, B., Bahdanau, D., Bengio, Y. (2014). On the properties of neural machine translation: Encoder-decoder approaches. arXiv preprint arXiv:1409.1259.

[6] Chung, J., Gulcehre, C., Cho, K., Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv preprint arXiv:1412.3555.

[7] Liu, P., Qiu, X., Huang, X. (2016). Recurrent neural network for text classification with multi-task learning. arXiv preprint arXiv:1605.05101.

[8] Tai, K. S., Socher, R., Manning, C. D. (2015). Improved semantic representations from tree-structured long short-term memory networks. arXiv preprint arXiv:1503.00075.

[9] Conneau, A., Schwenk, H., Barrault, L., Lecun, Y. (2016). Very deep convolutional networks for text classification. arXiv preprint arXiv:1606.01781.

[10] Kalchbrenner, N., Grefenstette, E., Blunsom, P. (2014). A convolutional neural network for modelling sentences. arXiv preprint arXiv:1404.2188.

[11] Yoon Kim. 2014. Convolutional neural networks for sentence classication. arXiv preprint arXiv:1408.5882.

[12] Wang, P., Xu, B., Xu, J., Tian, G., Liu, C. L., Hao, H. (2016). Semantic expansion using word embedding clustering and convolutional neural network for improving short text classification. Neurocomputing, 174, 806-814

[13] Lee, J. Y., Dernoncourt, F. (2016). Sequential short-text classification with recurrent and convolutional neural networks. arXiv preprint arXiv:1603.03827.

[14] Wang, X., Jiang, W., Luo, Z. (2016). Combination of convolutional and recurrent neural network for sentiment analysis of short texts. In Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers (pp. 2428-2437).

[15] Zhou, C., Sun, C., Liu, Z., Lau, F. (2015). A C-LSTM neural network for text classification. arXiv preprint arXiv:1511.08630.

[16] Zhou, P., Qi, Z., Zheng, S., Xu, J., Bao, H., Xu, B. (2016). Text classification improved by integrating bidirectional LSTM with two-dimensional max pooling. arXiv preprint arXiv:1611.06639.

[17] Yang, Z., Yang, D., Dyer, C., He, X., Smola, A., Hovy, E. (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 (pp. 1480-1489).

[18] Zhou, P., Shi, W., Tian, J., Qi, Z., Li, B., Hao, H., Xu, B. (2016). Attention-based bidirectional long short-term memory networks for relation classification. In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers) (Vol. 2, pp. 207-212).

[19] Devlin, J., Chang, M. W., Lee, K., Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805.

[20] Joulin, A., Grave, E., Bojanowski, P., Douze, M., Jégou, H., Mikolov, T. (2016). Fasttext. zip: Compressing text classification models. arXiv preprint arXiv:1612.03651.

[21] Smith, S. L., Kindermans, P. J., Ying, C., Le, Q. V. (2017). Don't decay the learning rate, increase the batch size. arXiv preprint arXiv:1711.00489.

Special Thanks

We thank our project TA **Hojat Ghorbani** for providing extremely helpful feedback and advice every week, **Gael Colas** for extensive help with everything during weekly office hours, **Weini Yu** for initial advice on our project and troubleshooting parts of our final codes, **Shervine Amidi** and **Farzan Farnia** for initial advice with python packages to use for modeling, **Sarah Najmark** for advice on loss function, **Ahmad Momeni** for advice with troubleshooting text processing bugs and evaluation metrics, and **Hoormazd Rezaei** for advice on final project report.