2020-07-30

Age-Suitability Prediction for Literature Using Deep Neural Networks

Eric Robert Brewer
Brigham Young University

Age-Suitability Prediction for Literature Using Deep Neural Networks

Eric Robert Brewer

A thesis submitted to the faculty of
Brigham Young University
in partial fulfillment of the requirements for the degree of

Master of Science

Yiu-Kai Ng, Chair
Deryle W. Lonsdale
Kent Seamons

Department of Computer Science

Brigham Young University

ABSTRACT

Age-Suitability Prediction for Literature Using Deep Neural Networks

Eric Robert Brewer
Department of Computer Science, BYU
Master of Science

Digital media holds a strong presence in society today. Providers of digital media may choose to obtain a content rating for a given media item by submitting that item to a content rating authority. That authority will then issue a content rating that denotes to which age groups that media item is appropriate. Content rating authorities serve publishers in many countries for different forms of media such as television, music, video games, and mobile applications. Content ratings allow consumers to quickly determine whether or not a given media item is suitable to their age or preference. Literature, on the other hand, remains devoid of a comparable content rating authority. If a new, human-driven rating authority for literature were to be implemented, it would be impeded by the fact that literary content is published far more rapidly than are other forms of digital media; humans working for such an authority simply would not be able to issue accurate content ratings for items of literature at their current rate of production. Thus, to provide fast, automated content ratings to items of literature (i.e., books), we propose a computer-driven rating system which predicts a book's content rating within each of seven categories: 1) *crude humor/language*; 2) *drug, alcohol, and tobacco use*; 3) *kissing*; 4) *profanity*; 5) *nudity*; 6) *sex and intimacy*; and 7) *violence and horror* given the text of that book. Our computer-driven system circumvents the major hindrance to any theoretical human-driven rating system previously mentioned—namely infeasibility in time spent. Our work has demonstrated that mature content of literature can be accurately predicted through the use of natural language processing and machine learning techniques.

Keywords: machine learning, neural net, document classification, book content rating

ACKNOWLEDGMENTS

Barbara, Joy, Mom, Dad, Dr. Ng, Alisha Banskota.

Table of Contents

1 INTRODUCTION

In order to inform consumers about the presence of mature content in digital media, various *content rating authorities* have been established. Examples of content rating authorities which operate in the US include the Motion Picture Association (MPA)[1] and the Entertainment Software Rating Board (ESRB)[2]. These authorities primarily listen to, watch, and *rate* media items—films, television, video games, music, or mobile applications—according to their level of maturity. A content rating is meant to be objective and unbiased and is not to be construed as a judgment on the quality of that media item. Then, the assigned content rating is prominently displayed alongside the media item, designed to be visible to a consumer at the point of purchase. An authorized content rating for a particular media item informs consumers about the level of maturity of that item's content and ultimately helps consumers decide whether or not to view or listen to that media.

Publishers of most forms of digital media may request (or are required to obtain) a content rating from a rating authority in the country where that media is published. For example, the MPA issues content ratings to films based on age-appropriateness as depicted in Table 1.1.

Rating	**Name**	**Description**
G	General Audiences	All ages admitted
PG	Parental Guidance Suggested	Some material may not be suitable for children
PG-13	Parents Strongly Cautioned	Some material may be inappropriate for children under 13
R	Restricted	Under 17 requires accompanying parent or adult guardian
NC-17	No One 17 and Under Admitted	

Table 1.1: Possible content ratings given to films by the MPA.

[1]MPA, `https://www.motionpictures.org/film-ratings/`
[2]ESRB, `https://www.esrb.org/ratings/`

When determining which rating a particular media item will receive, content rating authorities consider to what degree mature content is present in that media across different themes. Themes which are commonly considered when determining a content rating include: *drug, alcohol, and tobacco use*; *use of profanity and crude humor/language*; *sex, intimacy, and nudity*; and *violence/horror*. In the case of films in the U.S., for a film to be rated **G**, it must contain virtually no content related to any of these mature themes. A film that significantly exhibits some of the mentioned mature themes would be assigned a more stringent rating, such as **R**. Any reasons for a given content rating are commonly displayed alongside the rating itself. Examples of content ratings for various media are shown in Figure 1.1.

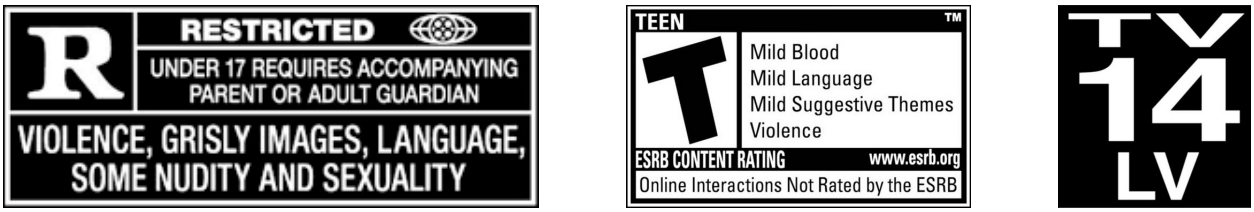


Figure 1.1: Authoritative content ratings for a movie (left), a video game (center), and a television program (right)—LV stands for language & violence.

In an organization that issues content ratings, such as the MPA, it is typical for human advisory boards to view, listen to, and/or read media content to determine its rating. In terms of time spent, human-based advisory is satisfactory because members of these advisory boards are able to view, discuss, and issue content ratings for media items in a timely manner, i.e., at least as quickly as the media itself is produced. For example, fewer than eight hundred films were rated by the MPA[3] in 2013. Likewise, Internet Movie Database (IMDB) lists eight hundred video games released in the same year[4], each of which has been issued a content rating by the ESRB. However, the International Publishers' Association (IPA) reported that over 400,000 books were published in 2013 in the United States alone[5]. Thus, to analyze

[3] `https://www.the-numbers.com/market/2013/mpaa-ratings`

[4] `https://www.imdb.com/search/title?title_type=video_game&year=2013-01-01,2013-12-31&view=simple`

[5] `https://www.internationalpublishers.org/images/aa-content/ipa-reports/ipa-annual-report-2013-14.pdf`

literature manually (i.e., by humans) would not be feasible in amount of time spent due to the large quantity of books to be rated.

Currently, literature is freely published without any unified rating system. This means that consumers may not be clearly informed of any mature content in a book before they read it. Though several third-party book rating organizations exist (discussed in Section 4.1), none have become an industry standard to which book publishers can turn.

It is reasonable to consider the use of computers instead of humans to analyze literary content because computer algorithms run quickly, thus overcoming the relatively slow speed at which humans are able to read literature. However, a computer-aided text analysis system must also classify mature content with a near-human level of accuracy.

In this work, we scratch the surface of computer-aided literary content analysis through the use of natural language processing and machine learning techniques. We framed the problem as a traditional supervised machine learning problem, specifically, text classification. Our model takes as its input the text of a book and predicts that book's maturity rating levels across seven categories: 1) *crude humor/language*; 2) *drug, alcohol, and tobacco use*; 3) *kissing*; 4) *profanity*; 5) *nudity*; 6) *sex and intimacy*; and 7) *violence and horror*. We collected the texts and quantifiable maturity rating levels for over six thousand books and compiled them into a novel data set. We used a neural network architecture to classify items of literature and to detect which parts of the book contain mature content. We adapted this network to be able to classify and detect mature content in considerably long texts, i.e., entire fictional novels.

Our work exhibits a practical application of leveraging existing human-annotated labels (treated as ground-truth) to predict labels for novel instances. We feel that our work could benefit the reading community as a whole, specifically parents and children. Further, relating to the field of machine learning as a whole, we feel that our work is a significant example of overcoming the challenge of extracting meaningful features from textual data within very large documents.

2 RELATED WORK

This chapter discusses previous work in the context of literary analysis of mature themes as performed by humans and by computers. It ends with a discussion on the field of document classification in general.

2.1 Human Analysis of Mature Content

Various researchers have successfully detected the presence of substance use [11], profanity [13], sexuality [5, 39], and violence [12] in adolescent and young adult literature. In each of these studies, mature contents were *coded*, i.e., classified, by human annotators trained "by jointly coding non-sample examples and openly discussing coding protocols as they were applied" [12]. Then, each annotator independently read and coded the contents of five to seven books. These studies are useful in that they shed light on the prevalence of mature content in popular literature and underscore the need to inform interested readers of any mature themes in a book without having to read it. Further, these studies highlight the relative slowness of humans in comparison to computers of reading and annotating literature. To us, these studies provoke the question, "can computers repeat these analyses with comparable accuracy?"

2.2 Computer Analysis of Mature Content

A study similar to ours has been conducted by Wanner et al. [37] who not only quantified the mature contents of a book, but also considered its readability, story complexity, and other elements in order to answer the question, "is this book suitable for children?". In their study, Wanner et al. estimate mature contents of books by compiling a hand-picked list of sensitive

terms for each of six topics: *war*, *crime*, *sex*, *horror*, *fantasy*, and *science fiction*. They then extend their lists of sensitive words with synonyms and hypernyms taken from WordNet [15]. Wanner et al. developed a heuristic score function where a higher score denotes a greater magnitude of mature content. This work was exciting to us in that it showed us how other researchers are working on the same problem.

Chen et al. [6] have investigated the discrepancies between maturity ratings for applications published through two of the largest mobile application services, Apple App Store and Google Play. In their study, they build a novel maturity rating classifier using a vocabulary of human-selected terms extracted from user reviews. Their work showed promise in that mature content can indeed be accurately detected in text.

Following the work done by Chen et al., Hu et al. [19] performed an analysis of mature content in mobile applications. They used textual descriptions of apps collected from the Apple App Store and Google Play to predict their maturity levels. They first trained a word2vec [28] embedding over their corpus of app descriptions to obtain a word vector for each unique word token in the vocabulary. Next, they hand-picked *sensitive* words from the App Store and Google Play maturity rating policy descriptions that directly refer to mature content such as *violence*, *horror*, *humor*, *profanity*, *sex*, *nudity*, *mature*, *alcohol*, *tobacco*, *drugs*, *gambling*, etc. Then, for each word in the list of sensitive words, they find the two hundred words whose vector representations are most similar to that word as measured by cosine similarity as *augmented* sensitive words. Finally, they use the sensitive words, the augmented sensitive words, and a TF-IDF-weighted bag-of-words vector model to predict maturity levels and calculate an overall maturity rating. Their results and approach were significant contributions to the subject of text analysis. However, their data set significantly differs from ours in terms of size—an app description for an app is on the order of a few paragraphs whereas a typical book in our data set contains hundreds of paragraphs.

3 DOCUMENT CLASSIFICATION

We first discuss in this chapter the necessary pre-processing step of *tokenization* (not to be confused with the concept with the same name in the field of computer security) over a text corpus. We then discuss and compare two different methods of representing textual tokens to learning algorithms—*bag-of-words* and *word vectors*. Finally, we briefly summarize the current state of the art in document classification algorithms for both of the representation methods mentioned.

3.1 Tokenization

Our task qualifies as document classification, since it is to classify the level of maturity for a book given a textual input—the text of the book. In order to perform numerical analysis on text, i.e., presenting the text to a machine learning classifier, the entirety of the text must be transformed into numerical values. To a computer, a document is simply an ordered sequence of symbols where each symbol is represented by a unique byte value (or sequence of byte values). Before using either of the bag-of-words or word vector approaches, an ordered sequence of symbols must be divided into smaller, more meaningful parts, usually as an ordered sequence of markers, i.e., punctuation, and words, collectively called *tokens*. Consider the trivial document '`Her handbag wasn't returned to her.`'. In the English language, words are separated by a space; thus, a simple first step would be to divide the document into parts by splitting on one or more whitespace symbols. To further tokenize this document, other, smaller decisions may be made such as: whether or not to convert the first word, '`Her`', to lower case; whether the contraction '`wasn't` should been treated as a single token

or as two (‘was’, ‘n’t’) or three (‘wasn’, ‘’’, ‘t’); or whether the end of sentence marker (period) should be joined with the final word as ‘her.’, treated as a separate token ‘.’, or omitted altogether. Each tokenization decision can directly affect the size of the vocabulary, or the total number of unique tokens among all documents, as well as the representations of documents as bag-of-words or word vectors. Popular algorithms for tokenizing documents include Stanford[1] and Treebank[2].

For large corpora, it is usually not reasonable to uniquely represent all word tokens in the vocabulary, since, according to Zipf's law [27], tokens that appear less frequently in a corpus tend to appear at an exponentially decreasing rate. Thus, it is common to choose a fixed vocabulary size v when tokenizing a corpus, where only the most frequently-occurring v tokens are uniquely represented. Tokens that appear less frequently than the v-th token can either be omitted entirely, or represented with a OOV, or out-of-vocabulary placeholder value. For our task, we wanted to reduce the size of the vocabulary as much as possible, which would, in turn, increase the probability that any particular token in a document could be uniquely represented. Hence, we converted all text to lowercase to avoid duplicate lowercase and uppercase spellings of the same word. We also used the Stanford Tokenizer which treats end-of-sentence markers as separate tokens and splits common two-part contractions into its parts as ‘wasn’t’ becomes ‘was’ and ‘n’t’.

3.2 Bag-of-words

There are two classes of approaches to quantify a textual document. The first approach is to represent the entire document as a *bag-of-words*. The second is to represent all of the individual word (or character) tokens in the document as *vectors* in $\mathbb{R}^n$.

A simple way to represent a document is to count the number of occurrences of each unique word token in that document [17]. If the vocabulary among all documents was, for example, {‘give’, ‘handbag’, ‘her’, ‘returned’, ‘the’, ‘to’, ‘wasn’t’, ‘.’}, then the document

[1] https://nlp.stanford.edu/static/software/tokenizer.shtml
[2] ftp://ftp.cis.upenn.edu/pub/treebank/public_html/tokenization.html

from the previous section would be represented as a vector of the frequency of occurrence for each token, {`0`, `1`, `2`, `1`, `0`, `1`, `1`, `1`}, since the tokens '`give`' did not appear at all, '`handbag`' appeared once, '`her`' appeared twice, etc. Aside from using the raw frequency of occurrence of each token, a binary representation could be used, where a '`1`' indicates that a token exists in the document, and a '`0`' indicates its absence. Alternatively, the vector of counts can be weighted using TF-IDF (*term frequency, inverse-document frequency*) [34].

One appealing factor of bag-of-words representations is their simplicity to calculate. One of their main drawbacks, however, is their general sparsity—for most documents, many tokens in the vocabulary will not appear at all, but must be represented as a zero in the vector. The sparse vector may result in increased computational complexity in space and/or time. Another significant oversight of bag-of-word models is their inability to capture the context of a document, since word order is not preserved. This means that two documents with different word orderings and/or different semantics may have the same token frequencies, and thus, will have the same bag-of-words representation. Because we want our classifier to capture contextual clues in texts, we chose to represent documents as sequences of word vectors rather than as a bag-of-words.

3.3 Word Vectors

A collection of static word vectors, also called a word *embedding*, can be obtained using algorithms such as word2vec [28], GloVe [29], or fastText [2] over a text corpus. Such algorithms construct vectors, or an embedding, for word tokens by first creating a mapping from tokens to arbitrary points in an n-dimensional space, then iteratively optimizing these vectors in such a way that tokens which frequently occur in similar *contexts* (as measured by the probability of co-occurrence of surrounding tokens within a fixed-size window) should have similar vector representations by cosine similarity or Euclidean distance or both. After word vectors are sufficiently optimized, one would expect that different tokens which are used in similar contexts such as `car` and `automobile` should have similar vector representations

(i.e., a cosine similarity close to 1, and/or a small Euclidean distance). Although they are commonly phrased *word* vectors, any token or symbol that is not a word (such as a punctuation symbol) may have its own vector representation.

A word embedding serves as a mapping from unique tokens to their corresponding vectors in $\mathbb{R}^d$ where d is chosen as the number of dimensions in each word vector. Certain classes of machine learning algorithms are able to utilize sequences of word vectors in place of the token sequences. One advantage to doing so is that the ordering of the tokens in the original text can be preserved. Thus, patterns that occur sequentially in the text can be observed by the classifying algorithm.

3.4 Learning Algorithms—Bag-of-words

This section includes a variety of general purpose machine learning algorithms which are able to effectively process large textual inputs represented as a bag-of-words.

3.4.1 K-Nearest Neighbors

K-Nearest Neighbors [10] is unique from other supervised learning algorithms in that it does not construct a formal model through iterating over a training set. Instead, the class for a novel instance is simply predicted as the majority class among the k (≥ 1) instances in the training set that are nearest to the novel instance. To quantify 'nearest to the novel instance', a distance function $d(x, y)$ must be defined. Typical distance functions include Euclidean, Manhattan (city block), or cosine similarity. It is recommended to choose a k that avoids ties during polling, for example, one that is relatively prime to the number of classes.

3.4.2 Logistic Regression

Linear regression attempts to find a best-fit line through a series of data points with any number of independent variables $\vec{x}$ and a free, or dependent variable y. Logistic regression, on the other hand, attempts to find a best-fit *logistic curve* through a similar series of data

points on the condition that y is binary, meaning it either occurs (denoted by a 1) or does not (0). A logistic curve in the form of $p = \frac{1}{1+e^{-(\vec{m}\cdot\vec{x}+b)}}$ guarantees that the output will fall in the bounds $[0, 1]$ and allows its output to be treated as a probability. Figure 3.1 compares linear regression and logistic regression.

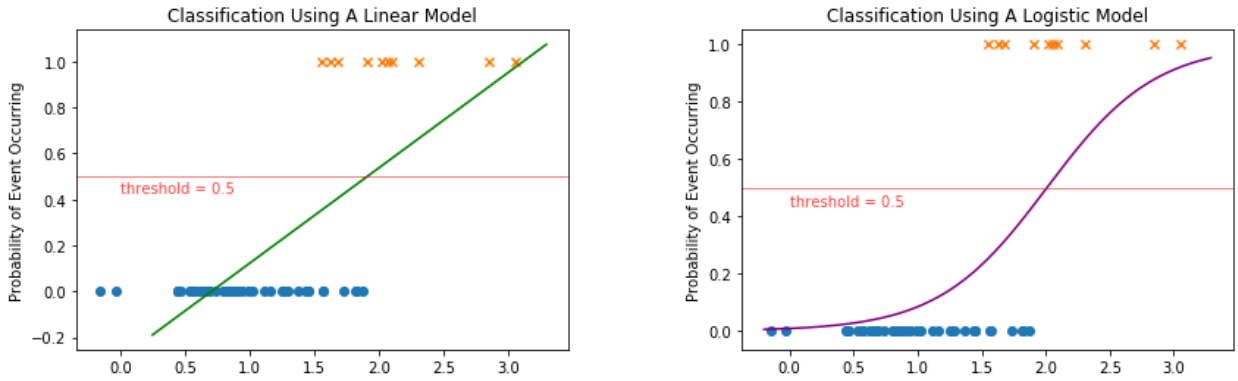


Figure 3.1: A linear model and a logistic model fit to the same data set. Where a linear regression model can exceed 1 or fall below 0, the logistic function is bounded by $[0, 1]$. For classification tasks, a threshold is typically set to 0.5.

3.4.3 Multi-layer Perceptron

Also called a feed-forward neural network, a multi-layer perceptron [31] is made up of ordered layers of nodes (sometimes called neurons in reference to the nervous system of animals) starting from the input layer and ending at the output layer where each node represents a single input value. Between successive layers of nodes, $\vec{x}$ and $\vec{z}$, lies a complete bipartite graph of edges W termed *weights* or a *weight matrix*, where each weight $W_{j,i}$ represents the relative importance of the input value $\vec{x}_i$ to the corresponding output value $\vec{z}_j$. The values of a given layer $\vec{z}$ are calculated as the product of the weight matrix W and the input values $\vec{x}$, or $\vec{z} = W\vec{x}$. All subsequent layers are calculated in the same way until the final output layer is reached. Figure 3.2 depicts a basic neural network.

The network is trained via a process called backpropagation [33], a class of stochastic gradient descent [22, 30]. For a given input vector, the gradient at all layers is calculated

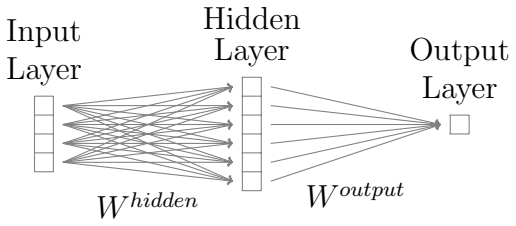


Figure 3.2: The architecture of a very simple multi-layer perceptron with one hidden layer and a single output node. One may imagine that more hidden layers precede the output layer, resulting in increasing complexity.

according to a minimizing loss function. Then, the weights are adjusted a tiny step in the direction of the gradient, approximating a descent toward some minimum—ideally the global minimum—in the convex loss landscape. In practice, each cycle of backpropagation occurs with a *batch*, or random sample, of inputs (rather than with a single input) where the gradient from which weights are adjusted is averaged over the gradients calculated for each input in the batch. Training with small batches better approximates descent and can be orders of magnitude faster than calculating the gradient over the entire data set, i.e., the "true" gradient [38].

The multi-layer perceptron serves as the backbone of each of the algorithms that utilize word vectors described in Section 3.5 in that each algorithm is itself a neural network with special layers designed for handling sequences of tokens as their input.

3.4.4 Multi-nomial Naïve Bayes

For classification tasks, standard Naïve Bayes computes the *posterior* probability of a particular data point $\vec{x} \in \mathbb{R}^n$ belonging to the class C_k as $p\left(C_k|\vec{x}\right) = \frac{p(C_k)p(\vec{x}|C_k)}{p(\vec{x})}$ based on the general Bayes' theorem $p\left(A|B\right) = \frac{p(A)p(B|A)}{p(B)}$. Because Naïve Bayes assumes that the *evidence* $p\left(\vec{x}\right)$ is constant, it is omitted, giving $p\left(C_k|\vec{x}\right) \propto p\left(C_k\right)p\left(\vec{x}|C_k\right)$. The class *prior* $p\left(C_k\right)$ is easily calculated as the probability of a data point belonging to the class C_k. And the *likelihood* is calculated with the "naïve" assumption that all dimensions of $\vec{x}$ occur independently as $p\left(\vec{x}|C_k\right) = \prod_{i=1}^{n} p\left(x_i|C_k\right)$ where $p\left(x_i|C_k\right)$ is the probability of x_i among

all data points whose class is C_k. To predict which class a particular data point belongs to, the posterior is calculated for each class $C_k, 1 \leq k \leq K$, where K is the number of distinct classes, and the most probable class wins.

In the context of text classification, multi-nomial Naïve Bayes [21] extends the traditional Naïve Bayes approach by considering the *frequency* of tokens occurring in a document (as a continuous value), not merely whether it exists or not (as a binary value).

3.4.5 Random Forest

A Random Forest [4] classifier is an ensemble of many independent *decision trees*. An ordinary decision tree model is built by recursively choosing decision boundaries represented by *decision nodes* for features in the input $\vec{x}$ that cause the data points to be divided most evenly. *Leaf nodes* in the tree represent the most likely class for data points which would traverse there by following decision paths starting from the root. A simple decision tree is shown in Figure 3.3.

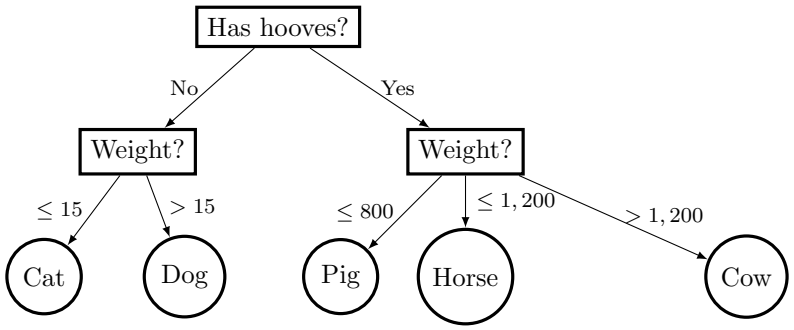


Figure 3.3: A decision tree to classify animals using two features. Decision nodes are denoted by rectangular nodes. Leaf nodes are denoted by circular nodes. During prediction time, an unknown instance representing an animal would traverse the tree starting at the root until it reaches a leaf node.

3.4.6 Support Vector Machine

A support vector machine [9] is a binary linear classifier which deterministically finds the multi-dimensional hyperplane that divides data points in the positive class from those in the negative class such that the margin between the nearest data points to the hyperplane

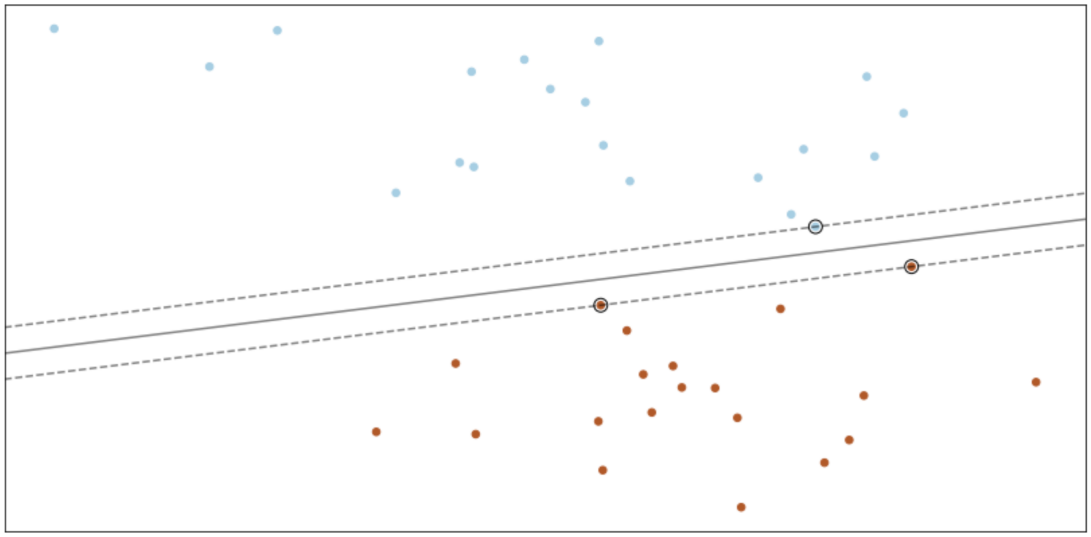


Figure 3.4: The maximally-dividing decision boundary (solid line) found by a support vector machine. The support vectors (circled points) are equidistant from the decision boundary.

and the hyperplane itself are maximized. Finding the maximally-dividing hyperplane among a training set of data points tends to decrease the likelihood of misclassifying novel data points. The data points which are equidistant from the maximally-dividing hyperplane, termed support vectors, are found through an optimization method. Figure 3.4 depicts such a hyperplane between two classes of data points. Because SVM is inherently a linear classifier, input values may be need to be transformed via a kernel function to a higher-dimensional space where a hyperplane can sufficiently divide data points of different classes.

3.5 Learning Algorithms—Word Vectors

Each algorithm which we describe below happens to be a neural network that is able to process variable-sized textual input.

3.5.1 Recurrent Neural Networks

In general, recurrent neural networks process sequential input, x, from the beginning of the sequence to the end, meaning they are able to capture patterns that can only be observed in

the context of neighboring elements [20, 33]. For each token vector x_t that is processed at time t, $1 \leq t \leq T$, where T is the number of tokens in the sequence, a new vector h_t—called the *hidden state*, or context vector, at time t—is computed as a function of the previous hidden state and the current input, or, $h_t = f(h_{t-1}; x_t)$. A unique hidden state vector is produced for every item in the sequence.

Two popular variants of RNNs are the Long-Short Term Memory (LSTM) [18] and the Gated Recurrent Unit (GRU) [7]. Both the LSTM and the GRU are specific implementations of the recurrent function f. They have both been shown to perform well on a variety of sequential learning tasks [8].

RNNs tend to be particularly effective for solving text classification tasks, since, when given a sequence of word vectors, an RNN continually updates its hidden state, or context vector, based on the next word in the sequence in a manner similarly to the way humans process a textual passage.

3.5.2 Convolutional Neural Networks

In the context of text classification, a convolution operation works as follows. The input to the operation is a sequence of T word vectors x_1 through x_T in $\mathbb{R}^d$. A *filter size*, or *kernel size*, r ($\leq T$), is chosen as the number of words in the sequence to process at a time. A *filter* with r rows and d columns is instantiated containing adjustable weights which will be optimized during training. First, the filter produces one scalar output as the sum of element-wise products between each element in the filter and each element in the first r word vectors x_1 through x_r. Then, another scalar output is produced from the sum of element-wise products between the filter and the next sequence of r word vectors—x_2 through x_{r+1}. New scalar outputs are iteratively produced in this fashion until the filter has passed over all r-length sequences of word vectors, ending on the word vectors x_{T-r+1} through x_T. Thus, a total of $T - r + 1$ scalar values are produced by the convolution operation. This operation

can be thought of as a fixed-sized window of width r "sliding" from the beginning of each input sequence to its end. An example is shown in Figure 3.5.

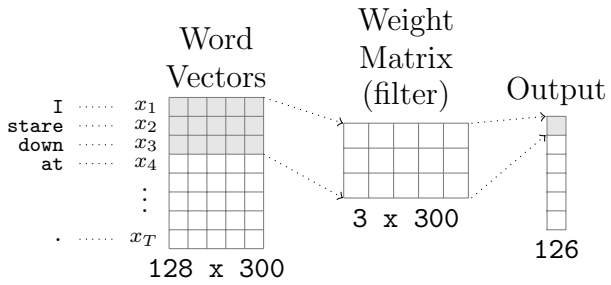


Figure 3.5: A single convolution operation. These operations are performed on sequences of word vectors. The filter contains floating point-valued weights that are optimized during training. The output for a single operation is a scalar. In this example, the input paragraph contains $T = 128$ tokens and the filter has $size = 3$; thus, the length of the output vector is $T - r + 1 = 128 - 3 + 1 = 126$.

In practice, one convolution operation of a filter containing r rows upon a sequence of T word vectors produces a single vector with $T - r + 1$ dimensions. Typically following a convolutional layer in a neural network is a *max-pooling* layer, which simply outputs the maximum scalar value from its input. A max-pooling layer in a neural network is strictly functional, i.e., it contains no trainable weights. The intuition behind these two types of layers is that the convolutional layer captures the semantics of every group of r consecutive word tokens, while the max-pooling layer extracts the strongest "signal", or what is assumed to be the most informative group of word tokens among the entire sequence.

Convolutional layers in deep neural networks are highly effective in the field of image classification [25] and have more recently been adopted for text classification [23].

Recurrent layers and convolutional layers differ in that a recurrent layer is theoretically able to capture relationships in input sequences over long distances, whereas convolutional layers can only process short spatial localities within a sequence.

3.5.3 Self-attention

A self-attention mechanism is an extension of a recurrent neural network where subsequent hidden state vectors h_t are not produced solely based on the previous hidden state vector h_{t-1} and the current input x_t, but also on a weighted combination of all previous hidden state vectors produced by the RNN [1]. The weighting function can be learned through backpropagation along with the rest of the network. This approach prevents the neural network from having to encode the context of an entire sequence into a single fixed-length vector. Self-attention has gained popularity in recent years for tasks such as machine translation [26, 36] since the self-attention mechanism "emulates searching through a source sentence during decoding a translation".

3.5.4 Hierarchical Attention Networks

Another type of network which has increased in popularity in recent years is the Hierarchical Attention Network, or HAN [40]. A HAN works in two phases: first, the features of each sentence are extracted from word tokens by a shared self-attention mechanism, called the word encoder; then, features of each sentence of the document are extracted from the outputs of the word encoder via another self-attention mechanism, called the sentence encoder.

4 METHODOLOGY

We define our problem as a traditional supervised machine learning problem: our input datum for a single book consists only of its text; our targets are distinct rating levels, one for each maturity category. To give consumers insight toward the suitability of the book as a whole, an overall maturity rating is also derived from the categorical maturity rating levels.

In addition to predicting the content rating for a book as a whole, we also predicted the content rating levels of each paragraph of each book in order to detect the paragraphs in those books that contain mature content.

This chapter is split into four parts. First, we briefly discuss how input and output data was obtained. Second, we describe the nature of our categorical rating levels and how we manipulate them into labels used as the ground-truth by our machine learning model. Third, we explain in detail the baseline classifiers we used to classify maturity rating levels. Finally, we introduce the neural network that we designed for this task.

4.1 Data Collection

Before we built a predictive model, we obtained both the input and output data. We began by collecting the output data, namely, content ratings for books.

After searching through various third-party, online rating systems for literature, we settled on the one that was the most robust and complete—Book Cave[1]. As of June 2020, the publicly available Book Cave database contains over seventeen thousand books, each of

[1] `https://mybookcave.com/mybookratings/`

which has been assigned at least one overall content rating. Ratings for books on Book Cave are issued by users of the website who have acknowledged that they have read through the entire book at least once. For this reason, we were fairly confident in treating the ratings as 'ground-truth', though we were aware that the data would be unavoidably skewed by some amount of human subjectivity. We assumed that user bias would be present in our data regardless of the online rating system we would adopt.

We investigated other third-party rating systems, but dismissed them in favor of Book Cave. CommonSenseMedia's[2] database contains primarily books written for young children, adolescents, and teens, whereas we are interested in books for all audiences. The rating data for both of Rated Reads[3] and Parental Book Reviews[4] are more qualitative than quantitative—users post feedback for books in the form of reviews, noting any mature themes, but lacking a discrete *level* of maturity across categories. Further, the number of obtainable, rated books on each of these sites was too few (on the order of hundreds of books compared to Book Cave's thousands) for our purposes of machine classification.

Book Cave users assign content ratings to books across the following seven categories: *crude humor/language*; *drug, alcohol, and tobacco use*; *kissing*; *profanity*; *nudity*; *sex and intimacy*; *violence and horror*. An overall rating for a book is determined by the maximum overall rating level yielded by the assigned categorical rating levels over all categories as shown in Table 4.1. The possible overall ratings for books are, in order of severity: *All Ages*, *Mild*, *Mild+*, *Moderate*, *Moderate+*, *Adult*, and *Adult+*. If more than one user has rated a particular book, all of the overall ratings assigned by those users are averaged, rounding up to the nearest overall rating. An example user rating from the Book Cave website is shown in Figure 4.1. We collected the overall ratings and categorical rating levels for all books in the Book Cave database.

[2] `https://www.commonsensemedia.org/`
[3] `http://ratedreads.com/`
[4] `https://sites.google.com/site/parentalbookreviews/`

Category	Rating Level	Overall Rating	Base Overall Rating
Crude humor/ language	None	All Ages	All Ages
	Mild crude humor	Mild	Mild
	Moderate crude humor/language Significant crude humor/language	Moderate Moderate+	Moderate
	Extensive crude humor/language	Adult+	Adult
Drug, alcohol, & tobacco use	None	All Ages	All Ages
	Mild substance use Some substance use	Mild Mild+	Mild
	Moderate substance use by adults and/or some use by minors Significant substance use	Moderate Moderate+	Moderate
	Extensive substance abuse	Adult	Adult
Kissing	None	All Ages	All Ages
	Mild kissing Passionate kissing	Mild Mild+	Mild
Profanity	None	All Ages	All Ages
	Mild language	Mild+	Mild
	Some profanity (6 to 40) Moderate profanity (41 to 100)	Moderate Moderate+	Moderate
	Significant profanity (101 to 200) Significant profanity (201 to 500) Extensive profanity (501+)	Adult Adult Adult+	Adult
Nudity	None	All Ages	All Ages
	Brief (nonsexual) nudity Brief nudity	Mild Mild+	Mild
	Some nudity	Moderate	Moderate
	Extensive nudity	Adult	Adult
Sex & intimacy	None	All Ages	All Ages
	Mild sensuality	Mild+	Mild
	Non-graphic sexual references Non-detailed fade-out sensuality Fade-out sensuality with details or significant sexual discussion	Moderate Moderate Moderate+	Moderate
	Semi-detailed onscreen love scenes Detailed onscreen love scenes Repeated graphic sex Menage or BDSM sex	Adult Adult Adult+ Adult+	Adult
Violence & horror	None	All Ages	All Ages
	Mild (nonsexual) violence or horror Some violence or horror	Mild Mild+	Mild
	Moderate violence or horror	Moderate	Moderate
	Graphic violence or horror Extended gruesome and depraved violence or horror	Adult Adult+	Adult

Table 4.1: Categorical rating levels per maturity category and corresponding overall ratings used by Book Cave. Base rating levels are shown and discussed in Section 4.2.

The next, more difficult step came when we set out to collect book texts. Luckily for us, the web page for over 98% of books in the Book Cave database also includes a hyperlink

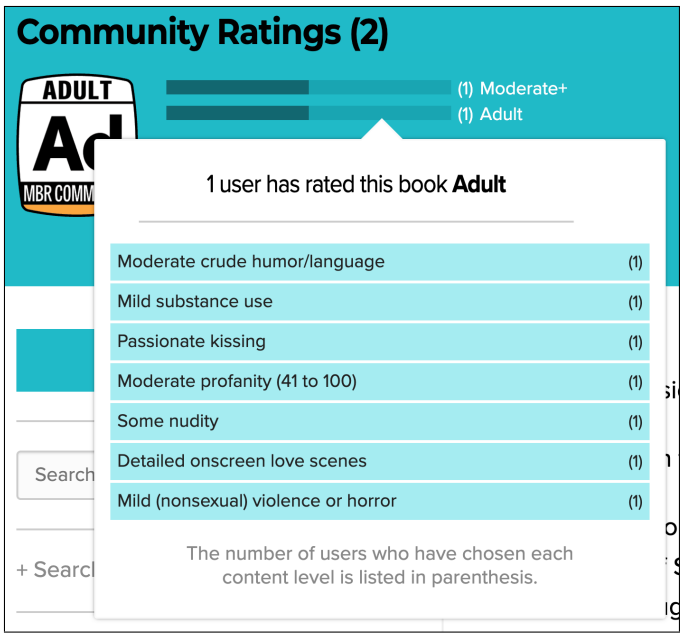


Figure 4.1: An overall maturity rating with corresponding categorical rating levels for a book on the Book Cave website. This book has been rated *Moderate+* and *Adult*, respectively, by two different users of the website (top right). The categorical rating levels that correspond to the *Adult* rating are shown in the popover (light blue bars). The two distinct overall ratings have been averaged and rounded up, resulting in the book's overall *Adult* rating (top left).

to the Amazon Store web page for the Kindle (electronic) version of that book. From the Amazon Kindle Cloud Reader[5] web page we acquired complete texts for over six thousand books[6]. The texts were already broken up into paragraphs, which is an important aspect that we discuss further in Section 4.4. All textual data that we collected is in the English language.

4.2 Content Rating Levels

The content of each book in the Book Cave database has been given a categorical rating level in each of the seven maturity categories. The overall maturity rating for a book is derived as the maximum overall rating over the seven categorical rating levels as seen in the `Overall Rating` column in Table 4.1. In order to represent the categorical rating levels as labels to our predictive model, we grouped together rating levels within each category that have the same *base* (i.e., not including a '+') overall maturity rating. For example, each of 'Significant profanity (101 to 200 [uses of profane terms])', 'Significant profanity (201 to

[5] `https://read.amazon.com/`

[6] Alongside the texts, we also collected a high-resolution front cover image for each of these books. We decided not to use them as input to our classifiers, however, on the basis that the collected labels don't necessarily describe the book's cover.

500)', and 'Extensive profanity (500+)' yield a base overall maturity rating of *Adult*. Thus, books with any of these three rating levels were labeled as the same class (*Adult*) within the *profanity* category. Rating levels for each other maturity category are also similarly merged and are specified in the `Base Overall Rating` column in Table 4.1. We did this to simplify the boundaries between categorical rating levels which yield identical or almost identical overall ratings.

4.3 Ordinal Regression

Because the categorical rating levels naturally represent increasing magnitudes of mature content rather than distinct topics or themes, we follow the approach described by Frank et al. [16] by representing merged categorical rating levels *ordinally*, that is, as a binary vector instead of as an integer, or nominal, value. More formally, an integer-valued merged categorical rating level z is transformed into its vector representation $\vec{y} \in \{0,1\}^{k-1}$ where k is the number of distinct merged integer values for that category and $\vec{y}_i = 1$ if $z \leq i$, else 0, for $1 \leq i \leq k-1$ as seen in Table 4.2 in the column labelled `Class Represented Ordinally`.

Rating Level	**Overall Rating**	**Class ID**	**Class Represented Ordinally**
None	All Ages	0	0 0 0
Mild language	Mild+	1	1 0 0
Some profanity (6 to 40) Moderate profanity (41 to 100)	Moderate Moderate+	2	1 1 0
Significant profanity (101 to 200) Significant profanity (201 to 500) Extensive profanity (501+)	Adult Adult Adult+	3	1 1 1

Table 4.2: Example of merged rating levels and ordinal representation within the *profanity* category. In this case, rating levels which yield an overall rating of *Moderate* or *Moderate+* have been merged as well as those which yield overall ratings of *Adult* or *Adult+*. A class ID value and derived ordinal class representation based on this ID are shown for each merged rating level.

The intuition behind the binary ordinal representation is that books with relatively high categorical rating levels may include harsher words in their vocabularies than those with lower rating levels. In other words, vocabularies of books with higher rating levels are assumed to be supersets of vocabularies of books with lower rating levels. Thus, for a particular category for a book, each `1` in the binary vector expresses that that book contains *at least* that magnitude of maturity for this category. A machine learning model will benefit from this binary vector representation more so than from the original integer value because the model is free to independently learn whether certain subsets of vocabulary terms are associated with different elements in the vector. In contrast, a machine learning model which is given only integer-valued labels may associate a mildly mature word or set of words with a rating level which yields a 'Mild' overall rating but falsely *dissociate* that word from rating levels which yield higher overall ratings.

4.4 Text Pre-processing

We pre-processed the text of each book by converting the text to lowercase and tokenizing the text of each book using the Stanford Tokenizer[7]. We chose the Stanford Tokenizer, since all punctuation marks are treated as separate tokens, aside from apostrophes when used in contractions. This is desirable because our vocabulary is large and other tokenizers would have treated a word immediately followed by an end-of-sentence marker (a period, question mark, exclamation mark, etc.) as a single token. The Stanford tokenizer separates the end-of-sentence marker and the last word of the sentence which decreases the redundancy of the vocabulary.

4.5 Bag-of-words Classifiers

In order to acquire baseline results with which to compare our method, we tried a few traditional machine learning algorithms using bag-of-words representations of texts. The

[7] `https://nlp.stanford.edu/static/software/tokenizer.shtml`

bag-of-words vectors were weighted by TF-IDF. We chose six different classification algorithms given identical TF-IDF-weighted bag-of-words vectors representing texts of books: K-Nearest Neighbor [10]; Logistic Regression; Multi-layer Perceptron, [31] or shallow neural network (i.e., with a single hidden layer); Multi-nomial Naïve Bayes [21]; Random Forest [4]; and multi-class Support Vector Machine [9], as discussed in Section 3.4.

For our K-Nearest Neighbor classifier, we chose $k = 5$ since 5 is relatively prime to the number of classes in each maturity category. As a distance metric, we chose Euclidean distance as $d(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$. We also chose to weigh the 'vote' of each of the 5 nearest neighbors inversely proportionally to its distance from the novel instance, thus allowing nearer neighbors to more strongly influence the predicted class [14].

4.6 Neural Networks

We tested a few different configurations of neural networks for our approach. Each of our network configurations contains two main components which we call *modules*. The first module is the *paragraph encoder*, which takes as input a sequence of word tokens of a single paragraph and produces a fixed-length vector. The second module is the *aggregation* module, or the component which aggregates, or summarizes, all vectors produced by the paragraph encoder for all paragraphs of a book.

The job of the paragraph encoder is to learn a representation of any fixed-length token sequence as another fixed-length feature vector. The text of each book in the Book Cave database was already divided into paragraphs for us. However, because we must pass fixed-length token sequences to the paragraph encoder, we had to choose a fixed number of tokens to process in each paragraph. We chose 128 as the number of tokens, since fewer than four percent of paragraphs in our data set contained more than 128 tokens. If a paragraph has fewer than 128 tokens, the sequence is padded with zeros as placeholder tokens; if a paragraph has more than 128 tokens, it is truncated. We experimented with two different neural network implementations as the paragraph encoder. The first is a recurrent neural

network, and the second is a convolutional neural network. Their specifics are described in Sections 4.6.1 and 4.6.2, respectively.

For our word embedding, we chose pre-trained word vectors trained by the GloVe [29] algorithm over a 6-billion word corpus[8]. We chose the highest-dimensional word embedding available (300), since vectors with higher numbers of dimensions contain more information, at the cost of computational efficiency.

The aggregation module is discussed in Section 4.6.3.

4.6.1 Paragraph Encoder—Recurrent Neural Network

A recurrent neural network used as the paragraph encoder is constructed as follows. The structure of this network is depicted in Figure 4.2 and its properties are listed in Table 4.3.

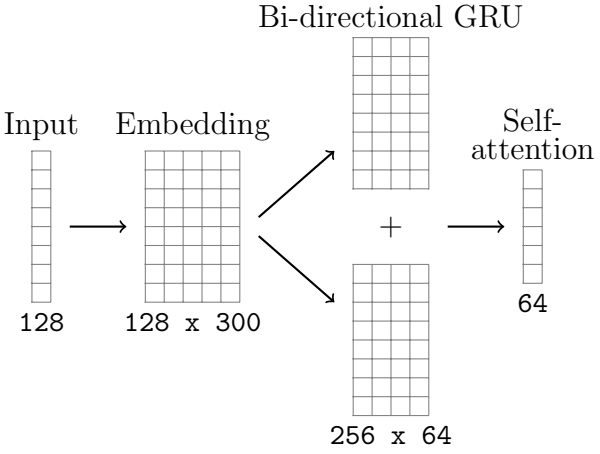


Figure 4.2: Output dimensions at each step of the Recurrent Neural Network paragraph encoder.

Layer	**Output Size**
Input	128
Embedding	128 x 300
Bi-directional GRU [hidden size=64]	256 x 64
Self-attention	64

Table 4.3: Properties of the Recurrent Neural Network paragraph encoder.

- **Input.** The input to the network is a sequence of 128 tokens representing a single paragraph.
- **Embedding Layer.** This layer will simply map each word token in the sequence to its corresponding vector representation in the 300-dimensional GloVe embedding. The

[8] https://nlp.stanford.edu/projects/glove/

result is a sequence of vectors each of length 300, or, a matrix with 128 rows and 300 columns.

- **Bi-directional GRU Layer.** This layer processes sequences of word tokens as produced by the embedding layer.

  The Gated Recurrent Unit (GRU) [7] is a specific implementation of the recurrent function f.

  Unlike a standard GRU, a *bi-directional* GRU additionally processes the input sequence in the backwards direction. Processing sequences of word vectors in both directions tends to lead to better performance [35] with the intuition that contextual clues in text may not always appear in the direction they are written.

  We chose 64 as the number of dimensions of each hidden state vector h_t, since it is a reasonably small power of 2 and vectors with sizes that are powers of 2 may be processed more efficiently by computer hardware. This hyper-parameter can be chosen arbitrarily with consideration to the fact that adding more dimensions may capture more latent features of tokens—which may improve accuracy overall—with the risk of over-fitting the model to the training data. The result is a sequence of $128 + 128 = 256$ vectors (128 from each of the forward and backward directions) each with 64 elements, or, in practice, a matrix with 256 rows and 64 columns.

- **Self-attention Layer.** The output from the bi-directional GRU layer is a sequence of vectors—essentially a matrix—which needs to be reduced to a vector in order to perform classification at the last step of the network. The self-attention layer learns a relative weighting function for all hidden state vectors produced by the bi-directional GRU—where more informative vectors are given more weight—and returns a weighted sum of those vectors. Its output is a single vector of size 64.

4.6.2 Paragraph Encoder—Convolutional Neural Network

Aside from using an RNN, we also experimented with a convolutional neural network as the implementation of the paragraph encoder. Figure 4.3 shows the structure of the network; Table 4.4 lists the configuration of each layer.

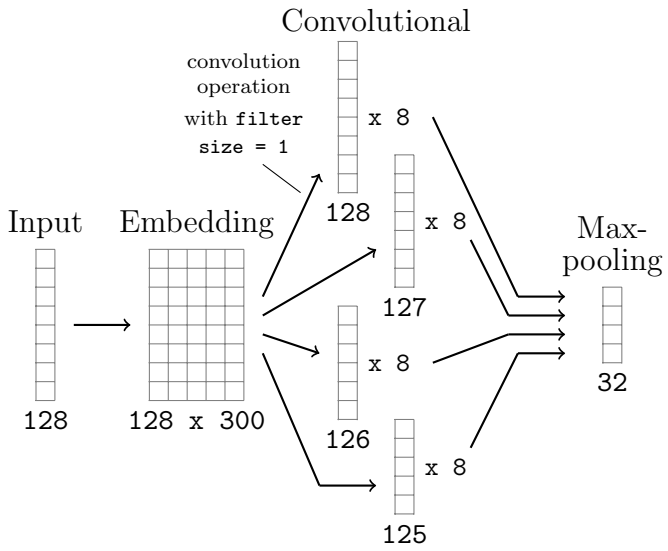


Figure 4.3: Output dimensions at each step of the Convolutional Neural Network paragraph encoder.

Layer	Output Size
Input	128
Embedding	128 x 300
Convolutional [filter sizes=1,2,3,4; filters=8]	128 (x 8), 127 (x 8), 126 (x 8), 125 (x 8)
Max-pooling	32

Table 4.4: Properties of the Convolutional Neural Network paragraph encoder.

- **Input.** The input to the network is a sequence of 128 tokens representing a single paragraph. (Identical to RNN implementation of the paragraph encoder.)
- **Embedding Layer.** Maps word tokens to word vectors in $\mathbb{R}^d$, $d = 300$. Functionally identical to that in RNN paragraph encoder. Output is a matrix with 128 rows and 300 columns.
- **Convolutional Layer.** The purpose of this layer is to extract important local sequences of word vectors.

  For this layer, we had to decide: (1) the number of filters, and (2) the size of the filters, i.e., the number of consecutive word vectors to be convolved. Because it is possible to choose variable filter sizes, we chose to use eight filters each of sizes 1, 2, 3, and 4.

These filter sizes correspond to convolving each word vector in isolation (size 1), as well as each consecutive group of 2, 3, and 4 word vectors (sizes 2, 3, and 4) in the entire sequence. The number of filters can be chosen arbitrarily with respect that larger numbers may tend to cause overfitting.

Recall that the output from a single convolutional filter with size r given an input sequence with length T is a vector of $T - r + 1$ elements. Because we process 128 tokens in each paragraph, or $T = 128$, our filter with size $r = 1$ will then produce a vector with $128 - 1 + 1 = 128$ elements. Thus, given our filter sizes of 1, 2, 3, and 4, the outputs from this layer are eight vectors with lengths 128, 127, 126, and 125.

- **Max-pooling Layer.** The convolutional layer produces a total of 32 vectors, and some vectors vary in the number of dimensions. This layer pools the maximum scalar value from each of the 32 vectors, producing a vector with 32 elements.

4.6.3 Aggregation Module

When predicting the maturity level of a particular book, we didn't want to exclude any paragraphs of that book (by random sampling of paragraphs or other means) because a high level of maturity in just one paragraph may strongly influence the maturity level of the entire book. Thus, we want our neural network to be able to draw information from every paragraph of a given book during prediction. Our method to do so is described here. Figure 4.4 and Table 4.5 show the structure of the network and its properties respectively.

- **Input Layer.** Unlike the paragraph encoder, which accepts only a single paragraph of a book, this module can take an arbitrary number of paragraphs as input. During training, we passed to this layer all paragraphs from each book as input.
- **Distributed Paragraph Encoder Layer.** Either implementation of the paragraph encoder outputs a fixed-length feature vector given a sequence of word tokens—a paragraph. This layer simply invokes the same paragraph encoder over every paragraph

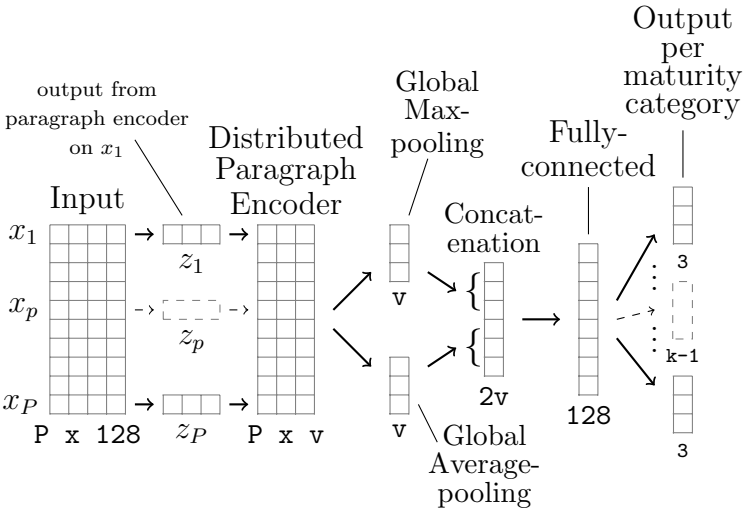


Figure 4.4: Output dimensions at each step of the Aggregation Module.

Layer	**Output Size**
Input	`P x 128`
Distributed Paragraph Encoder	`P x v`
Global Max-pooling	`v`
Global Average-pooling	`v`
Concatenation	`2v`
Fully-connected `[size=128]`	`128`
Output	`3, 3, 1, 3 3, 3, 3`

Table 4.5: Properties of the Aggregation Module.

of a given book, producing P fixed-length feature vectors where P is the number of paragraphs in that book. The length v of the feature vectors is 64 when the the implementation of the paragraph encoder is the RNN, and 32 when it is the CNN.

- **Global Max-pooling Layer.** Global max-pooling, or max-over-time pooling, in general, outputs the maximum value in each column of an input matrix, resulting in a vector. If we consider the input to this layer to be a matrix with P rows and v columns where v is the length of one vector produced by the paragraph encoder, then the output from this layer is a single vector of size v.

  The intuition behind global max-pooling is as follows: each vector produced by the paragraph encoder z_p is comprised of v elements where each element represents some latent feature of books for paragraph p. When a particular latent feature is expressed strongly in the paragraph (e.g., at the appearance of a profane word), then the i-th element in z_p, $1 \leq i \leq v$, which represents that feature will be a relatively large number. By taking the maximum values over every z_p for each i, we presumably capture the maximum level of expressiveness for each latent feature over the entire book.

- **Global Average-pooling Layer.** Similarly to global max-pooling, this layer outputs the average value over all columns of a given input matrix. This layer produces a vector with size v.

  We used this layer with the intuition that the maturity content rating for a book may be influenced not only by the most severe instance of mature content in that book, but by prolonged mature themes, i.e., ones than appear throughout the book.

  Both the global max-pooling and the global average-pooling layers are functional and, therefore, contain no trainable parameters.

- **Concatenation Layer.** We concatenate the two vectors which came from the output of the global max-pooling and global average-pooling layers to create one longer vector of size $2v$.

- **Fully-connected Layer.** A fully-connected, or dense, layer simply extracts more hidden, or latent, features from the concatenation layer. The result is a vector with 128 elements.

- **Output Layers.** The final layers of the network output the prediction of the maturity rating level as a binary ordinal vector (Section 4.2) for each maturity category.

Finally, the Adam optimization algorithm [3, 24], a commonly-used extension of stochastic gradient descent, was chosen to minimize the binary cross-entropy loss for the networks. Adam has been shown to perform well in comparison to other optimization algorithms [32].

5 EXPERIMENTAL RESULTS

In this chapter, we describe the results of two different experiments. In the first experiment, we attempted to classify the mature content of each book as a whole. The second experiment was an effort to evaluate whether our classifier would be able to identify the portions of the book that contained mature content.

For both experiments, we shuffled then divided the data set of books (6395 in total) into a train, validation, and test set using a 60/20/20 distribution, resulting in 3837, 1279, and 1279 books in their respective sets. We trained baseline and neural network models using the same train data set. The neural networks were evaluated on the validation set during each epoch of training to monitor their improvement over time. The neural networks quit training when the loss measured on the validation set stopped improving. In this section, we report the evaluation metrics of all classifiers on the test set.

In addition to the six baseline classifiers presented in Section 3.4, we compare the results of the neural network models to the 'Zero Rule', or Zero R, benchmark, which simply always predicts the majority class, ignoring all input values.

5.1 Classify Mature Content in Entire Books

We measured the mean squared error of the six baseline classifiers and our neural network models on the same randomly-sampled test set for each maturity category and the overall rating. The results after averaging the mean squared error over all maturity categories are shown in Figure 5.1. The results for each maturity category individually can be found in Appendix A.

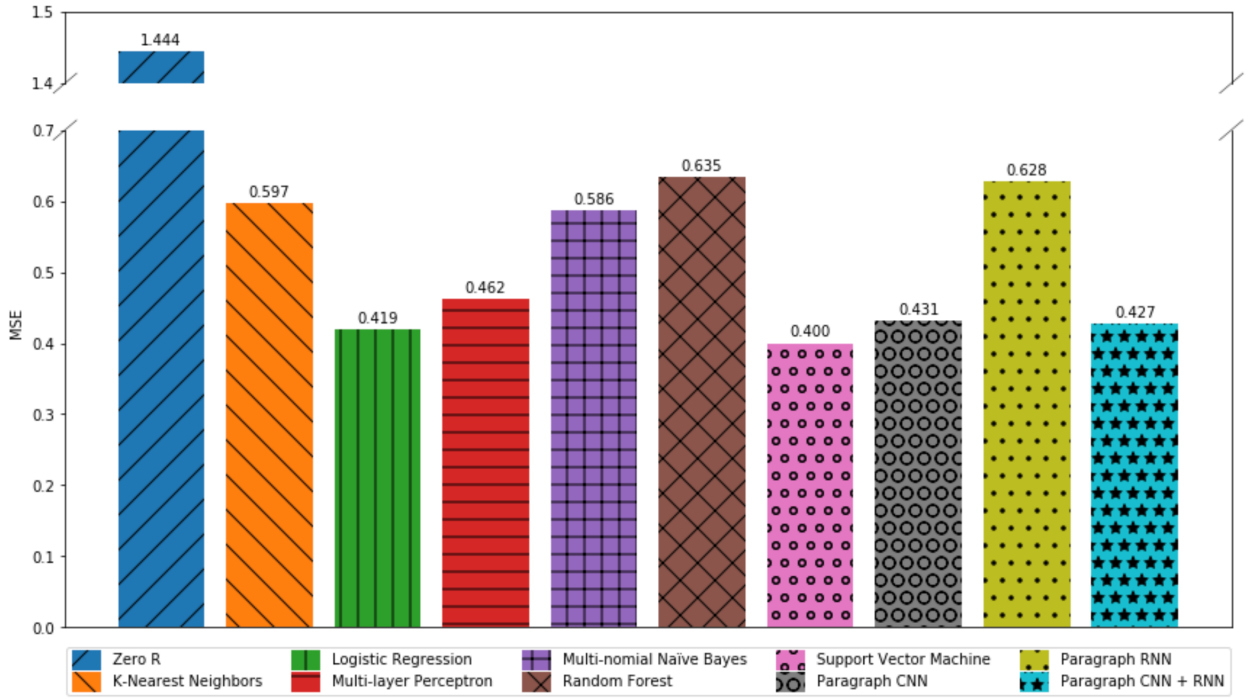


Figure 5.1: Mean squared error of content rating levels for entire books for all classifiers averaged over all maturity categories.

Our results show that the support vector machine outperforms each of our neural network methods in mean squared error when averaged for average classification accuracy (as the average of each classification accuracy score over all maturity categories) and average F1-score, respectively. We found that the results of the SVM were shown to be statistically significant by the Wilcoxon signed-rank test (all $p < 0.05$) when compared to each neural network model. We hypothesize that this occurred because the bag-of-words representation, which the SVM and the other baseline classifiers take as input, is actually quite dense and informative, since the number of words in an entire book is large relative to the size of the combined vocabulary of all books. These results suggest that, when classifying very long text sequences, the *order* of words may not be as informative as the overall distribution of words.

5.2 Detect Mature Content in Paragraphs

We also wanted to evaluate whether or not our classifiers would be able to *detect* the portions of books that contained mature content. Since both the baseline classifiers and the neural network models were trained to classify the levels of mature content in texts of variable size, it is valid to give as input to the classifiers only a portion of the book instead of the entire book. We hypothesized that each of our classifiers would also be able to compute a meaningful maturity content level for individual paragraphs and groups of paragraphs of books.

Since the labels that we collected pertained only to entire books, we were unable to definitively determine which paragraphs of those books contributed to their content rating levels, i.e., which paragraphs actually contained profanity, scenes of violence, sexuality, etc. Without ground-truth labels that specifically describe the mature content of paragraphs of books, it was impossible for us to train new classifiers to predict mature content within paragraphs. But, without training new classifiers, we were able to evaluate the effectiveness of the trained models from the previous experiment to detect mature content in portions of books via the following procedure.

For each book, we pulled out all ordered sequences of w paragraphs, called *paragraph windows*, for a chosen window size, w. Then, for each book, we passed as input to our classifiers each paragraph window of that book, resulting in a unique predicted content rating level for every paragraph window. We then evaluated the maximum predicted content rating level over all paragraph windows for each book against the ground-truth labels for those books. From the maximum predicted content rating levels for books, we were able to measure mean squared error and, thus, meaningfully compare the classifiers' abilities to detect mature content in smaller texts, i.e., paragraphs.

We feel that it is meaningful to aggregate the predictions of a given learning algorithm by the max function over the paragraph windows of a book with a given maturity content rating under the following two assumptions: (1) that book contains at least one scene, i.e.,

over the span of a few paragraphs, that contains content that is exactly as mature as the content rating for that book as a whole; and (2) that book contains no scenes that contain content that exceeds the maturity level of the book as a whole. For example, for a book that is rated *Mild* in Violence and Horror, we assume that it contains at least one mildly violent or horrific scene and contains no scenes that meet any of the *Moderate* or *Adult* criteria for this category.

Figure 5.2 shows the results of this experiment averaged over all maturity categories. Our neural network methods performed best when mean squared error scores for all categories are averaged, though other classifiers performed better within certain categories. The full results are shown in Appendix B. Further, the predictions of all three neural networks were shown to be statistically significant by the Wilcoxon signed-rank test (all $p < 0.05$) when compared to each baseline classifier.

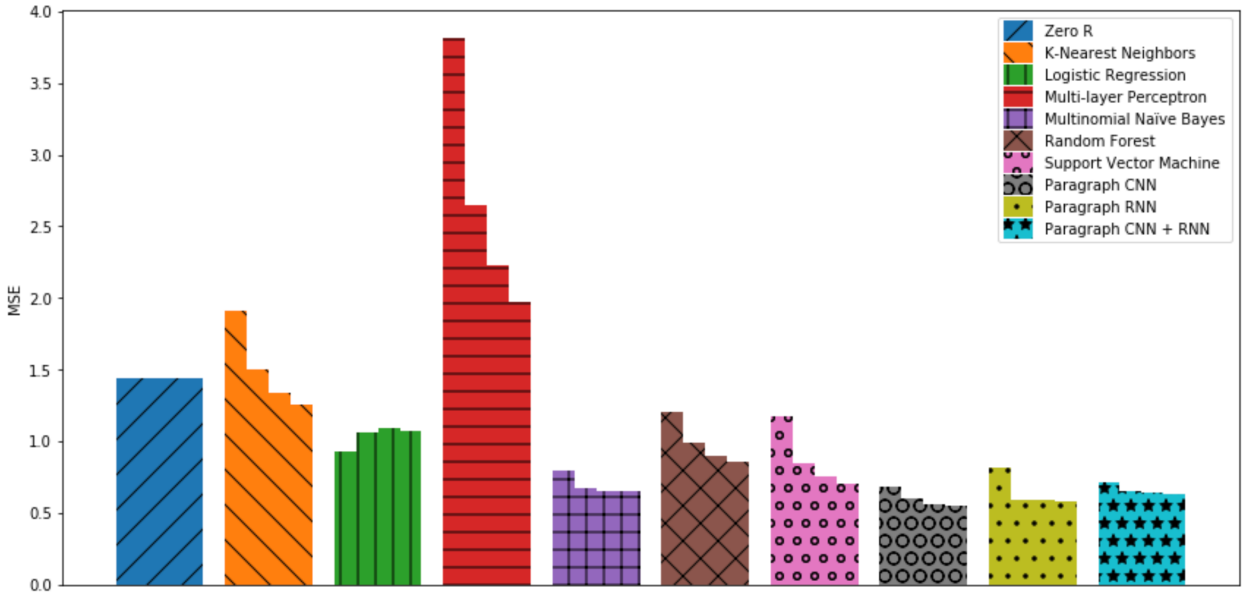


Figure 5.2: Mean squared error of content rating levels after classifier predictions are aggregated over paragraph windows of sizes 1, 3, 5, and 7 averaged over all categories.

We hypothesize that the neural network methods performed better on average than the baseline methods because the networks trained in two steps—by first encoding each paragraph of a given book, then aggregating an arbitrary number of encoded paragraph

vectors into an overall content rating prediction. The neural networks are able to capture more semantic meaning in portions of text that are far shorter than the book itself, whereas the baseline classifiers have no concept of portions of a text or of aggregation.

6 CONCLUSION

Although many forms of digital media are subject to content rating processes which inform consumers of any content that is against their standards or preferences, literature has never been able to call upon a robust system to rate its content. We feel that an automated system to deterministically assess the level at which mature content is present in literature would greatly benefit readers—especially children and parents. Since such a system is not widely available or standardized, we present our method as a way for readers or distributors to, for multiple maturity categories, (1) classify the maturity content level of an item of literature as a whole, and (2) detect the portions of that item of literature which contain mature content.

A real application of our method is that it could be implemented by Amazon for users of their massive digital library, Kindle Books[1], to optionally rate the content level of books that they rent or purchase. Other large eBook retailers which could implement our method include Apple Books[2], Kobo[3], Baker & Taylor[4], Barnes & Noble[5], and Google Play Books[6].

To the field of natural language processing, our work displays an interesting victory for neural network architectures in that they are actually able to reasonably detect semantic meaning in portions of text that are far smaller than the input texts when only labels for the input texts are present, i.e., even when ground-truth labels for the smaller portions of text are *not* available.

[1] `https://www.amazon.com/Kindle-eBooks`
[2] `https://www.apple.com/apple-books/`
[3] `https://www.kobo.com/`
[4] `https://www.baker-taylor.com/library.cfm`
[5] `https://www.barnesandnoble.com/b/nook-books`
[6] `https://play.google.com/store/books`

References

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[2] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146, 2017.

[3] Léon Bottou, Frank E Curtis, and Jorge Nocedal. Optimization methods for large-scale machine learning. *SIAM Review*, 60(2):223–311, 2018.

[4] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[5] Mark Callister, Sarah M Coyne, Lesa A Stern, Laura Stockdale, Malinda J Miller, and Brian M Wells. A content analysis of the prevalence and portrayal of sexual activity in adolescent literature. *Journal of Sex Research*, 49(5):477–486, 2012.

[6] Ying Chen, Heng Xu, Yilu Zhou, and Sencun Zhu. Is this app safe for children? A comparison study of maturity ratings on Android and iOS applications. In *Proceedings of the 22nd International World Wide Web Conference (WWW)*, pages 201–212, 2013.

[7] KyungHyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[8] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[9] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20 (3):273–297, 1995.

[10] Thomas Cover and Peter Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27, 1967.

[11] Sarah M Coyne, Mark Callister, and James C Phillips. Getting boozy in books: Substance use in adolescent literature. *Health Communication*, 26(6):512–515, 2011.

[12] Sarah M Coyne, Mark Callister, Talita Pruett, David A Nelson, Laura Stockdale, and Brian M Wells. A mean read: Aggression in adolescent English literature. *Journal of Children and Media*, 5(4):411–425, 2011.

[13] Sarah M Coyne, Mark Callister, Laura A Stockdale, David A Nelson, and Brian M Wells. A helluva read: Profanity in adolescent literature. *Mass Communication and Society*, 15 (3):360–383, 2012.

[14] Sahibsingh A Dudani. The distance-weighted k-nearest-neighbor rule. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-6(4):325–327, 1976.

[15] Christiane Fellbaum ed. *WordNet: An electronic lexical database*. MIT Press, 1998.

[16] Eibe Frank and Mark Hall. A simple approach to ordinal classification. In *European Conference on Machine Learning*, pages 145–156. Springer, 2001.

[17] Yoav Goldberg. Neural network methods for natural language processing. *Synthesis Lectures on Human Language Technologies*, 10(1):69, 2017.

[18] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[19] Bing Hu, Bin Liu, Neil Zhenqiang Gong, Deguang Kong, and Hongxia Jin. Protecting your children from inappropriate content in mobile apps: An automatic maturity rating framework. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 1111–1120. ACM, 2015.

[20] Michael I Jordan. Serial order: A parallel distributed processing approach, ICS Report 8604. *Institute for Cognitive Science, UCSD, La Jolla*, 1986.

[21] Ashraf M Kibriya, Eibe Frank, Bernhard Pfahringer, and Geoffrey Holmes. Multinomial naive bayes for text categorization revisited. In *Australasian Joint Conference on Artificial Intelligence*, pages 488–499. Springer, 2004.

[22] Jack Kiefer and Jacob Wolfowitz. Stochastic estimation of the maximum of a regression function. *The Annals of Mathematical Statistics*, 23(3):462–466, 1952.

[23] Yoon Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014.

[24] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[25] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1097–1105, 2012.

[26] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*, 2015.

[27] Christopher D Manning and Hinrich Schütze. *Foundations of statistical natural language processing*. MIT Press, 1999.

[28] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[29] Jeffrey Pennington, Richard Socher, and Christopher Manning. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.

[30] Herbert Robbins and Sutton Monro. A stochastic approximation method. *The Annals of Mathematical Statistics*, 22(3):400–407, 1951.

[31] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386, 1958.

[32] Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

[33] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. Technical report, UCSD La Jolla Inst for Cognitive Science, 1985.

[34] Gerard Salton and Michael J McGill. Introduction to modern information retrieval, 1986.

[35] Mike Schuster and Kuldip K Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681, 1997.

[36] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems 30*, pages 5998–6008, 2017.

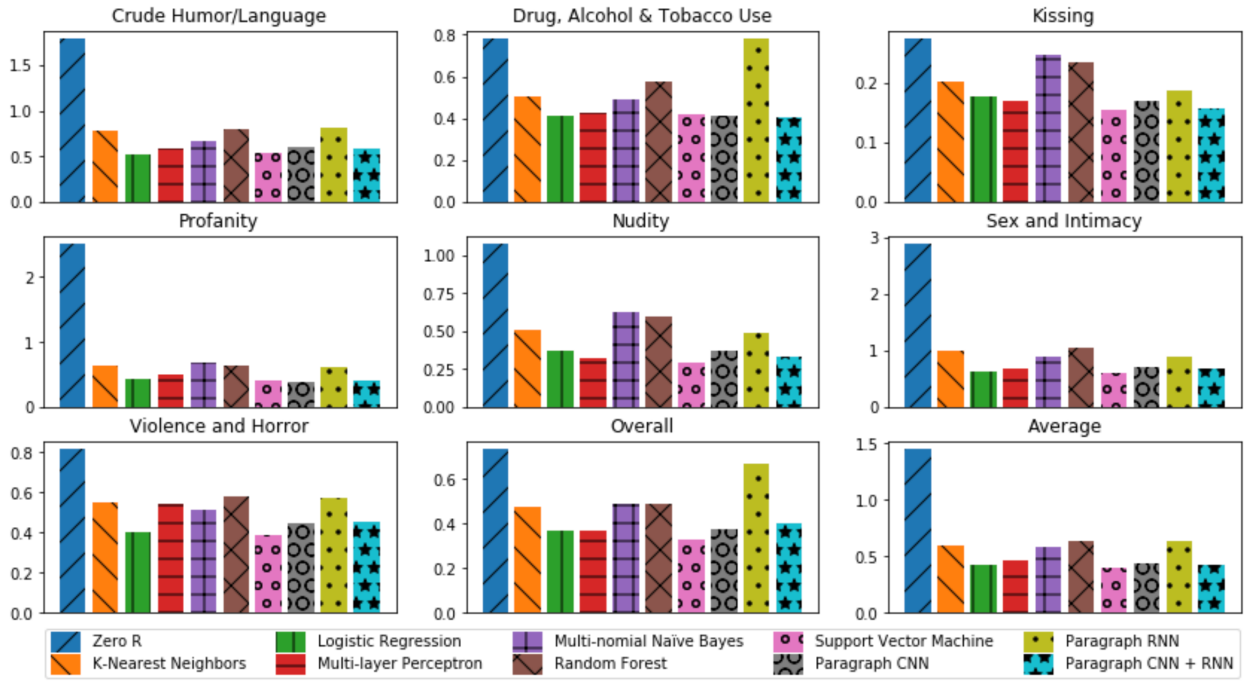
[37] Franz Wanner, Johannes Fuchs, Daniela Oelke, and Daniel A Keim. Are my children old enough to read these books? age suitability analysis. *POLIBITS*, 43:93–100, 2011.

[38] D Randall Wilson and Tony R Martinez. The general inefficiency of batch training for gradient descent learning. *Neural Networks*, 16(10):1429–1451, 2003.

[39] Eleanor Wood. Pushing the envelope: Exploring sexuality in teen literature. *Journal of Research on Libraries and Young Adults*, 1(1), 2010.

[40] Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 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, 2016.

A CLASSIFY MATURE CONTENT IN ENTIRE BOOKS - ALL RESULTS



	Zero R	K-Nearest Neighbors	Logistic Regression	Multi-layer Perceptron	Multi-nomial Naïve Bayes	Random Forest	Support Vector Machine	Paragraph CNN	Paragraph RNN	Paragraph CNN+RNN
Crude Humor/Language	1.7905	0.7826	0.5145	0.5817	0.6669	0.8014	0.5356	0.5966	0.8194	0.5848
Drug, Alcohol & Tobacco	0.7811	0.5059	0.4105	0.4245	0.4887	0.5739	0.4191	0.4120	0.7811	0.4019
Kissing	0.2737	0.2025	0.1759	0.1689	0.2471	0.2346	0.1532	0.1697	0.1861	0.1564
Profanity	2.5027	0.6427	0.4269	0.5067	0.6763	0.6263	0.4073	0.3855	0.6028	0.3987
Nudity	1.0719	0.5035	0.3690	0.3221	0.6231	0.5950	0.2948	0.3675	0.4918	0.3284
Sex and Intimacy	2.8780	0.9930	0.6357	0.6865	0.8874	1.0367	0.6075	0.6935	0.8984	0.6927
Violence and Horror	0.8124	0.5496	0.4027	0.5403	0.5145	0.5754	0.3823	0.4480	0.5731	0.4504
Overall	0.7334	0.4754	0.3683	0.3667	0.4887	0.4863	0.3299	0.3737	0.6724	0.4034
Average	1.4443	0.5971	0.4193	0.4615	0.5863	0.6348	0.4000	0.4308	0.6281	0.4271

Figure/Table A.1: Mean squared error scores of all classifiers for predicted content rating levels of entire books. All maturity categories are shown. Underlines denote the lowest score within each maturity category. Classifiers were trained using the same training set and were evaluated on the same test set.

B DETECT MATURE CONTENT IN PARAGRAPHS - ALL RESULTS

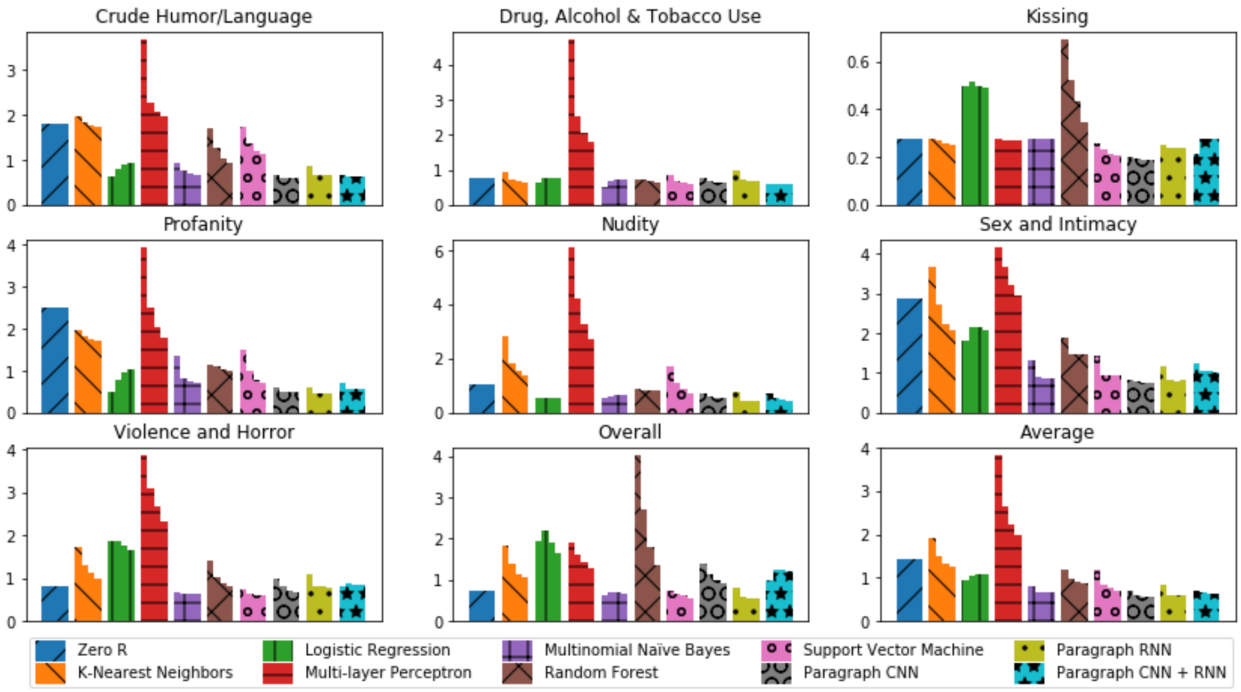


Figure B.1: Mean squared error of content rating levels after classifier predictions are aggregated over paragraph windows of sizes 1, 3, 5, and 7, respectively, for each individual category.

	Window Size	Zero R	K-Nearest Neighbors	Logistic Regression	Multi-layer Perceptron	Multinomial Nave Bayes	Random Forest	Support Vector Machine	Paragraph CNN	Paragraph RNN	Paragraph CNN + RNN
Crude Humor/ Language	1	1.7905	1.9679	0.6435	3.6841	0.9437	1.6896	1.7303	0.6693	0.8600	0.6755
	3	1.7905	1.8554	0.7928	2.2674	0.7584	1.2674	1.3636	0.6091	0.6708	0.6278
	5	1.7905	1.7881	0.9077	2.0680	0.6919	1.0242	1.2174	0.5934	0.6701	0.6450
	7	1.7905	1.7443	0.9359	1.9633	0.6810	0.9234	1.1181	0.5841	0.6661	0.6349
Drug, Alcohol & Tobacco	1	0.7811	0.9398	0.6294	4.7193	0.4996	0.7084	0.8303	0.7678	0.9969	0.5919
	3	0.7811	0.7209	0.7576	2.5199	0.6779	0.7084	0.6888	0.6880	0.7123	0.6114
	5	0.7811	0.6638	0.7670	2.0649	0.7131	0.6583	0.6388	0.6450	0.6927	0.6005
	7	0.7811	0.6294	0.7694	1.7811	0.7334	0.6474	0.5958	0.6278	0.6880	0.6044
Kissing	1	0.2737	0.2737	0.4941	0.2737	0.2737	0.6919	0.2588	0.2017	0.2494	0.2103
	3	0.2737	0.2674	0.5137	0.2729	0.2737	0.5246	0.2306	0.1916	0.2400	0.2768
	5	0.2737	0.2588	0.4988	0.2729	0.2737	0.4324	0.2119	0.1892	0.2392	0.2791
	7	0.2737	0.2510	0.4934	0.2713	0.2737	0.3487	0.2080	0.1869	0.2361	0.2776
Profanity	1	2.5027	1.9797	0.4996	3.9226	1.3667	1.1306	1.4918	0.5966	0.6099	0.7279
	3	2.5027	1.8264	0.7873	2.5113	0.8233	1.0923	0.9875	0.5035	0.4738	0.5833
	5	2.5027	1.7615	0.9719	2.0430	0.7475	1.0266	0.7787	0.4902	0.4762	0.5723
	7	2.5027	1.7123	1.0305	1.7998	0.6982	1.0125	0.7021	0.4840	0.4793	0.5708
Nudity	1	1.0719	2.8084	0.5270	6.0938	0.5301	0.8882	1.7334	0.7076	0.7514	0.7256
	3	1.0719	1.8163	0.5332	4.2088	0.6044	0.8256	1.0868	0.6036	0.4418	0.5364
	5	1.0719	1.5238	0.5489	3.2713	0.6450	0.8124	0.8765	0.5723	0.4285	0.4793
	7	1.0719	1.3745	0.5575	2.6974	0.6833	0.8311	0.7365	0.5528	0.4230	0.4550
Sex and Intimacy	1	2.8780	3.6849	1.8194	4.1509	1.3135	1.8780	1.4527	0.8186	1.1751	1.2314
	3	2.8780	2.7389	2.1579	3.6794	0.9218	1.4785	0.9476	0.7959	0.8210	1.0672
	5	2.8780	2.2424	2.1618	3.2299	0.8757	1.4793	0.9335	0.7615	0.8077	1.0414
	7	2.8780	2.0758	2.0876	2.9398	0.8858	1.4644	0.9343	0.7412	0.8108	1.0367
Violence and Horror	1	0.8124	1.7303	1.8577	3.8491	0.6638	1.4183	0.7365	0.9984	1.1032	0.8217
	3	0.8124	1.2869	1.8686	3.1032	0.6505	1.0133	0.6458	0.7983	0.7944	0.8679
	5	0.8124	1.1181	1.7686	2.6568	0.6458	0.8647	0.6106	0.6943	0.7920	0.8608
	7	0.8124	0.9844	1.6419	2.3206	0.6364	0.8014	0.6075	0.6575	0.7647	0.8327
Overall	1	0.7334	1.8170	1.9593	1.9132	0.6388	4.0141	0.7498	1.3980	0.8241	0.9758
	3	0.7334	1.3831	2.1986	1.6263	0.6794	2.6990	0.6466	1.1282	0.5833	1.2346
	5	0.7334	1.1423	1.9171	1.4277	0.6888	1.8061	0.6169	0.9922	0.5629	1.2447
	7	0.7334	1.0790	1.6599	1.2776	0.6755	1.3698	0.5575	0.9148	0.5614	1.1931
Average	1	1.4443	1.9121	0.9244	3.8134	0.7987	1.2007	1.1763	0.6800	0.8208	0.7120
	3	1.4443	1.5017	1.0587	2.6518	0.6729	0.9872	0.8501	0.5986	0.5934	0.6530
	5	1.4443	1.3366	1.0892	2.2295	0.6561	0.8997	0.7525	0.5637	0.5866	0.6398
	7	1.4443	1.2531	1.0737	1.9676	0.6560	0.8613	0.7003	0.5478	0.5811	0.6303

Table B.1: Mean squared error scores for all classifiers and all categories after content rating level predictions are aggregated over paragraph windows of sizes 1, 3, 5, and 7, respectively. Underlines denote the lowest score within each maturity category. Classifiers were trained using the same training set and were evaluated on the same test set.