Linguistic Features for Readability Assessment

by
Tovly Deutsch

presented to the Department of Computer Science
and the Department of Linguistics
in partial fulfillment of the requirements
for the degree with honors
of Bachelor of Arts

Harvard College
March 2020

Abstract

Readability assessment aims to automatically classify text by the level appropriate for learning readers. Traditional approaches to this task utilize a large variety of linguistically motivated features paired with simple machine learning models. More recent methods have improved performance by discarding these features and utilizing deep learning models. This thesis attempts to combine these two approaches with the goal of improving overall model performance. My primary method involves incorporating the output of a deep learning model as a feature itself, used in conjunction with linguistic features. Evaluating on two large readability corpora, I find that this fused approach is ineffective, failing to improve upon state-of-the-art performance. These results suggest that other avenues of research would be more fruitful in improving readability assessment.

Acknowledgements

I would like to thank my thesis advisors Professor Stuart Shieber and Masoud Jasbi for guiding my research in useful directions. I would also like to thank Professor Finale Doshi-Velez for agreeing to read my thesis. I sincerely appreciate the support and feedback provided by Dorothy Ahn and Gunnar Lund in their linguistics thesis tutorials. Finally, I am indebted to my family and friends for their support over the past year.

Contents

Chapter 1

Introduction

Literacy is a crucial skill in the modern economy yet many populations, even in developed nations, struggle with poor literacy (OECD 2016). A key component of developing literacy is simply reading; however, selecting appropriate materials for the reading development of a learner is not a straightforward task. For this process a variety of competing heuristics, methodologies, and systems have been developed (Flesch 1948; Kincaid et al. 1975; Reutzel et al. 2008; D. R. Smith 1989). These processes have significant active use being codified into legislation (*Readable Language in Insurance Policies.* 1982) and educational guidelines (National Governors Association Center for Best Practices and Council of Chief State School Officers 2010). These processes generally take as input some text and output a label, such as a grade level, indicating the appropriate audience for that text. Some involve manual human annotation which poses challenges. Human annotation is time consuming and expensive. Limited educational funding, the immensity of published work, a desire for multiple annotations, and a need for qualified annotators all hinder this method. Beyond financial and temporal constraints, human annotation contains substantial bias. Differing backgrounds between the annotator and reader may diminish annotation accuracy. The annotator may also struggle to inhabit the minds of readers at different stages of learning. Finally, individual annotator inconsistency and disagreement between annotators diminishes label reliability. The issues of annotator qualification could be mitigated by utilizing annotators with a variety of reading levels, e.g. students. However, this approach requires a larger number of annotators, likely inflating time and cost. Additionally, other problems like differing backgrounds remain. Due to these challenges, automated methods for determining readability are used as a compelling alternative.

Stretching back to at least the 1940s (Flesch 1948), researchers have devised systematic and automated methods for measuring a text's readability. They began with simplistic formulas based on statistical counts like word counts and sentence count; such counts or analyses of an input are termed "features". With the rise of machine learning, researchers began incorporating more complex models and linguistic features like age of acquisition and word concreteness. Following trends in natural language processing, the most recent state-of-the-art methods dispense with manually chosen features and utilize deep neural networks (Martinc et al. 2019). Despite impressive results in assessing readability, such models still diverge substantially from perfect performance. In many applications of machine learning, performance can be improved by increasing the amount of training data. Indeed, in a variety of tasks state-of-the-art models are trained on millions of examples. However, gathering readability data on this scale is extremely difficult due to the aforementioned challenges of cost, time, and annotator qualification. Thus, other innovations are necessary to improve the performance of readability assessment.

Given the history of using linguistic features for readability assessment, I hypothesized that these features could be combined with modern deep learning methods for performance improvements. Deep learning methods automatically extract relevant features thereby diminishing the need for manual feature engineering. However, this premise may only hold given sufficient data and model complexity. Thus, given the lack of readability data and non-optimal model structures, linguistic features may fill in the feature gaps that deep learning models are unable to learn.

To test this hypothesis, I utilized two of the largest and most popular readability corpora, WeeBit and Newsela. They contain documents intended for learning readers and second language learners labeled by reading difficulty. I trained and evaluated a variety of models on these corpora, including model types that produced state-of-the-art results on readability corpora in previous literature. Some of these models involved both deep neural networks and linguistic features; the linguistic features were collected from a variety of existing sources along with novel features I introduce in this thesis. Such methods failed to improve model performance beyond state-of-the-art results. These results suggest that deep learning methods have progressed to the point that manual feature engineering is no longer useful for readability assessment. Instead, other avenues of research should be investigated in pursuit of improving readability assessment. Indeed, given the vague construction methodology of readability corpora, particular attention should be paid to the validity and consistency of such corpora.

This thesis begins with an exploration of existing work on readability assessment in chapter 2. Chapter 3 details the techniques used to construct, train, and evaluate the readability assessment models used in this thesis. Evaluations of these models on each corpus are presented in chapter 4, along with an analysis of the predictive power of individual linguistic features. Chapter 5 concludes with a discussion on the effects of linguistic features and provides suggestions for future work on the topic.

Chapter 2

Background

Readability assessment research has undertaken many approaches to assess document complexity. Across these approaches, the input and output of the task have generally followed the same paradigm. The input is text and the output is some measure of how difficult it would be for a human to read that text. To complete this task, researchers have often utilized the methods of machine learning. Broadly, this approach involves gathering data, computing features on that data, and using those features to train a model. In this chapter I will outline work done on these three core components, corpora, features, and models, as they relate to readability research.

2.1 Corpora

One significant challenge in the task of readability assessment, as with many other machine learning tasks, is a lack of data and the low quality of data. With readability assessment in particular, data acquisition can be costly, time consuming, and inconsistent. For a human to produce a high-quality readability label they must read an entire document and be knowledgeable about reading development and ability. Additionally, labels are likely to be inconsistent across annotators. The need for reading development knowledge can be eliminated by using developing readers as annotators. However, this approach requires a greater number of annotators to cover different reading levels. By contrast, in other tasks such as image classification labels can be produced quickly and accurately by nearly any individual. For this reason, readability corpora are relatively small in comparison to other machine learning and natural language processing tasks. One breakthrough in readability corpora came with the introduction of the WeeBit corpus by Vajjala and Meurers (2012). It is composed of about 6,000 articles from two educational periodicals which separated their articles by grade level. According to Vajjala and Meurers (2012), the methodology by which the periodicals separated articles into grade levels is not publicly known. Presumably, the appropriate grade level was determined by some combination of editor assessment and author consideration.

Another significant corpus published recently is the Newsela corpus (Xu et al. 2015) which attempts to provide texts of similar topic but varying difficulty. It consists of sets of texts where each document in the set covers the same material but in up to 5 different levels of difficulty. Each document set was constructed by having authors rewrite a base document at specific levels of difficulty. This type of corpus allows researchers to isolate surface level linguistic complexity from the difficulty of the concepts being conveyed.

2.2 Features

Researchers have constructed a variety of features that attempt to capture different aspects of document complexity. A feature is some data derived from an input; generally, a feature is a numeric summary of the input, e.g. number of words and average sentence length. The number of linguistic features utilized is large (Vajjala Balakrishna (2015) used 151 features), therefore in this section I will describe the general categories of features and some prominent examples. I will also explain the concept of word vectors which can be thought of as features that attempt to capture the entirety of the input rather than relying on summarization.

2.2.1 Linguistic Features

A useful guiding framework is the one developed by Vajjala Balakrishna (2015) and Vajjala and Meurers (2012) who construct categories such as traditional features, lexical and POS features, syntactic features, and word characteristic features.

Traditional features are the most basic features used and are termed traditional for their use in traditional feature formulas such as the Flesch formula (Flesch 1948). They include the number of characters, number of words, number of sentences, and the number of syllables in a document. Additionally, Vajjala and Meurers (2012) include various formulas based on these features, such as the Flesch-Kincaid score (Kincaid et al. 1975) and Coleman-Liau formula (Coleman and Liau 1975), as features themselves.

A more complex class of features are the lexical features which attempt to capture the lexical diversity or "richness" of a text. Most prominently, they include type token ratios, ratios of the number of word types to the number of word tokens (Lu 2011). Word type in this context means unique word. Thus, repeated instances of a word would not increase the number of word types but would increase the number of word tokens. As an extreme elucidating example, a document consisting of only one repeated word would have a word type count of 1 and a large word token count; thus, the document would a have low type token ratio which implies a low lexical diversity. Additional lexical features include such measures as the ratio of lexical words to words, ratios between part of speech counts and lexical words, and ratios between part of speech counts and sentence counts.

In addition to assessing author word choice, researchers have attempted to characterize the syntactic complexity of documents largely via constructing measurements of syntactic trees. At the most basic level, one can extend the POS count ratios to phrase count ratios. For instance, Schwarm and Ostendorf (2005) constructed ratios of noun and verb phrase counts to sentence counts. Similarly, Feng (2010) considered the average length of those phrases. Lu (2010) constructed a variety of features based on counts of phrases, clauses, and *t-units* which are defined as a clause and any of its embedded structures such as subordinate clauses; these features were shown to correlate with the development of second language acquisition in writing samples. Zooming out to the syntactic tree as a whole, Schwarm and Ostendorf (2005) also considered average parse tree height.

In a similar fashion to lexical and POS features, Vajjala Balakrishna (2015) constructed a variety of features based on the morphological, syntactic, and semantic properties of words. Using the Celex Lexical Database, they constructed counts of a variety of word categories such as foreign words, words containing an affix, transitive verbs, and countable nouns. While the syntactic features associated with this category may have some overlap with the syntactic complexity features, they are properties not available via syntactic parses or POS tags alone. In addition to morpho-syntactic and semantic properties, Vajjala Balakrishna (2015) characterized the lexical ambiguity of words by calculating their average number of senses. Finally, they drew from psycholinguistic databases to calculate measures of word complexity such as average word concreteness, familiarity, and age of acquisition.

The lexical richness and syntactic complexity features appear to both be especially predictive of readability. Vajjala and Meurers (2012) found that these two categories of features produced comparable accuracy scores when fed into an SVM for readability classification. Furthermore, the accuracy improved substantially, by about 10 percentage points, when the two feature sets were combined. This result indicates that some predictive information is not shared between the two feature sets. Within these feature sets some features are substantially more predictive than others. For instance, Vajjala Balakrishna (2015) found that some features had model weights three orders of magnitude lower than others making their impact negligible. Some of the most predictive features were found to be the average age of acquisition of lemmas, the number of prepositional phrases per sentence, and average syntactic parse tree height.

2.2.2 Word Vectors

While linguistic features summarize the input text, word vectors attempt to avoid summary by encoding the input words themselves as features. The most basic approach to this encoding is the one-hot or 1-of-V approach. For a 1-of-V encoding, given a vocabulary V, each word is represented by a vector of size V with one particular index corresponding to the word set to 1 and all other indices set to 0. This approach, while simple, fails to encode any information about the word's usage and meaning. Additionally, the vectors are all equidistant from one another; thus, given two distinct vectors nothing meaningful can be inferred about their relationship.

A more complex and rich approach to word vectors involves learned word embeddings. In this approach, a language model is trained to either predict a "middle" word given some surrounding context words, known as "continuous bag-of-words", or to predict surrounding context words given a middle word, known as "continuous skip-gram" (Mikolov et al. 2013). The input word vectors are encoded using the 1-of-V method and then fed into a 2 layer neural network of a projection layer and output layer. The projection layer produces a vector of length D, where D is the desired word embedding length, which the output layer then uses to produce a probability distribution over the vocabulary; the most probable word in the output vector can then be chosen as the output word. Because the projection layer transforms $V \times 1$ vectors into $1 \times D$ vectors, its weight matrix is of size $V \times D$ and thus each row of the weight matrix can be taken as an embedding of the word corresponding with that row's index in the 1-of-V encoding. These methods allow word vectors to be learned efficiently while producing embeddings that can be used for tasks with competitive performance compared to other word embedding methods (Mikolov et al. 2013). Additionally, word vectors encode semantic relationships and analogies. Specifically, algebraic combinations of these word vectors often describe relationships between words. Let us define a function vec(word) that takes in a word in text form and outputs its corresponding word vector. It is likely to be the case that vec(dog) + vec(wild) $\approx$ vec(wolf) and that vec(Paris) - vec(France) + vec(Italy) $\approx$

vec(Rome). Intuitively, it seems that vec(Paris) - vec(France) should result in some notion of capitals in general. Then, by adding vec(Italy), we have added specific information about a country and thus the concepts of capital and Italy fuse to specify the capital of Italy, Rome.

2.3 Models

Once features are selected, they are fed into a statistical model. This model analyzes these features and produces an inference relevant to the task; in the case of readability this inference would be a reading level. In this section, I will describe some of the models popularly used in readability assessment. I will begin with an introduction to the processes of supervised learning and go on to explain linear models, SVMs, neural networks, and deep learning.

2.3.1 An Introduction to Supervised Learning

Generally, machine learning describes applied statistical methods for learning decision functions. One approach often studied in machine learning is *supervised learning.* Supervised learning involves learning some *labeling function* from training data with given ("ground-truth") labels. In the context of readability, this labeling function would take text as input and output an inferred label indicating how difficult that text would be to read. Models often compute features, such as those described in section 2.2, on the text as part of the inference process. This approach is termed "supervised" because the data that the model is trained on is pre-labeled by humans, e.g. with readability scores. Thus, humans are in a sense "supervising" the training of the model. When the model is fed new, previously unobserved input it must infer the input's label given the information it learned in the training stage.

Throughout this section, $\boldsymbol{x}$ refers to a vector composed of features. Similarly, $\boldsymbol{w}$ refers to a vector of model weights. $f(\boldsymbol{x})$ refers to the inferred label produced by a model while y refers to the ground-truth label of an input. For both $\boldsymbol{x}_i$ and y_i, the index indicates that the input or label is from example i in the dataset.

2.3.2 Linear models

One of the simplest model types used for inference is the *linear model*. A linear model computes the dot product of an input feature vector $\boldsymbol{x}$ and weight vector $\boldsymbol{w}$ to produce a label $f(\boldsymbol{x})$. A constant bias b can also be added to the results to shift the range of output labels. For notational simplicity, the bias factor b can be incorporated into the $\boldsymbol{w}$ vector by appending a constant value to $\boldsymbol{x}$, an approach taken in this thesis. This linear model calculation is shown in equation 2.1. In the context of readability, a reasonable labeling scheme could associate a larger $f(\boldsymbol{x})$ with a more difficult document.

$$f(\boldsymbol{x}) = \boldsymbol{x} \cdot \boldsymbol{w} \tag{2.1}$$

The weights of $\boldsymbol{w}$ are learned in the training of the model. To train a model, first a *loss function* is defined which measures how well the model is performing. A simple loss function, known as the absolute error function, measures the absolute difference between the true label of an input and the label inferred by the model. If the model produces the correct label, then its loss is 0; otherwise, the loss increases as the inference drifts further from the true label. Once this loss function is defined, a variety of optimization methods can be used to modify the model weights to reduce this error during the training stage.

2.3.3 Shallow Feature Formulas

The earliest research into readability assessment utilized linear models. It began with the extraction of simple, later termed "shallow" and "traditional", document features. These features are multiplied by manually chosen weights and summed to produce difficulty scores. This methodology began with Flesch (1948) who used syllables per word and words per sentence to construct the Flesch Reading Ease formula presented in equation 2.2. In the samples chosen by Flesch, the score tended to range from 0 (extremely difficult to read) to 100 (readable by those who are "barely functionally literate").

$$RE = 206.835 - 0.846(\frac{\text{number of syllables}}{\text{number of words}}) - 1.015(\frac{\text{number of words}}{\text{number of sentences}}) \qquad (2.2)$$

It is not clear how Flesch chose the weights but one can speculate it was likely some combination of correlation analyses, manual experimentation, and intuition. The formula is equivalent to a linear model (defined in section 2.3.2) albeit with the weights chosen manually rather than through modern computational optimization methods. On the corpus selected by Flesch, *Standard test lessons in reading* (McCall and Crabbs 1926), the Flesch reading formula achieved a correlation with estimated grade level of 0.7407. No mention was made of other evaluative metrics for regression, like absolute error, presumably because the scale of the formula was different from the scale of the grade levels labeled in the corpus.

The Flesch Reading Ease Formula saw some usage in non-academic settings. For instance, it was used as a metric in a Florida law mandating a minimum level of readability for insurance policies (*Readable Language in Insurance Policies.* 1982). The formula, along with other readability formulas like the Fog Index (Gunning 1952) and Automated Readability Index (E. A. Smith and Senter 1967), was modified in 1975 by Kincaid et al. (1975) to produce scores comparable to US Grade levels for easier interpretation. This paradigm of measuring readability using grade levels is reused in much of the later literature. This measure is likely popular not only because of its interpretability but also because most existing readability datasets are labeled with grade levels.

2.3.4 SVM

Beyond the basic approach of a linear model, researchers have used more complex models for assessing readability. This additional model complexity was motivated by the hypothesis, and eventual empirical validation, that it could improve performance. Starting with Schwarm and Ostendorf (2005), support vector machines (SVMs) were used for readability assessment because of their demonstrated effectiveness with other classification tasks (Joachims 1998).

In its most basic form, an SVM (Cortes and Vapnik 1995) is a binary classifier and linear model trained with a specific loss function, the hinge loss, that is designed to allow the model to generalize well on new data. The hinge loss L_h (equation 2.3) is defined as the maximum of 0 and $1 - y \cdot f(\boldsymbol{x})$. When the true and predicted label are identical, the hinge loss is 0. When the labels disagree in sign, the hinge loss increases as the difference between the labels increase. Thus, the hinge loss is bounded below at 0, indicating a perfect prediction, and unbounded above indicating any degree of error.

$$L_h(y, f(\boldsymbol{x})) = max(0, 1 - y \cdot f(\boldsymbol{x})) \tag{2.3}$$

The overall SVM loss function to be minimized, $J(w)$, is the total hinge loss (L_h) plus the L2 norm of the separator (weighted by hyperparameter λ), as presented in equation 2.4. This L2 norm is added as a form of *regularization*, a technique to reduce model complexity and thus decrease a model's tendency to overfit on training data. Thus, increasing lambda tends to increase the model's generalizability. However, if lambda becomes too large the model can become overly simple and make poor predictions, e.g. ignoring the input and outputting a near-constant output for the sake of simplicity.

$$J(w) = \lambda ||w||^2 + \frac{1}{n} \sum_{i=1}^{n} L_h(y_i, f(\boldsymbol{x}_i)) \tag{2.4}$$

Although the basic formulation of an SVM is similar to a linear model, a common technique when using an SVM is to apply a transformation or kernel to the data. This kernel often transforms the data into a higher dimensional space that a linear model would be unable to separate otherwise. Similarly, while the formulation of an SVM is framed as a binary classifier, it can be extended to more than two classes via a variety of methods. For instance,

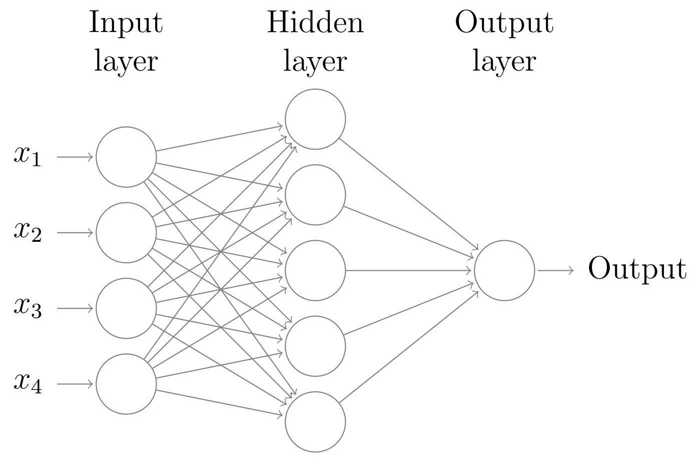

Figure 2.1: A basic neural network (MLP). Each arrow represents a weight multiplication, bias addition, and application of a nonlinear function. The output need not be a singular node. Figure is derived from Fauske (2006).

one technique involves training a binary classifier for each possible class and selecting the classifier with the largest output as the final class.

2.3.5 Multi-Layer Perceptron

Another model popularly used in machine learning is the multi-layer perceptron, the most basic form of a neural network. It has seen some limited success in readability assessment, most notably in the classification work of Vajjala and Meurers (2012) in which it outperformed other models such as SVMs, decision trees, and logistic regression.

A neural network is a directed acyclic graph (DAG) in which the edges between nodes represents a weight multiplication, bias addition, and application of a nonlinear function. A basic example of such a graph is presented in figure 2.1. The weight multiplication and bias addition are identical to the techniques described previously for a linear model. Where neural networks differentiate themselves is in the addition of nonlinear functions. In an MLP, the input vector is fed into the input layer, which is then connected to one or more hidden layers, which ultimately lead to the output layer. The basic model of neural networks has been extended in a variety of ways some of which are described in section 2.3.6.

2.3.6 Deep Learning

Recently, the best performance on a variety of natural language processing tasks has been achieved by deep learning methods (Devlin et al. 2019). These methods involve complex and multilayered neural networks usually paired with large amounts of data. In this paradigm, the model's inputs are learned mathematical representations of the text, such as word vectors, rather than manually selected linguistic features. This approach is motivated by a hypothesis that complex data-rich models may be able to automatically extract relevant features. If this hypothesis is true, manual feature selection would be redundant and perhaps detrimental to performance. Empirically, this hypothesis is supported by state-of-the-art performance for many "feature-less" approaches (Martinc et al. 2019).

CNN

One class of neural networks used in deep learning are convolutional neural networks (CNN). CNNs perform well on a variety of NLP tasks including text classification (Kim 2014); in an NLP context, CNNs often take as input pretrained word vectors (described in section 2.2.2). CNNs operate by applying convolutions, windowed weighted filters, over the input. A convolution involves a fixed size vector, known as a filter or kernel, that is dotted with some subsection of the input and then shifted to another section of the input. An instance of this dot product as part of a convolution is shown in figure 2.2. These dot products are then concatenated to produce the overall result of the convolution. The motivation behind this approach is that each filter can be trained to identify some local or specific aspect of the input. In an image this property might be an edge, shape, or texture. In NLP, these qualities are more abstract but could perhaps be aspects of semantics or syntax, e.g. dependent clauses or referents. In addition to convolutions, CNNs often employ max-pooling layers, layers that select the largest value within the input as the output. Placed after a convolution layer, a max-pooling layer has the effect of further reducing dimensionality and selecting the most prominent elements that the convolution has filtered for.

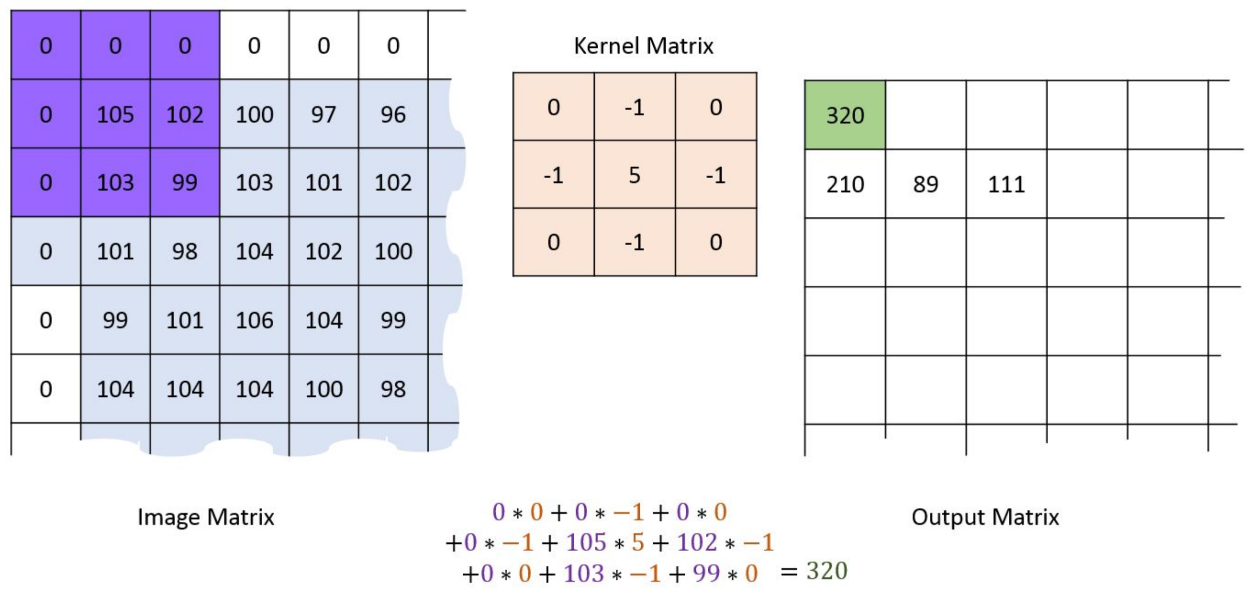

Figure 2.2: A demonstration of one convolution of a 3x3 filter, from Naidu (2020)
.

Attention-Based Models

Aside from using separately pretrained word vectors, other deep learning approaches have been applied to NLP tasks. Recently, especially successful approaches have used a mechanism known as attention. Fundamentally, attention mechanisms attempt to influence the impact each portion of the model input has on producing the output by weighting ("attending to") different portions of the vectors produced from the input. One deep learning approach utilizing attention mechanisms is the transformer (Vaswani et al. 2017), an architecture that has produced state-of-the-art results on a variety of NLP tasks including readability assessment on the WeeBit corpus (Martinc et al. 2019). These transformers are often pretrained on a large generic corpus of text and then trained again ("fine-tuned") for a specific data-poor task like readability assessment.

Another attention-based approach used for readability assessment is the hierarchical attention network (HAN) proposed by Yang et al. (2016). This architecture attempts to mimic the structure of document composition by constructing two layers of attention mechanisms. One layer attends to different words within a sentence while another attends to different sentences within a document. This formal representation of document structure has led to improved document classification results including in state-of-the-art readability classification on the Newsela corpus (Martinc et al. 2019).

2.4 Fusing Deep Learning and Linguistic Features

Modern deep learning approaches in NLP generally exclude any specific linguistic features like those described in section 2.2.1. Some have tried to incorporate additional semantic information into word embeddings (Trask et al. 2015) but this information is calculated on the data used for pretraining word vectors, not on the texts being classified during the task itself. In general, a "feature-less" approach is sensible given the hypothesis that, with enough data, training, and model complexity, a model should learn any linguistic features that researchers might attempt to precompute. However, precomputed linguistic features may be useful in particular data-poor contexts, especially where acquiring more data is expensive and error-prone. This scenario may be the case for the task of readability (see section 2.1 for details on this issue). For this reason, in this thesis I attempt to incorporate linguistic features with deep learning methods in order to improve readability assessment.

Chapter 3

Methodology

In this chapter I describe the methods used in this thesis for training and testing readability assessment models. Specifically, I discuss corpora composition, model structures, and the trade-offs between different evaluation methods.

3.1 Corpora

Readability corpora are scarce and those that exist are generally small compared to other NLP datasets. This scarcity arises from the aforementioned difficulty in labeling documents for readability. In this section I will detail two of the largest readability corpora, the WeeBit and Newsela corpora, specifically focusing on their composition and validity. I chose to use two corpora because of their substantially different compositions. These differences have led different models to perform best on each corpus (Martinc et al. 2019), further validating the need for using multiple corpora.

3.1.1 WeeBit

Composition

The WeeBit corpus was assembled by Vajjala and Meurers (2012) by combining documents from the WeeklyReader educational magazine and the BBC Bitesize educational website. Data from WeeklyReader were already in use in readability research prior to incorporation in the WeeBit corpus, most notably in early work by Schwarm and Ostendorf (2005). The WeeklyReader data is composed of documents divided into four readability levels roughly corresponding with U.S. grade levels: Level 2, Level 3, Level 4, and Senior intended for readers of ages 7-8, 8-9, 9-10, and 10-12 respectively. The articles are all non-fiction and diverse in topic including subjects ranging “from science to current affairs” (Vajjala and Meurers 2012). The BBC-Bitesize data is also divided into four classes roughly corresponding to the UK education system. In increasing order of difficulty, the classes are KS1, KS2, KS3, and GCSE, corresponding to the age ranges 5-7, 8-11, 11-14, and 14-16 respectively. Vajjala and Meurers (2012) excluded the KS1 level from their analysis as it consisted of mostly non-text content like videos, images, and games.

Using the documents from WeeklyReader and BBC-Bitesize, Vajjala and Meurers (2012) combined the classes selectively to form the WeeBit corpus. They selected classes from each corpus in order to have non-overlapping classes while assembling a broad range of readability levels. They selected the Level 2, Level 3, and Level 4 documents from WeeklyReader along with the KS3 and GCSE documents from BBC-Bitesize. This selection provides an intended reader age ranging from 7 to 16. To avoid classification bias, they undersampled classes in order to equalize the number of documents in each class to 625, an approach mimicked in later literature (Martinc et al. 2019). I will term this downsampled corpus “WeeBit downsampled”. Most experiments in this thesis were performed with the downsampled corpus for the sake of comparison with previous work. However, final model evaluations were performed for both the downsampled and standard WeeBit corpora.

Readability Level	Age Range	Number of Documents
Level 2	7-8	603
Level 3	8-9	779
Level 4	9-10	795
KS3	11-14	633
GCSE	14-16	2521

Table 3.1: WeeBit corpus composition after preprocessing

Following the methodologies of Xia et al. (2016) and Martinc et al. (2019), I applied additional preprocessing to the WeeBit corpus in order to remove extraneous material, e.g. headers, that may artificially boost the performance of assessed models. Documents with fewer than 1 sentence after this preprocessing were removed. This removal caused one class, Level 2, to have fewer than 625 documents in the WeeBit (and downsampled WeeBit) corpus. The number of documents for each class in the preprocessed WeeBit corpus is shown in table 3.1.

Labeling Approaches

Previous usage of readability corpora tended to treat the readability classes as unrelated. These approaches simply used the raw labels as distinct unordered classes. However, readability labels are clearly ordinal, ranging from lower to higher readability. To test the benefit of acknowledging this ordinal quality, I devised three methods for labeling the documents: "classification", "age regression", and "ordered class regression".

The classification approach uses the classes originally given, e.g. "Level 2", "Level 3", "KS3", etc. This approach does not suppose any ordinality of the classes. Avoiding such ordinality may be desirable for the sake of simplicity. Classification is the approach taken by much of the readability literature (Vajjala and Meurers 2012; Martinc et al. 2019).

"Age regression" applies the mean of the age ranges given by the constituent datasets. For instance, in this approach Level 2 documents from Weekly Reader would be given the label of 7.5 as they are intended for readers of ages 7-8. The advantage of this age regression over standard classification is that it provides more precise information about the magnitude of readability difference between different documents and classes. With these age labels, the task becomes one of determining the correct reader age for a new document and is thus framed as a regression rather than classification problem.

Finally, “ordered class regression” assigns the classes equidistant integers ordered by difficulty. The least difficult class would be labeled “0”, the second least difficult class would be labeled “1” and so on. As with age regression, this labeling treats the task of readability assessment as a regression rather than classification problem. This method retains the advantage of age regression in demonstrating the ranking of the class difficulties. However, ordered regression labeling removes information about the relative differences in difficulty between the classes, instead asserting that they are equidistant in difficulty. The motivation behind this loss of information is that such age differences between classes may not directly translate into differences of difficulty. For instance, the readability difference between documents intended for 7 or 8 year-olds may be much greater than between documents intended for 15 or 16 year-olds because reading development is likely accelerated in younger years.

Labeling Validity and Practicality

Given the choices of labeling described, a natural question is how valid and useful such labels would be in education settings. One important choice in selecting labeling methodology is a choice of classification or regression. In many scenarios, the motivation for classification over regression is obvious because the classes do not lie on a spectrum. In determining readability, however, the difficulty measures can be thought of as lying on a continuous spectrum. I suspect that classification is usually chosen because the data are labeled in a discrete manner. That is, there is a set collection of possible labels, e.g. 1st grade, 2nd grade, 3rd grade, etc., and labelers do not assign arbitrarily precise real number labels. This initial labeling is sensible as it is unreasonable to expect humans to have arbitrarily precise judgements of a subjective observation. However, when constructing readability assessment models, it may be desirable to treat the problem as regression as an aid in practical usage, especially considering that the collected labels may be too coarse for optimal reading development. For instance, in classroom settings it may be desirable to have a collection of works that span difficulty ranges within a grade level or age. This aim could also be aided by collecting data with a finer gradation of labels. Barring such new data, another approach involves transforming the discrete labels onto some regression scale, as in the age and ordered class regression described previously. However, evaluating a regression model against a classification labeled dataset leads to misleading results. Any attempt the regression model makes at interpolating finer outputs between class values is penalized in evaluation metrics; therefore, the model's ability to produce those finer outputs is not measured. Thus, given the class labels of the datasets used, regression models will not be evaluated directly in this thesis. Instead, they may be used in intermediate stages of the model as described in section 3.3.5.

Another important choice is the scale of the output; two previously described scales are grade level and age. Both of these scales are desirable for their interpretability. Unlike arbitrarily valued scales like the Flesch score, age and grade levels allow educators and readers to quickly ascertain the appropriate audience for a text. One concern is that age and grade labels ascribe a normative prototype that will not apply to all readers. A solution to this issue is to calibrate the scale to each student: based on a student's progress, they may be suited for books below or above their ascribed grade. Another issue with the measures of age and grade level is their specificity to native language speakers learning to read. Second language learners may have different needs and salient features for determining readability. For these second language learners, different scales may be appropriate.

3.1.2 Newsela

The classes in the WeeBit corpus may be dissimilar in content. For instance, editors are likely to select more complex topics for articles aimed at older readers. This dissimilarity may limit the performance of models trained on WeeBit because the model becomes overly attentive to vocabulary and subject matter. For instance, suppose a child wishes to read about a topic they find challenging, often only studied by older students. A model trained on a corpus like WeeBit may judge all or most material on such a topic as too challenging for the reader, regardless of authors' attempts to make their works approachable for younger audiences.

This limitation encourages the use of corpora in which topics are shared across readability levels. One such corpus is the Newsela corpus assembled by Xu et al. (2015). It consists of 1,911 news articles each re-written up to 4 times in simplified manners for readers at different reading levels. This simplification process means that, for any given topic, there exist examples of material on that topic suited for multiple reading levels. This overlap in topic should also make the corpus more challenging to label than the WeeBit corpus. In a similar manner to the WeeBit corpus, the Newsela corpus is labeled with grade levels ranging from grade 2 to grade 12. As with WeeBit, these labels can either be treated as classes or transformed into numeric labels for regression.

Given that each article is repeated multiple times within the Newsela corpus, one concern is that there could be overlap between the training and test sets. This could lead to overfitting this particular corpus diminishing future generalizability and inflating results. To address this concern, one approach is to not separate document sets, that is, documents originating from the same article. With this approach, if one document from an article is assigned to the test set, then all the differing difficulty documents originating from that article must also be assigned to the test set. This thesis contains experiments both maintaining and splitting document sets to test the significance of such concerns.

3.2 Linguistic Features

Motivated by the success in using linguistic features for modeling readability, I considered a large range of textual analyses relevant to readability. In addition to utilizing features posed in the existing readability research, I investigated formulating new features with a focus on syntactic ambiguity. This challenging aspect of language appeared to be underutilized in existing readability literature.

3.2.1 Existing Features

To capture a broad array of features related to complexity, I utilized existing linguistic feature computation software[1] developed by Vajjala Balakrishna (2015), Vajjala and Meurers (2016), and Vajjala and Meurers (2014) based on 88 feature descriptions in existing readability literature. Given the large number of features, in this section I will focus on the categories of features and their psycholinguistic motivations (where available) and properties.

The most basic features involve what Vajjala and Meurers (2012) refer to as "traditional features" for their use in long-standing readability formulae. They include the characters per word, syllables per word, and formulas based on such features like the Flesch-Kincaid formula (Kincaid et al. 1975).

Another set of feature types consists of counts and ratios of part-of-speech tags, extracted using the Stanford parser (Klein and Manning 2003). In addition to basic parts of speech like nouns and adjectives, some features include phrase level constituent counts like noun phrases and verb phrases. All of these counts are normalized by either the number of word tokens or number of sentences to make them comparable across documents of differing lengths. These counts are not provided with any psycholinguistic motivation for their use; however, it is not an unreasonable hypothesis that the relative usage of these constituents varies across reading levels. Empirically, these features were shown to have some predictive power for readability. In addition to parts of speech counts, I also utilized word type counts as a simple baseline feature, that is, counting the number of instances of each possible word in the vocabulary. These counts are also divided by document length to generate proportions.

Becoming more abstract than parts of speech, some features count even more complex

[1]This code can be found at `https://bitbucket.org/nishkalavallabhi/complexity-features/src/master/`.

syntactic constituents like clauses and SBARs[2]. Specifically, Lu (2010) found a variety of ratios involving sentences, clauses, and t-units[3] correlated with second language learners' abilities to read a document. For many of the multi-word syntactic constituents previously described such as noun phrases and clauses, features were also constructed of their mean lengths. Finally, properties of the syntactic trees themselves were analyzed such as their mean heights.

Moving beyond basic features from syntactic parses, Vajjala Balakrishna (2015) also incorporated "word characteristic" features from linguistic databases. A significant source was the Celex Lexical Database (Baayen et al. 1995) which "consists of information on the orthography, phonology, morphology, syntax and frequency for more than 50,000 English lemmas". The database appears to have a focus on morphological data such as whether a word may be considered a loan word and whether it contains affixes. It also contains syntactic properties that may not be apparent from a syntactic parse, e.g. whether a noun is countable. The MRC Psycholinguistic Database (Wilson 1988) was also used with a focus on its age of acquisition ratings for words, a clear indicator of the appropriateness of a document's vocabulary.

3.2.2 Novel Syntactic Features

Motivated by the syntactic feature work of Vajjala Balakrishna (2015), I investigated creating additional syntactic features that may be relevant for readability but whose qualities were not targeted by existing features. Specifically, I focused on constructing features that indicate the syntactic regularity and ambiguity of a document. I measured these features using analyses of syntactic parses and POS counts.

For generating syntactic parses, I used the PCFG (probabilistic context-free grammar) parser (Klein and Manning 2003) from the Stanford Parser package. I chose this parser as

[2]As defined by Bies et al. (1995), an SBAR is a "[c]lause introduced by a possibly empty subordinating conjunction".

[3]Defined by Vajjala and Meurers (2012) to be "one main clause plus any subordinate clause or non-clausal structure that is attached to or embedded in it".

it can output multiple possible parses for a given sentence, labeled with their likelihood of correctness. These scored parses are critical for the parser uncertainty measures discussed in this section. This parser uses the tags of the Penn Treebank project, the description of which can be found in Bies et al. (1995).

Parser uncertainty

Any given sentence often has multiple possible syntactic parses that adhere to the grammar of the language. However, all of these parses are not equally likely to be the one intended by the author. In light of these unequal probabilities, syntactic parsers often make judgments of parse likelihood. Furthermore, some can output a list of possible parses, each annotated with a probability or derivation of probability (in the case of the PCFG parser, a log probability). Using these probabilities, syntactic ambiguity measures can be constructed.

Initially, it may seem sensible to use the number of parses generated as a measure of ambiguity. However, this measure is extremely sensitive to sentence length as longer sentences tend to have more possible syntactic parses. Instead, if this list of probabilities is viewed as a distribution, the standard deviation of this distribution is likely to have some correlation with perceptions of syntactic ambiguity while being less sensitive to sentence length.

Definition 3.2.1. PD_x

The parse deviation, $PD_x(s)$, of sentence s is the standard deviation of the distribution of the x most probable parse log probabilities for s. If s has less than x valid parses, the distribution is taken from all the valid parses.

For a highly unambiguous sentence, I hypothesized that one parse is likely to have a much higher probability than other possible parses, leading to a small $PD_x(s)$. By contrast, I predicted that the parser is likely to have more trouble with highly ambiguous sentences, leading to similar probabilities among multiple parses and thus a higher $PD_x(s)$.

For large values of x, $PD_x(s)$ can be substantially sensitive to sentence length: longer sentences are likely to have more valid syntactic parses and thus create low probability tails that increase standard deviation. To reduce this sensitivity, some possibilities include decreasing x, dividing $PD_x(s)$ by the sentence length, or measuring the difference between the largest and mean parse probability.

Definition 3.2.2. PDM_x

$PDM_x(s)$ is the difference between the largest parse log probability and the mean of the log probabilities of the x most probable parses for a sentence s. If s has less than x valid parses, the mean is taken over all the valid parses.

Normalizing by the sentence length or using PDM_x (for large x), while somewhat reducing the effect of sentence length, still incorporates information about syntactic parses that many readers describe as nonsensical, uninformative, and beyond consideration. As an example of this poor parsing, the 96th best (96th highest log probability) PCFG parse for the sentence "I like trains" is shown in figure 3.1. The parser interprets "like" as a preposition, an interpretation that is unlikely to occur for a human.

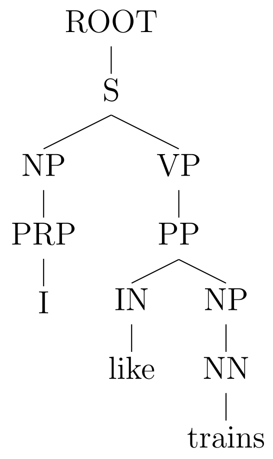

Figure 3.1: The 96th most probable syntactic tree produced by the PCFG parser for the sentence "I like trains"

These poor parses can have the effect of adding noise and obfuscating measurement of ambiguity. For instance, the sentences in example 1 have similar length and content and are distinguished in readability primarily by syntactic ambiguity. Yet, even when length is equivalent, the PD_{100} of each of the sentences in example 1 is nearly identical, with the more ambiguous sentence 1a[4] having a slightly lower PD ($PD(s_{1a}) \approx 2.013$) than the less ambiguous 1b ($PD(s_{1b}) \approx 2.105$). Clearly, PD_x for large values of x fails to ascertain ambiguity, likely due to the many nonsensical noise-inducing parses generated like the one shown in figure 3.1. Limiting x to smaller values tends to yield better measures of syntactic ambiguity, as shown in figure 3.2. In particular, I hypothesized that most sentences that readers deem ambiguous sentences would have only two possible interpretations. This hypothesis would support using PD_2 or PDM_2 as a measure of syntactic ambiguity.

(1) a. The professor said on Monday he would give an exam.
    b. The professor said he would give a Monday exam.

(2) a. I saw the pirate with the telescope.
    b. Using the telescope, I saw the pirate.

(3) a. Visiting relatives can be exhausting.
    b. Relatives who are visiting can be exhausting.

(4) a. Let us stop controlling people.
    b. Let us stop controlling other people.

(5) a. I fed her cat food.
    b. I fed cat food to her.

(6) a. The person sat at the piano with a broken leg.
    b. The person with a broken leg sat at the piano.

Indeed, PD_2 often seems to correlate with syntactic ambiguity. The PD_2 value of example 1a is 0.058 while the PD_2 value for example 1b is 0.910, supporting the hypothesis that a more ambiguous sentence will result in lower PD_2 values. Figure 3.2 shows differences in PD_x values between the ambiguous a and unambiguous b sentences of example 1 through

[4]In example 1a, the ambiguity concerns which verb the prepositional phrase "on Monday" applies to: either the exam will take place on Monday, or the professor made the statement on Monday.

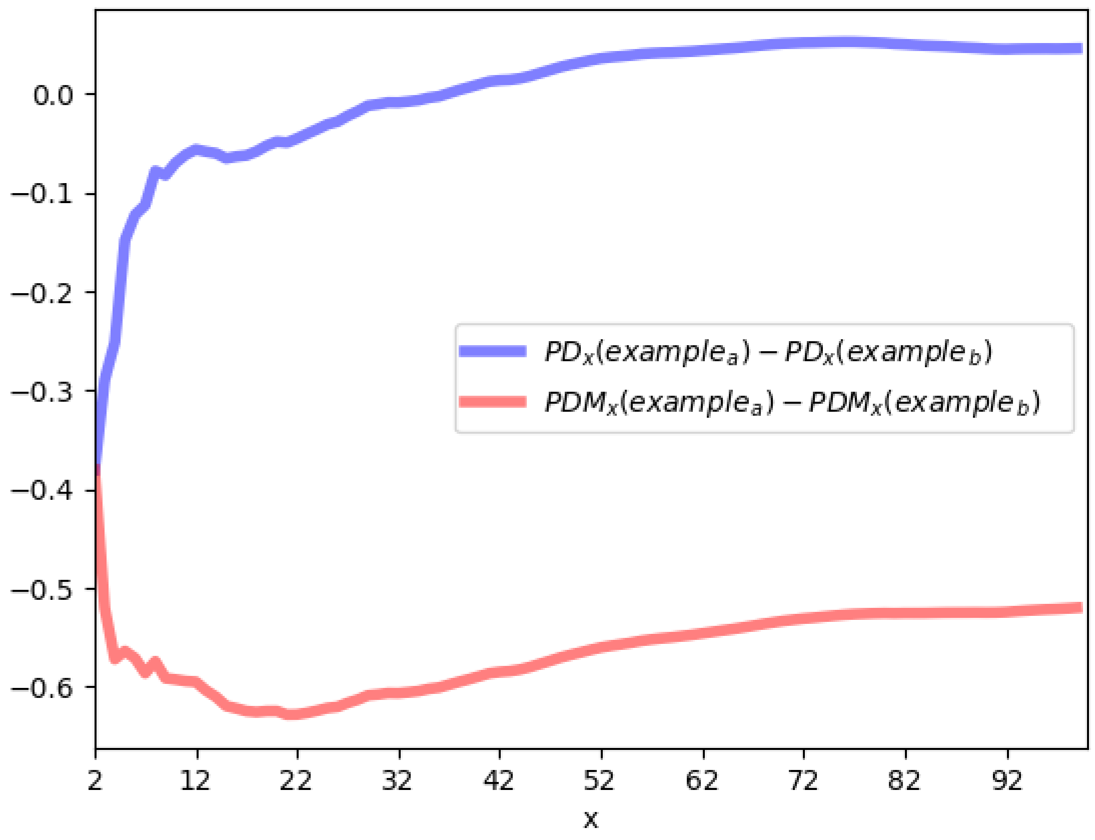

Figure 3.2: Differences in PD_x and PDM_x between ambiguous and unambiguous averaged over the sentences in example 1 through example 6.

example 6.[5] The initially falling slope of the PDM_x plot demonstrates that $x = 2$ is not always the most optimal choice for detecting syntactic ambiguity. This result may be biased by the choice of samples; therefore, further investigation is needed to determine optimal values of x. Additionally, the PDM_x plot demonstrates that PDM_x is less sensitive than PD_x to overly large values of x. As a compromise between parse investigation and the noise of implausible parses, I selected PDM_{10}, PD_{10}, and PD_2 as features to use in the models of this thesis.

[5]Because the a and b example sentences greatly differ in syntactic ambiguity, one would expect the measures of their ambiguity to diverge substantially. Thus, the y-axis of the plot is the difference of their ambiguity measures and a larger absolute value on the y-axis indicates better measurement of syntactic ambiguity.

Corpus	Metric	Pearson correlation	Pearson p-value	Spearman correlation	Spearman p-value
WeeBit	PD_2	0.2375	<0.001	0.3292	<0.001
WeeBit	PD_{10}	0.3157	<0.001	0.4090	<0.001
WeeBit	PDM_{10}	0.3360	<0.001	0.4127	<0.001
Newsela	PD_2	-0.7072	<0.001	-0.7699	<0.001
Newsela	PD_{10}	-0.8044	<0.0001	-0.8633	<0.001
Newsela	PDM_{10}	-0.7896	<0.0001	-0.8518	<0.001

Table 3.2: Parse Standard Deviation Correlations

Thus, PD_x or PDM_x both provide some measure of syntactic ambiguity, with PDM_X being preferred due to its more stable properties over values of x. Correlation of these metrics with the corpora (ordered class) labels, shown in table 3.2, muddies this hypothesis. On one hand, the correlations for the WeeBit corpus are significant and positive, supporting the hypothesis that a greater standard deviation in the syntactic parse likelihoods indicates a greater difficulty. However, the correlations for the Newsela corpus are also significant and of opposite direction indicating that this correlation is spurious.

Part-of-Speech Divergence

To capture the grammatical makeup of a sentence or document, we can count the usage of each part of speech ("POS"), phrase, or clause. These counts can be used as features themselves, providing information about the usage of a given grammatical feature. For instance, Schwarm and Ostendorf (2005) counted the number of noun phrases and verb phrases per sentence. Extending this idea, the counts can be collected into a distribution. Then, the standard deviation of this distribution, $POSD_{dev}$, may measure some notion of a sentence's grammatical heterogeneity; this measure is motivated by my hypothesis that more difficult texts will contain more grammatical heterogeneity.

Definition 3.2.3. $POSD_{dev}$

$POSD_{dev}(d)$ is the standard deviation of the distribution of POS counts for document d.

Similarly, we may want to measure how the grammatical makeup of each sentence differs

Corpus	Metric	Pearson correlation	Pearson p-value	Spearman correlation	Spearman p-value
WeeBit	POS_{div}	0.2962	<0.0001	0.3695	<0.0001
WeeBit	$POSD_{dev}$	-0.0989	<0.0001	-0.1231	<0.0001
Newsela	POS_{div}	-0.7495	<0.0001	-0.8011	<0.0001
Newsela	$POSD_{dev}$	0.6394	<0.0001	0.7017	<0.0001

Table 3.3: Part-of-speech divergence and standard deviation correlations

from the composition of the document as a whole, a concept that might be termed syntactic uniqueness. To capture this concept, I measure the Kullback-Leibler divergence (Kullback and Leibler 1951) between the sentence POS count distribution and the document POS count distribution.

Definition 3.2.4. POS_{div}

Let $P(s)$ be the distribution of POS counts for sentence s in document d. Let Q be the distribution of POS counts for document d. Let $|d|$ be the number of sentences in d.

$$POS_{div}(d) = \sum_{s \in d} \frac{D_{KL}(P(s) \mid\mid Q)}{|d|}$$

Correlations for these metrics and the labels (ordered class) in the WeeBit corpus are shown in table 3.3. All metrics demonstrated a significant correlation. However, the correlation patterns are not consistent across corpora with the direction of correlation disagreeing in addition to substantial differences in magnitude. This inconsistency indicates that perhaps these POS metrics are not independently informative in determining a document's readability.

3.3 Models

Given these novel and existing features along with word vectors (described in section 2.2.2), I utilized a variety of machine learning models to infer document readability. Some of these models are prominently used in the readability literature. All have been widely used in a variety of machine learning tasks and achieved substantial performance in some subset of tasks. A large range of model complexities were evaluated in order to ascertain the performance improvements, or lack thereof, of additional model complexity. In this section I will describe the specific construction and usage of these models for the experiments conducted in this thesis, ordered roughly by model complexity.

3.3.1 SVMs, Linear Models, and Logistic Regression

I used the Scikit-Learn Python library (Pedregosa et al. 2011) for constructing SVM models. Hyper-parameter optimization was performed each time an SVM was trained using the guidelines suggested by Hsu et al. (2003) which were utilized by Schwarm and Ostendorf (2005) in constructing models for readability. These guidelines prescribe a grid search over exponentially growing hyperparameters with a preference for utilizing linear and RBF (Radial Basis Function) kernels. Accordingly, I perform an exhaustive grid search over the parameters shown in table 3.4. The grid search evaluates the model with five-fold cross-validation (using only the training data) for each possible combination of hyperparameters and returns the model that achieves the highest weighted F1 score (defined in section 3.4.2). The features are also linearly scaled to the range [0, 1] which has been shown to improve performance (Hsu et al. 2003) likely because it removes any a priori differences in weighting between features.

From the Scikit-Learn library, I also utilized the linear support vector classifier (essentially an SVM with a fixed linear kernel) and logistic regression classifier. These models were selected as simpler analogues of the SVM to observe if the additional complexity of the SVM (namely its possible RBF kernel) elicited any improvement in performance. As simplicity was the aim for these evaluations, no hyperparameter optimization was performed. The logistic regression classifier was trained using the stochastic average gradient descent ("sag") optimizer.

Kernel	Hyperparameter	Values
RBF	Gamma	$10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}$
RBF	C	$1, 10^{1}, 10^{2}, 10^{3}, 10^{4}$
Linear	C	$1, 10^{1}, 10^{2}, 10^{3}$

Table 3.4: SVM hyperparameters used for grid search

3.3.2 CNN

Convolutional neural networks were selected for their demonstrated performance on sentence classification (Kim 2014). As neural-network-based models, CNNs are more complex than the models described and used previously like SVMs and logistic regression. Subjectively, they are more complex in their specification, i.e. it is more difficult to describe their construction. Objectively, they are more complex in that they tend to have many more parameters. While generally more complex than the non-neural-network-based models, CNNs tend to be less complex than the sequence-to-sequence and attention-based models described in the following sections. The CNN model used in this thesis is based on the one described by Kim (2014) and implemented using the Keras (Chollet et al. 2015), Tensorflow (Abadi et al. 2015), and Magpie libraries (*Magpie* 2020).

Prior to feeding data into the CNN, the input words are transformed into word vectors via a pretrained collection of word vectors trained on the Google News Dataset (around one billion words) (Mikolov et al. 2013). The model itself is composed first of a convolution layer composed of multiple filters with varying window lengths. The window length is the width, in number of words, of the filter. For the models in this thesis, window lengths ranging from 1 to 5 words, inclusive, are used. Each of these convolutions uses the hyperbolic tangent activation function. After this convolution layer, a max pooling layer is applied that selects the maximum value from the result of each convolution. The penultimate layer takes this max-pooling result and applies a form of regularization known as dropout. Dropout involves randomly zeroing some proportion of hidden units during the training step thereby encouraging generalization and diversity among the weights as a whole. The dropout rate is set to 0.5 for the CNN models meaning half of the dropout layer inputs are zeroed during training. The final layer is a simple fully connected linear layer that produces either a numeric label in the case of regression or a list of probabilities over the classes in the case of classification. For classification the final activation function is the sigmoid function and for regression the final activation function is linear.

3.3.3 Transformer

The transformer (Vaswani et al. 2017) is a neural-network-based model that has achieved state-of-the-art results on a wide array of natural language tasks including readability assessment (Martinc et al. 2019). Transformers are formulated as sequence-to-sequence models often translating between languages or generating text based on a prompt. They utilize the mechanism of attention which allows the model to attend to specific parts of the input when constructing the output. Although they are formulated as sequence-to-sequence models, they can be modified to complete a variety of NLP tasks by placing an additional linear layer at the end of the network and training that layer to produce the desired output. This approach often achieves state-of-the-art results when combined with pretraining. In the pretraining stage, the transformer is trained on a large generic corpus of text to simply act as a language model, that is predict some text given some initial portion of the text. Then, the transformer's weights are frozen, the final linear layer is added, and the weights of that final layer are trained ("fine-tuned") on a smaller corpus specific to the desired task. In this way, the model is meant to gain a general understanding of language in pretraining and use that knowledge to achieve better results in a specific data-poor task.

Specifically in this thesis, I use the BERT (Devlin et al. 2019) transformer-based model that is pretrained on the BooksCorpus (800M words) (Zhu et al. 2015) and English Wikipedia. The model is then fine-tuned on a specific readability corpus such as WeeBit. The pretrained BERT model is sourced from the Huggingface transformers library (Wolf et al. 2019) and is composed of 12 hidden layers each of size 768 and 12 self-attention heads. The fine-tuning step utilizes an implementation by Martinc et al. (2019). The transformers are constrained to only accept inputs of fixed size. Among the pretrained transformers in the Huggingface library, there are transformers that can accept sequences of size 128, 256, and 512. The 128 sized model was chosen based on the finding by Martinc et al. (2019) that it achieved the highest performance on the WeeBit and Newsela corpora. Documents that exceeded the input sequence size were truncated.

3.3.4 HAN

The Hierarchical attention network involves feeding the input through two bidirectional RNNs (in this implementation GRUs) each accompanied by a separate attention mechanism. One attention mechanism attends to the different words within each sentence while the second mechanism attends to the sentences within the document. These hierarchical attention mechanisms are thought to better mimic the structure of the document and consequently produce superior classification results. The implementation of the model used in this thesis is identical to the original architecture described by Yang et al. (2016) and was provided by the authors of Martinc et al. (2019) based on code by Nguyen (2020).

3.3.5 Incorporating Linguistic Features with Neural Models

The neural network models thus far described take either the raw text or word vector embeddings of the text as input. Natively, they make no use of linguistic features such as those described in section 3.2. I hypothesized that combining these linguistic features with the deep neural models may improve their performance on readability assessment. Although these models theoretically represent similar features to those prescribed by the linguistic features, I hypothesized that the amount of data and model complexity may be insufficient to capture them. This can be evidenced in certain models failing to generalize across readability corpora. Martinc et al. (2019) found that the BERT model performed well on the WeeBit corpus, achieving a weighted F1 score of 0.8401, but performed poorly on the Newsela corpus only achieving an F1 score of 0.5759. They posit that this disparity occurred "because

BERT is pretrained as a language model, [therefore] it tends to rely more on semantic than structural differences during the classification phase and therefore performs better on problems with distinct semantic differences between readability classes". Similarly, a HAN was able to achieve better performance than BERT on the Newsela but performed substantially worse on the WeeBit corpus. Thus, under some evaluations the models have deficiencies and fail to generalize. Given these deficiencies, I hypothesized that the inductive bias provided by linguistic features may improve generalizability and overall model performance.

In order to weave together the linguistic features and neural models, I take the simple approach of using the output of a neural model as a feature itself, joined with linguistic features, and then fed into one of the simpler non-neural models such as SVMs. The output of the neural model could be any of the label approaches such as grade classes or age regressions described in section 3.1.

3.4 Evaluation

Assessing the performance of machine learning models is not entirely straightforward with many competing metrics and methodologies. Considerations must be made as to what properties and attributes are desired when used in real scenarios. Additionally, metrics most consider the makeup of the dataset used for evaluation. Skewed or unrepresentative datasets may fail to model future data in a way that diminishes the informativeness of evaluation metrics. In this section, I will explain and motivate the evaluation metrics and methodologies used in this thesis.

3.4.1 Evaluation Metrics

One of the simplest metrics is absolute error: the absolute value of the difference between the true and predicted label of a sample. This metric has the advantage of interpretability and simplicity. Its units are identical to those of the data and provides an intuitive sense of the model's ability to produce correct labels. Absolute error has the disadvantage of treating all errors equally; put another way, it has no specific mechanism for penalizing outliers. The desire for this property can be shown via a simple example. Imagine a regression dataset with two samples both with a true label of 1. One predictor, h_1 assigns predicted labels of 2 to both samples. A second predictor, h_2, assigns a predicted label of 3 to the first sample and a label of 1 to the second example. While both predictors have an equivalent average absolute error of 1, in some scenarios h_1 may be preferable because the harm caused by outliers is more substantial than the advantage gained by greater accuracy close to the true labels. The metric of mean squared error (MSE) achieves this aim by defining its error as the square of the difference between true and predicted labels thereby imposing greater penalties on inferences more distant from true labels. Root mean squared error (RMSE) simply takes the square root of MSE so that its units are equivalent to those of the dataset labels. In the results provided in this thesis, the units of RMSE are the number of classes as in the ordered class regression paradigm.

Absolute error, MSE, and RMSE are metrics framed in the paradigm of regression, a regime that may be appropriate for readability as described in section 3.1. If the task of readability assessment is to be framed as a classification problem, as it has often been in the literature, a different set of metrics is needed. Two basic classification metrics include precision and recall. Recall refers to the number of positive examples labeled correctly (true positives) divided by the total number of positive examples (true positives + false negatives). Intuitively, given an example with a ground-truth positive label, recall measures the ability of the model to correctly identify it as positive. By contrast, precision refers to the number of positive examples labeled correctly (true positives) divided by all examples that the model labeled as positive (true positive + false positives). Intuitively, given an example that the model labeled as positive, precision measures how confident one can be in that predicted label. Neither of these metrics alone do a good job capturing the performance of a model. For instance, recall can be inflated by labeling all samples as positive and similarly precision can be inflated by biasing toward negative results. These weaknesses indicate that inflating one metric will tend to decrease the other. For this reason, they are often combined into a composite metric such as the popular F1 score first proposed by Chinchor (1992). The F1 score, defined in equation 3.1, is the harmonic mean of precision and recall, the harmonic mean being used instead of the arithmetic mean in order to penalize extreme precision or recall values (Koehrsen 2019).

$$\mathrm{F1} = 2 \cdot \frac{precision \cdot recall}{precision + recall} \tag{3.1}$$

Given that the F1 score is defined for binary classification, it must be extended to multiclass classification for use in readability assessment. One simple approach is to treat the multiclass classification problem as multiple binary classification problems. For each class, the number of correctly identified samples are deemed true positives and the number of non-class examples identified as class members are the false positives. Similarly, class members negatively identified are false negatives and non-class examples identified as negative are true negatives. Pooling these true and false positives and negatives across all classes to calculate the F1 score is known as the micro F1 score. Another method is to calculate the F1 score for each class independently and then average the F1 scores across the classes; this method is known as the macro F1 score. This macro method has the advantage of assigning equal weight to each class thereby not favoring classes with more samples. This equality across classes could also be a disadvantage by allowing the model to seemingly perform well by only correctly classifying a small number of samples from smaller classes. A final method is the weighted F1 score in which the F1 score is calculated for each class and then combined in a weighted average in which the weights are proportional to the number of samples in the class. Weighted F1 and micro F1 achieve similar aims in placing greater weight on larger classes; in this thesis I utilize weighted F1 for comparison to previous literature.

3.4.2 Cross Validation

In machine learning often a large segment of the data is taken as training data and the remaining data is used solely for evaluation. This splitting prevents the model from being highly tuned ("overfitted") to the data it is trained on. However, the choice of split may affect the model performance. For this reason, the technique of cross validation is used. In cross validation, a certain k number of splits are selected (often 5). For each split, $\frac{1}{k}$ samples of the data are selected as a test set and the remainder as the training set. Each of the test sets are disjoint thereby testing the robustness of the model to different evaluation data. The results reported are then the averages across all of the splits as is done in this thesis.

Chapter 4

Results

In this chapter I report the experimental results of incorporating linguistic features into readability assessment models. The two corpora, WeeBit and Newsela, are analyzed individually and then compared. Additional analysis is provided of the effects of linguistics features at different dataset sizes and the significance of some individual features. My results demonstrate that, given sufficient data, linguistic features provide little to no benefit compared to independent deep learning models. While the corpus experiment results demonstrate a portion of the approaches tested, the full results are available in appendix A.

4.1 Newsela Experiments

The Newsela corpus was evaluated under two paradigms: standard evaluation and document set evaluation. The standard evaluation used the corpus ordinarily as it has been used in the previous literature. The document set evaluation involved ensuring that documents originating from a common article were not separated between the training and test sets (see section 3.1.2 for further discussion of this issue). In this section, I describe and compare model performance across the two paradigms.

Features	Weighted F1	Macro F1	RMSE	SD Weighted F1	SD macro F1	SD RSME
SVM with transformer and Flesch features	0.7186	0.6305	1.127	0.0074	0.0282	0.0290
SVM with CNN ordered classes regression and linguistic features	0.7231	0.6053	1.204	0.0062	0.0331	0.0280
SVM with CNN age regression and linguistic features	0.7281	0.6104	1.186	0.0057	0.0337	0.0347
SVM with Linguistic features	0.7582	0.6432	1.070	0.0089	0.0379	0.0369
SVM with transformer, Flesch features, and linguistic features	0.7627	0.6263	1.062	0.0075	0.0301	0.0352
SVM with transformer and linguistic features	0.7678	0.6656	1.023	0.0230	0.0385	0.1157
SVM with linguistic features and Flesch Features	0.7694	0.6446	1.049	0.0060	0.0406	0.0306
SVM with HAN	0.7931	0.6724	0.5562	0.0448	0.0449	0.0868
SVM with HAN and linguistic features	0.8014	0.6751	0.6426	0.0263	0.0379	0.2346
HAN	0.8024	0.6775	0.4897	0.1116	0.1825	0.2756

Table 4.1: Top 10 performing model results on the Newsela corpus

4.1.1 Standard Experiments

For the Newsela corpus, while linguistic features were able to improve the performance of some models, the top performers did not utilize linguistic features. The results from the top performing models are presented in table 4.1.

While the HAN performance was not surpassed by models with linguistic features, the transformer models were. This improvement indicates that linguistic features likely capture readability information that transformers cannot capture alone or have insufficient training data to learn. The outsize effect of adding the linguistic features to the transformer models, resulting in a weighted F1 score improvement of 0.22, may reveal what types of information they are suited to addressing. Martinc et al. (2019) hypothesize that a pretrained language model "tends to rely more on semantic than structural differences" indicating that these features are especially suited to providing non-semantic information such as syntactic qualities. The linguistic capacities of these models may be analyzed more specifically via probing techniques (Conneau et al. 2018).

4.1.2 Document Set Experiments

The experiments on the Newsela corpus were also performed without separating document sets, as described in section 3.1.2, the results of which are shown in table 4.2. This modification had little effect on the performance of the models, with many models observing less than 0.01 change in weighted F1 score. Many of the changes in performance were small enough to be insubstantial and are likely to be attributed to the stochastic nature of the models rather than true differences in task difficulty. While the ranking of models did change, this mostly occurred where the difference in model performance was already very similar; thus, small modifications in results were likely to change rankings. The largest change occurred for the HAN models which decreased in weighted F1 performance by about 0.05. Perhaps this decrease in performance occurred because in the non-document set experiments the HAN models were able to gain information from the document sets that spanned both the training and test sets. However, this explanation seems dubious considering such an effect was not observed, or smaller, for other model types; in fact, some other model types like logistic regression with word types showed increases in performance when moving to the document set experiments. Instead, the HAN models' decrease in performance may be due to sensitivity to different random splits of the training and test data rather than the methodology of splitting.

This result indicates that the overlap between training and test sets in the non-document set experiments has little effect on model performance. This likely occurs because each document set forms a small component of the overall dataset: while each document set is comprised of at most 11 documents, the dataset as a whole contains 9,565 documents. Thus, the inductive bias provided by the training documents in each document set can only be utilized for a very small portion of the test set, so few that this information is not regularly incorporated into the model. Given document sets that formed a larger portion of the overall dataset, perhaps the effect of maintaining document sets would be larger. Additionally, this lack of effect may occur due to the dissimilarity between documents in a document set.

Features	Weighted F1	Macro F1	RMSE	SD Weighted F1	SD macro F1	SD RSME
SVM with CNN classifier, and linguistic features	0.7062	0.5571	1.1164	0.0157	0.0291	0.0642
SVM with transformer and Flesch features	0.7074	0.6063	1.1093	0.0141	0.0370	0.0188
SVM with Flesch features	0.7184	0.6133	1.2024	0.0063	0.0266	0.0265
HAN	0.7487	0.6203	0.6193	0.0149	0.0138	0.0389
SVM with transformer, Flesch features, and linguistic features	0.7556	0.6319	1.0547	0.0103	0.0305	0.0310
SVM with transformer and linguistic features	0.7569	0.6441	1.0645	0.0087	0.0328	0.0345
SVM with HAN and linguistic features	0.7611	0.6331	1.0610	0.0126	0.0328	0.0426
SVM with Linguistic features	0.7639	0.6252	1.0494	0.0102	0.0305	0.0391
SVM with linguistic features and Flesch Features	0.7673	0.6277	1.0420	0.0095	0.0315	0.0334
SVM with HAN	0.7760	0.6520	0.5888	0.0769	0.0553	0.1248

Table 4.2: Top 10 performing model results on the document set Newsela corpus

Although they originate from the same article and thus convey similar content, their altered difficulty levels may distinguish them enough that one article does not substantially aid in the readability assessment of another in the same document set.

4.2 WeeBit Experiments

The WeeBit corpus was also analyzed in two perspectives: the downsampled dataset and the full dataset. Raw results and model rankings were largely comparable between the two dataset sizes.

4.2.1 Downsampled WeeBit Experiments

As with the Newsela corpus, the downsampled WeeBit corpus demonstrates no gains from being analyzed with linguistic features. The best performing model, a transformer, did not utilize linguistic features. The results for some of the best performing models are shown in table 4.3.

Features	Weighted F1	Macro F1	RMSE	SD Weighted F1	SD macro F1	SD RSME
Logistic regression classifier with word types	0.7894	0.7887	0.5576	0.0151	0.0202	0.0234
Logistic regression classifier with word types and word count	0.7908	0.7899	0.5529	0.0130	0.0182	0.0198
SVM with CNN classifier and linguistic features	0.7923	0.7919	0.5894	0.0210	0.0193	0.0612
Logistic regression classifier with word types, word count, and Flesch features	0.7934	0.7926	0.5487	0.0135	0.0187	0.0166
Logistic regression classifier with word types, Flesch features, and linguistic features	0.8135	0.8130	0.5127	0.0131	0.0169	0.0130
SVM with transformer	0.8343	0.8340	0.4617	0.0131	0.0135	0.0320
SVM with transformer and linguistic features	0.8344	0.8347	0.4559	0.0106	0.0091	0.0134
SVM with transformer and Flesch features	0.8359	0.8358	0.4644	0.0151	0.0154	0.0216
SVM with transformer, Flesch features, and linguistic features	0.8381	0.8377	0.4567	0.0128	0.0118	0.0230
Transformer	0.8387	0.8388	0.4537	0.0097	0.0073	0.0129

Table 4.3: Top 10 performing model results on the downsampled WeeBit corpus

Differing with the Newsela corpus, the word type models performed near the top results on the WeeBit corpus comparably to the transformer models. Word type models have no access to word order, thus semantic and topic analysis form their core analysis. Therefore, this result supports the hypothesis of Martinc et al. (2019) that the pretrained transformer is especially attentive to semantic content. This result also indicates that the word type features can provide a substantial portion of the information needed for successful readability assessment. The fact that the CNN models performed worse than the word type models indicates that pretrained word vectors are not especially useful for readability assessment. Rather, letting models learn specific representations for the task, as they essentially do in the word type models, leads to greater performance.

The differing best performing model types between the two corpora are likely due to differing compositions. Unlike the Newsela corpus, the WeeBit corpus shows strong correlation between topic and difficulty. Extracting this topic and semantic content is thought to be a particular strength of the transformer (Martinc et al. 2019) leading to its improved results on this corpus.

4.2.2 Full WeeBit Experiments

All of the models were also tested on the full imbalanced WeeBit corpus, the top performing results of which are shown in table 4.4. As expected, given the additional data, most performance figures increased modestly. However, given the concerns about classification bias, it is difficult to tell if these gains would be seen in real-world usage where the set of evaluated documents may not match the distribution of this dataset. Additionally, the ranking of models between the downsampled and standard WeeBit corpora showed little change. This lack of change indicates that the results comparing model performance are robust to dataset size and imbalanced datasets. Although the SVM with transformer and linguistic features performed better than the transformer alone, this difference is extremely small (< 0.005) and thus not likely to be statistically significant. This lack of improvement further demonstrates that linguistic features cannot improve upon the best performing deep learning models.

4.3 Effects of Linguistic Features

Overall, the failure of linguistic features to improve the best performing models indicates that, given the available corpora, model complexity, and model structures, they do not aid readability assessment. Perhaps given more diverse and more accurately and consistently labeled corpora, the linguistic features could prove useful. It may be the case that the best performing models already achieve near the maximal possible performance on these corpora. The reason the maximal performance may be below a perfect score (an F1 score of 1) is disagreement and inconsistency in dataset labeling. Presumably the datasets were assessed by multiple labelers or authors who may not have always agreed with one another or even

Features	Weighted F1	Macro F1	RMSE	SD Weighted F1	SD macro F1	SD RSME
CNN	0.8282	0.7748	0.5609	0.0211	0.0183	0.0405
SVM with CNN classifier and linguistic features	0.8286	0.7753	0.5546	0.0222	0.0209	0.0470
Logistic regression classification with word types, Flesch features, and ling features	0.8293	0.7760	0.5192	0.0152	0.0172	0.0323
SVM with CNN classifier	0.8296	0.7754	0.5500	0.0163	0.0136	0.0430
SVM with HAN and linguistic features	0.8441	0.7970	0.6302	0.0643	0.0827	0.2253
SVM with transformer, Flesch features, and linguistic features	0.8721	0.8273	0.4133	0.0095	0.0121	0.0130
Transformer	0.8721	0.8272	0.4013	0.0071	0.0102	0.0184
SVM with transformer	0.8729	0.8288	0.4208	0.0064	0.0090	0.0225
SVM with transformer and Flesch features	0.8746	0.8305	0.4166	0.0054	0.0107	0.0260
SVM with transformer and linguistic features	0.8769	0.8343	0.4041	0.0077	0.0129	0.0199

Table 4.4: Top 10 performing model results on the WeeBit corpus

with themselves. Thus, if either a new set of human labelers or the original labelers are tasked with labeling readability in these corpora, they may only achieve performance similar to the best performance seen in these experiments. Performing this human experiment would be a useful analysis of corpora validity and consistency. Similarly, more diverse corpora (differing in length, topic, writing style, etc.) may prove more difficult for the models to label alone without additional training data; in this case, the linguistic features may prove more helpful in providing inductive bias.

4.3.1 Subsample Experiments

One hypothesis explaining the lack of effect of linguistic features is that models learn to extract those features given enough data. Thus, perhaps in more data-poor environments the linguistic features would prove more useful, even when the data is semantically distinct as it is in the WeeBit corpus. To test this hypothesis, I evaluated two CNN based models, one with linguistic features and one without, with various sized training subsets of the WeeBit

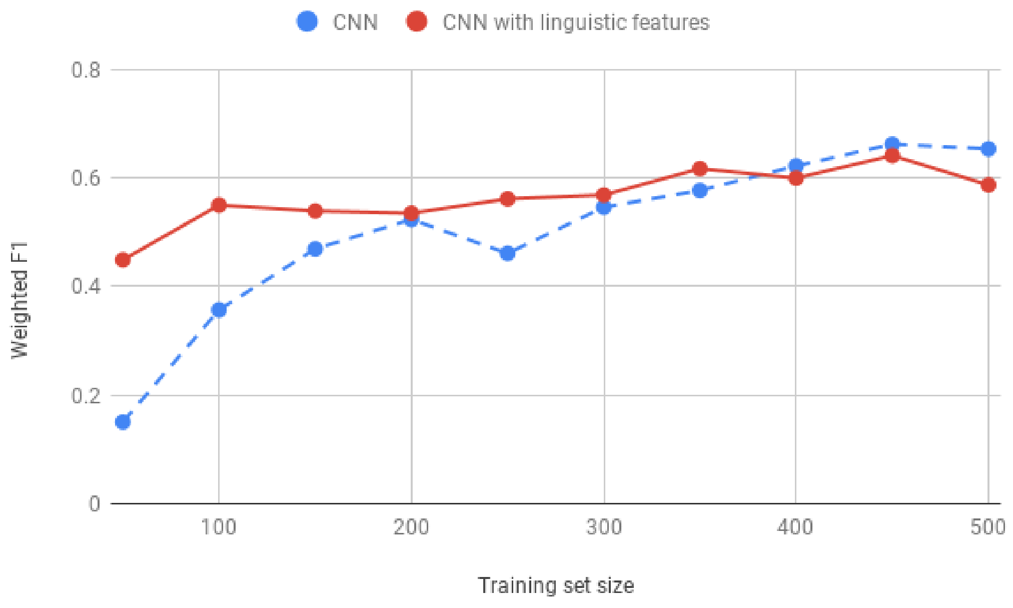

Figure 4.1: Performance differences across different training set sizes on the downsampled WeeBit corpus

corpus. The macro F1 at these various dataset sizes is shown in figure 4.1. Across the trials at different training set sizes, the test set is held constant thereby isolating the impact of training set size.

The hypothesis holds true for extremely small subsets of training data, those with fewer than 200 documents. Above this training set size, the addition of linguistic features results in insubstantial changes in performance. Thus, either the patterns exposed by the linguistic features are learnable with very little data or the patterns extracted by deep learning models differ significantly from the linguistic features. The latter appears more likely given that linguistic features are shown to improve performance for certain corpora (Newsela) and model types (transformers).

This result indicates that the use of linguistic features should be considered for small datasets. However, the dataset size at which those features lose utility is extremely small. Therefore, collecting additional data would often be more efficient than investing the time to incorporate linguistic features.

Feature	Weight
Coleman-Liau formula	1.437
POSD$_{dev}$	-1.245
SBARs / sentence	-0.7101
Subtrees / sentence	-0.6851
Average sentence length	-0.6402
Number of characters	-0.5928
Automated readability index	-0.5130
Squared verb variation ($\frac{(\text{non-modal verbs - modal verbs})^2}{\text{unique non-modal verbs}}$)	-0.5078
Clauses per sentence	0.4771
Average parse tree height / sentence	0.4721

Table 4.5: 10 most predictive features for a linear model on the downsampled WeeBit corpus

4.3.2 Feature Significance

Unsurprisingly, different linguistic features provide differing amounts of predictive power. Table 4.5 and table 4.6 show the predictive power of the 10 most predictive features where predictive power is determined by the feature's coefficient in a linear model. For both corpora, the Coleman-Liau formula (Coleman and Liau 1975) received the highest weight. This weight indicates that it is well calibrated to predict readability levels. Its consistently high weight may also indicate that it was used in the construction of the corpora, especially considering that other feature formulas received much smaller weights. The Coleman-Liau formula is similar to the Flesch score differing by discarding syllable counts in favor of character counts. The POSD$_{dev}$ feature also proved highly predictive, indicating that syntactic variety is a good indicator of readability. The prominence of features like the Coleman-Liau formula, average sentence length, and number of characters demonstrates that shallow features capture a significant portion of document readability.

The ranking and weights for novel features introduced in this thesis are shown in table 4.7 and table 4.8. Among the novel features introduced in this thesis POSD$_{dev}$ proved by far the most predictive while the other novel features had more average ranks. POS_{div} was the 54th and 52nd most predictive feature for the WeeBit and Newsela corpora respectively providing minor predictive power. Similarly, PD_2, PD_{10}, and PDM_{10} were sometimes minor

Feature	Weight
Coleman-Liau formula	3.149
POSD$_{dev}$	-1.879
Number of characters	-1.872
Average sentence length	-1.867
Automated Readability Index	-1.570
Subtrees / sentence	-1.557
Words / clauses	1.303
Clauses / t-units	1.173
Prepositional Phrases / sentence	-1.061
Corrected verb variation $(\frac{\text{non-modal verbs - modal verbs}}{\sqrt{2(\text{unique non-modal verbs})}})$	0.9994

Table 4.6: 10 most predictive features for a linear model on the Newsela corpus

Feature	Rank	Weight
POSD$_{dev}$	2	-1.245
PD_2	33	-0.2207
PD_{10}	36	-0.2093
POS_{div}	54	-0.1461
PDM_{10}	80	0.03748

Table 4.7: Ranking and weights of novel features in linear model on the downsampled WeeBit corpus

predictors for the corpora.

Interestingly, these most predictive features are not shared with the models created by Vajjala Balakrishna (2015) despite the use of nearly identical features. This may indicate that the relative importance of such features is heavily dependent on model type. If this hypothesis is true, many of these features carry redundant information and the differing models simply extract this information in different ways.

Given the differing predictive power of the features, it is elucidating to analyze how differ-

Feature	Rank	Weight
POSD$_{dev}$	2	1.879
PD_{10}	41	0.2028
PDM_{10}	48	0.1626
POS_{div}	52	0.1430
PD_2	63	0.08512

Table 4.8: Ranking and weights of novel features in linear model on the Newsela corpus

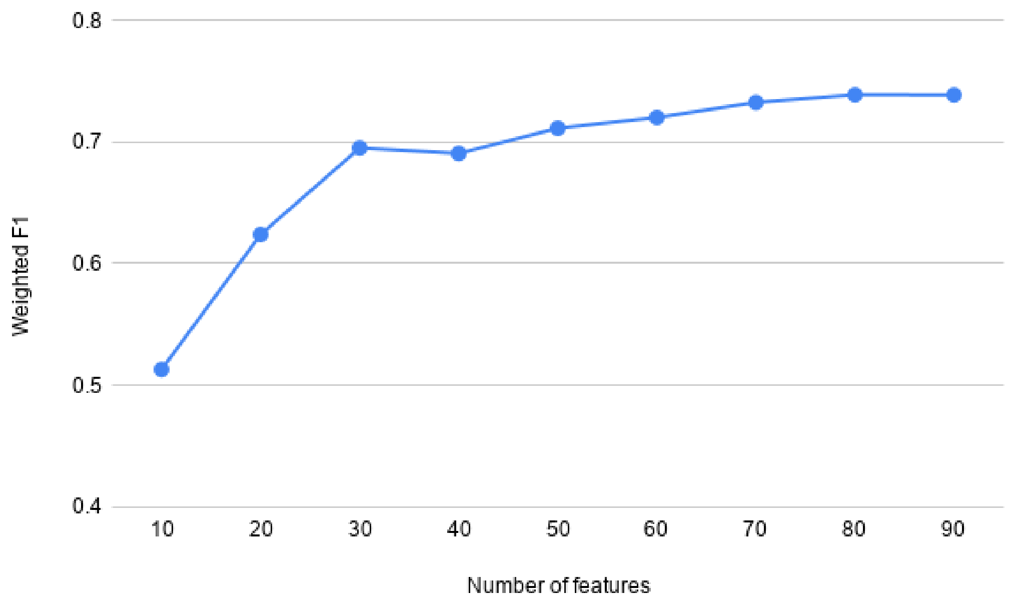

Figure 4.2: Linear model performance with increasing number of features on the downsampled WeeBit corpus

ing numbers of features affect model performance. For both the Newsela and downsampled WeeBit corpora, a linear model was tested with the top 10 most predictive features, top 20 most predictive features, and so on. The performance of these differing models is shown in figure 4.2 and figure 4.3. For the downsampled WeeBit corpus, the top 10 most predictive features achieve an F1 score of about 0.5. Substantial performance increases occur until 30 features after which the improvement is more gradual. For the Newsela corpus, substantial performance increases occur between 10 and 50 features. After this point, performance largely remains the same indicating that the features beyond the top 50 are of little to no use for this corpus and model.

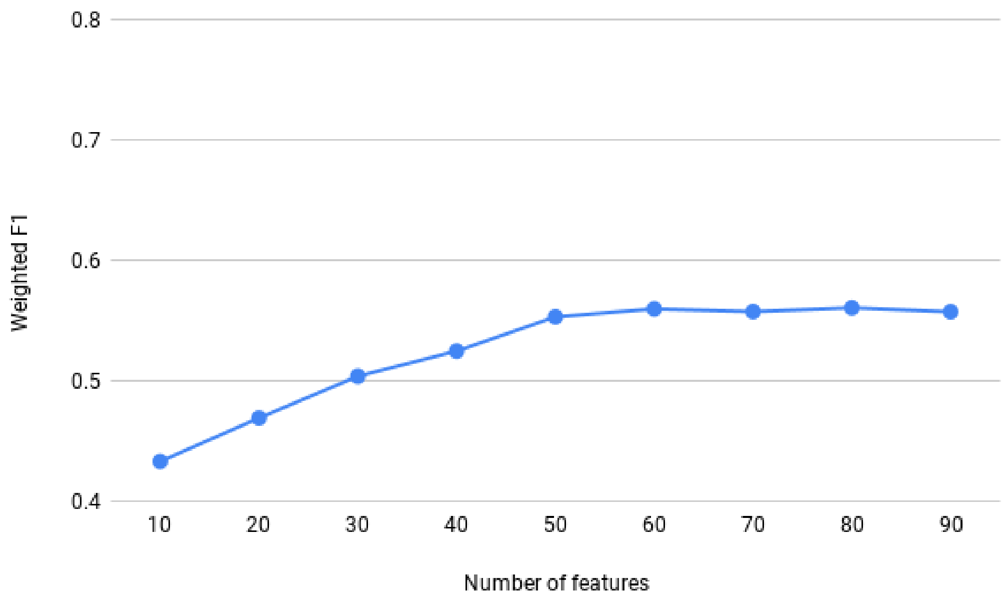

Figure 4.3: Linear model performance with increasing number of features on the Newsela corpus

Chapter 5

Conclusion

In this thesis I explored the role of linguistic features in deep learning methods for readability assessment. I constructed linguistic features focused on syntactic properties ignored by existing features. I incorporated these features into a variety of model types, both those commonly used in readability research and more modern deep learning methods. I evaluated these models on two distinct corpora that posed different sets of challenges for readability assessment. While linguistic features occasionally improved model performance, these models did not achieve state-of-the-art performance.

Although my current work supports disusing linguistic features in readability assessment models, this assertion is limited by available corpora. Specifically, ambiguity in the corpora construction methodology limits our ability to know the consistency and validity of their labels. This uncertainty prevents us from knowing the maximal possible performance on these corpora. Therefore, the maximal possible performance may already be achieved by state-of-the-art models. Thus, future work should explore constructing and evaluating readability corpora with rigorous consistent methodology; such corpora may be assessed most effectively using linguistic features. For instance, current corpora could be reanalyzed to measure human consistency and performance in replicating existing labels. Similarly, construction of new corpora could involve assigning labels as an average across multiple labelers. Additionally, an educator and researcher focused labeling methodology may fail to accurately assess reading difficulty. Instead, reader-focused labeling may be most effective (Vajjala and Lucic 2019). For instance, this labeling could involve asking learning readers to assess whether a given work was too easy, too difficult, or appropriate. Ultimately, the goal of readability assessment is to provide appropriately difficult material for learning readers; supposing those readers may be best equipped to assess such difficulty is a compelling hypothesis. Future systems could even be tailored to each individual learner, shaped by the knowledge of what they have already read and their previous assessments.

Overall, linguistic features do not appear to be useful for readability assessment. While often used in traditional readability assessment models, these features fail to improve the performance of deep learning methods. This assertion is limited by available corpora and our knowledge of their construction. Thus, this thesis provides a starting point to understanding the qualities and abilities of deep learning models in comparison to linguistic features. Through this comparison, we can analyze what types of information these models are well-suited to learning. Using this information could expand model learning abilities and ultimately lead to improvements in readability assessment.

Appendix A

Full Model Results

Features	Weighted F1	Macro F1	RMSE	SD weighted F1	SD macro F1	SD RSME
Linear classifier with Flesch Score	0.2147	0.2156	1.9024	0.0347	0.0253	0.0640
Linear classifier with Flesch features	0.3973	0.3976	1.3342	0.0154	0.0087	0.0492
SVM with HAN	0.5531	0.5499	1.1244	0.1944	0.1928	0.4021
SVM with Flesch features	0.5908	0.5905	1.0657	0.0157	0.0168	0.0462
SVM with CNN ordered class regression	0.6703	0.6700	0.6336	0.0360	0.0334	0.0529
SVM with CNN age regression	0.6743	0.6742	0.6308	0.0339	0.0314	0.0511
Linear classifier with word types	0.7202	0.7189	0.7638	0.0063	0.0085	0.0275
SVM with CNN ordered classes regression, and linguistic features	0.7265	0.7262	0.5820	0.0326	0.0297	0.0404
Logistic regression classification with word types, Flesch features, and linguistic features	0.7382	0.7376	0.7033	0.0710	0.0684	0.1608
SVM with CNN age regression and linguistic features	0.7384	0.7376	0.5672	0.0361	0.0346	0.0491
HAN	0.7507	0.7501	0.7014	0.0306	0.0302	0.0934
SVM with linguistic features and Flesch features	0.7664	0.7667	0.7011	0.0109	0.0114	0.0302
SVM with linguistic features	0.7665	0.7666	0.7022	0.0146	0.0153	0.0401
CNN	0.7859	0.7852	0.6122	0.0171	0.0166	0.0396
SVM with HAN and linguistic features	0.7862	0.7864	0.6487	0.0631	0.0633	0.1509
SVM with CNN classifier	0.7882	0.7879	0.5937	0.0217	0.0195	0.0525
Logistic regression with word types	0.7894	0.7887	0.5576	0.0151	0.0202	0.0234
Logistic regression classification with word types and word count	0.7908	0.7899	0.5529	0.0130	0.0182	0.0198
SVM with CNN classifier and linguistic features	0.7923	0.7919	0.5894	0.0210	0.0193	0.0612
Logistic regression classification with word types, word count, and Flesch features	0.7934	0.7926	0.5487	0.0135	0.0187	0.0166
Logistic regression with word types, Flesch features, and linguistic features	0.8135	0.8130	0.5127	0.0131	0.0169	0.0130
SVM with transformer	0.8343	0.8340	0.4617	0.0131	0.0135	0.0320
SVM with transformer and linguistic features	0.8344	0.8347	0.4559	0.0106	0.0091	0.0134
SVM with transformer and Flesch features	0.8359	0.8358	0.4644	0.0151	0.0154	0.0216
SVM with transformer, Flesch features, and linguistic features	0.8381	0.8377	0.4567	0.0128	0.0118	0.0230
Transformer	0.8387	0.8388	0.4537	0.0097	0.0073	0.0129

Table A.1: WeeBit downsampled model results sorted by weighted F1 score

Features	Weighted F1	Macro F1	RMSE	SD weighted F1	SD Macro F1	SD RSME
Linear classifier with Flesch Score	0.3357	0.1816	1.9700	0.0243	0.0079	0.0484
SVM with HAN	0.3625	0.2134	1.9618	0.0400	0.0331	0.0996
Linear classifier with Flesch features	0.3939	0.2639	1.7498	0.0239	0.0305	0.0990
SVM with Flesch features	0.4776	0.3609	1.4724	0.0222	0.0190	0.0594
SVM with CNN age regression	0.7279	0.6431	0.5832	0.0198	0.0205	0.0309
SVM with CNN ordered class regression	0.7316	0.6482	0.5815	0.0142	0.0141	0.0240
SVM with CNN age regression and linguistic features	0.7779	0.7088	0.5313	0.0156	0.0194	0.0284
SVM with CNN ordered classes regression, and linguistic features	0.7797	0.7114	0.5322	0.0130	0.0120	0.0160
Linear classifier with word types	0.7821	0.7109	0.6715	0.0162	0.0127	0.0210
SVM with Linguistic features and Flesch features	0.7952	0.7367	0.7176	0.0121	0.0157	0.0108
SVM with Linguistic features	0.7952	0.7366	0.7181	0.0130	0.0164	0.0142
HAN	0.8065	0.7435	0.5968	0.0123	0.0220	0.0289
Logistic regression classification with word types	0.8088	0.7497	0.5725	0.0127	0.0152	0.0272
Logistic regression classification with word types and word count	0.8088	0.7497	0.5720	0.0121	0.0148	0.0265
Logistic regression classification with word types, word count, and Flesch features	0.8098	0.7505	0.5667	0.0130	0.0163	0.0266
Logistic regression classification with word types, Flesch features, and linguistic features	0.8206	0.7664	0.5727	0.0428	0.0500	0.1418
CNN	0.8282	0.7748	0.5609	0.0211	0.0183	0.0405
SVM with CNN classifier and linguistic features	0.8286	0.7753	0.5546	0.0222	0.0209	0.0470
Logistic regression classification with word types, Flesch features, and ling features	0.8293	0.7760	0.5192	0.0152	0.0172	0.0323
SVM with CNN classifier	0.8296	0.7754	0.5500	0.0163	0.0136	0.0430
SVM with HAN and linguistic features	0.8441	0.7970	0.6302	0.0643	0.0827	0.2253
SVM with transformer, Flesch features, and linguistic features	0.8721	0.8273	0.4133	0.0095	0.0121	0.0130
Transformer	0.8721	0.8272	0.4013	0.0071	0.0102	0.0184
SVM with transformer	0.8729	0.8288	0.4208	0.0064	0.0090	0.0225
SVM with transformer and Flesch features	0.8746	0.8305	0.4166	0.0054	0.0107	0.0260
SVM with transformer and linguistic features	0.8769	0.8343	0.4041	0.0077	0.0129	0.0199

Table A.2: WeeBit model results sorted by weighted F1 score

Features	Weighted F1	Macro F1	RMSE	SD weighted F1	SD Macro F1	SD RSME
Linear classifier with Flesch Score	0.1668	0.0915	3.4103	0.0055	0.0043	0.0310
SVM with Flesch score	0.2653	0.1860	2.5729	0.0053	0.0086	0.0174
Logistic regression with word types	0.2964	0.2030	1.4384	0.0144	0.0103	0.0229
Logistic regression with word types and word count	0.2969	0.2039	1.4386	0.0145	0.0095	0.0291
Logistic regression with word types, word count, and Flesch features	0.3006	0.2097	1.4346	0.0139	0.0088	0.0259
Linear classifier with Flesch features	0.3080	0.2060	2.1705	0.0110	0.0077	0.0545
Logistic regression with word types, Flesch features, and linguistic features	0.3333	0.2489	1.4070	0.0118	0.0162	0.0335
Linear classifier with word types	0.3368	0.2485	1.4254	0.0089	0.0153	0.0299
CNN	0.3379	0.2574	1.9686	0.0038	0.0111	0.0506
SVM with CNN classifier	0.3407	0.2616	2.0142	0.0079	0.0142	0.0817
SVM with CNN ordered class regression	0.5207	0.4454	1.4464	0.0092	0.0193	0.0284
SVM with CNN age regression	0.5223	0.4469	1.4539	0.0149	0.0244	0.0365
SVM with transformer	0.5430	0.4711	1.3828	0.0095	0.0258	0.0336
Transformer	0.5435	0.4713	1.3885	0.0106	0.0264	0.0369
Linear classifier with linguistic features	0.5573	0.4748	1.3734	0.0053	0.0140	0.0358
SVM with CNN classifier, and linguistic features	0.7058	0.5510	1.1107	0.0079	0.0357	0.0333
SVM with Flesch features	0.7177	0.6257	1.2106	0.0079	0.0292	0.0113
SVM with transformer and Flesch features	0.7186	0.6305	1.1271	0.0074	0.0282	0.0290
SVM with CNN ordered classes regression and linguistic features	0.7231	0.6053	1.2039	0.0062	0.0331	0.0280
SVM with CNN age regression and linguistic features	0.7281	0.6104	1.1856	0.0057	0.0337	0.0347
SVM with linguistic features	0.7582	0.6432	1.0695	0.0089	0.0379	0.0369
SVM with transformer, Flesch features, and linguistic features	0.7627	0.6263	1.0615	0.0075	0.0301	0.0352
SVM with transformer and linguistic features	0.7678	0.6656	1.0231	0.0230	0.0385	0.1157
SVM with linguistic features and Flesch Features	0.7694	0.6446	1.0493	0.0060	0.0406	0.0306
SVM with HAN	0.7931	0.6724	0.5562	0.0448	0.0449	0.0868
SVM with HAN and linguistic features	0.8014	0.6751	0.6426	0.0263	0.0379	0.2346
HAN	0.8024	0.6775	0.4897	0.1116	0.1825	0.2756

Table A.3: Newsela model results sorted by weighted F1 score

Features	Weighted F1	Macro F1	RMSE	SD weighted F1	SD Macro F1	SD RSME
SVM with Flesch score	0.2638	0.1780	2.5747	0.0094	0.0044	0.0411
Linear classifier with Flesch features	0.3099	0.2043	2.1684	0.0161	0.0220	0.0584
CNN	0.4111	0.3263	1.8183	0.0116	0.0135	0.1238
SVM with CNN classifier	0.4148	0.3322	1.7931	0.0095	0.0199	0.1040
SVM with CNN age regression	0.4701	0.3901	1.5044	0.0114	0.0225	0.0283
SVM with CNN ordered class regression	0.4763	0.3976	1.5057	0.0115	0.0164	0.0289
Linear classifier with word types	0.4876	0.3795	1.3509	0.0105	0.0193	0.0332
Logistic regression classification with word types (non-normalized)	0.5120	0.3903	1.2144	0.0114	0.0228	0.0385
Logistic regression with word types	0.5134	0.3887	1.2475	0.0052	0.0245	0.0281
Logistic regression with word types and word count	0.5153	0.3916	1.2424	0.0072	0.0266	0.0330
Logistic regression with word types, word count, and Flesch features	0.5176	0.3948	1.2292	0.0078	0.0228	0.0414
SVM with transformer	0.5273	0.4462	1.3925	0.0076	0.0182	0.0348
Transformer	0.5278	0.4444	1.3846	0.0067	0.0204	0.0299
Logistic regression with word types, Flesch features, and linguistic features	0.5589	0.4399	1.0917	0.0119	0.0290	0.0486
Linear classifier with linguistic features	0.5652	0.4729	1.3454	0.0103	0.0215	0.0692
SVM with CNN age regression and linguistic features	0.7029	0.5849	1.1202	0.0159	0.0328	0.0249
SVM with CNN ordered classes regression, and linguistic features	0.7050	0.5881	1.1322	0.0156	0.0295	0.0297
SVM with CNN classifier, and linguistic features	0.7062	0.5571	1.1164	0.0157	0.0291	0.0642
SVM with transformer and Flesch features	0.7074	0.6063	1.1093	0.0141	0.0370	0.0188
SVM with Flesch features	0.7184	0.6133	1.2024	0.0063	0.0266	0.0265
HAN	0.7487	0.6203	0.6193	0.0149	0.0138	0.0389
SVM with transformer, Flesch features, and linguistic features	0.7556	0.6319	1.0547	0.0103	0.0305	0.0310
SVM with transformer and linguistic features	0.7569	0.6441	1.0645	0.0087	0.0328	0.0345
SVM with HAN and linguistic features	0.7611	0.6331	1.0610	0.0126	0.0328	0.0426
SVM with Linguistic features	0.7639	0.6252	1.0494	0.0102	0.0305	0.0391
SVM with linguistic features and Flesch Features	0.7673	0.6277	1.0420	0.0095	0.0315	0.0334
SVM with HAN	0.7760	0.6520	0.5888	0.0769	0.0553	0.1248

Table A.4: Newsela document set model results sorted by weighted F1 score

Bibliography

Abadi, Martín et al. (2015). *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. URL: `https://www.tensorflow.org/`.

Baayen, R. Harald, Richard Piepenbrock, and Leon Gulikers (1995). *The CELEX Lexical Database*.

Bies, Ann et al. (Apr. 16, 1995). *Bracketing Guidelines For Treebank II Style Penn Treebank Project*.

Chinchor, Nancy (1992). "MUC-4 Evaluation Metrics". In: *Proceedings of the 4th Conference on Message Understanding*. MUC4 '92. McLean, Virginia: Association for Computational Linguistics, pp. 22–29. ISBN: 978-1-55860-273-1. DOI: `10.3115/1072064.1072067`. URL: `https://doi.org/10.3115/1072064.1072067` (visited on 12/07/2019).

Chollet, François et al. (2015). *Keras*. URL: `https://keras.io`.

Coleman, Meri and T. L. Liau (1975). "A Computer Readability Formula Designed for Machine Scoring". In: *Journal of Applied Psychology* 60.2, pp. 283–284. ISSN: 1939-1854(Electronic),0021-9010(Print). DOI: `10.1037/h0076540`.

Conneau, Alexis et al. (July 2018). "What You Can Cram into a Single $&!#* Vector: Probing Sentence Embeddings for Linguistic Properties". In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. ACL 2018. Melbourne, Australia: Association for Computational Linguistics, pp. 2126–2136. DOI: `10.18653/v1/P18-1198`. URL: `https://www.aclweb.org/anthology/P18-1198` (visited on 03/06/2020).

Cortes, Corinna and Vladimir Vapnik (Sept. 1, 1995). “Support-Vector Networks”. In: *Machine Learning* 20.3, pp. 273–297. ISSN: 1573-0565. DOI: 10.1023/A:1022627411411. URL: https://doi.org/10.1023/A:1022627411411 (visited on 10/22/2019).

Devlin, Jacob et al. (June 2019). “BERT: Pre-Training of Deep Bidirectional Transformers for Language Understanding”. In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. NAACL-HLT 2019. Minneapolis, Minnesota: Association for Computational Linguistics, pp. 4171–4186. DOI: 10.18653/v1/N19-1423. URL: https://www.aclweb.org/anthology/N19-1423 (visited on 03/06/2020).

Fauske, Kjell (Dec. 7, 2006). *Neural Network — TikZ Example*. URL: http://www.texample.net/tikz/examples/neural-network/ (visited on 10/22/2019).

Feng, Lijun (2010). “Automatic Readability Assessment”. New York, NY, USA: City University of New York.

Flesch, Rudolph (1948). “A New Readability Yardstick”. In: *Journal of Applied Psychology* 32.3, pp. 221–233. ISSN: 1939-1854(Electronic),0021-9010(Print). DOI: 10.1037/h0057532.

Gunning, Robert (1952). *The Technique of Clear Writing*. McGraw-Hill. 312 pp. Google Books: ofI0AAAAMAAJ.

Hsu, Chih-wei, Chih-chung Chang, and Chih-Jen Lin (Nov. 2003). *A Practical Guide to Support Vector Classification*.

Joachims, Thorsten (1998). “Text Categorization with Support Vector Machines: Learning with Many Relevant Features”. In: *Machine Learning: ECML-98*. Ed. by Claire Nédellec and Céline Rouveirol. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, pp. 137–142. ISBN: 978-3-540-69781-7. DOI: 10.1007/BFb0026683.

Kim, Yoon (Oct. 2014). “Convolutional Neural Networks for Sentence Classification”. In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. EMNLP 2014. Doha, Qatar: Association for Computational Linguistics, pp. 1746–1751. DOI: `10.3115/v1/D14-1181`. URL: `https://www.aclweb.org/anthology/D14-1181` (visited on 03/06/2020).

Kincaid, J. P. et al. (Feb. 1, 1975). *Derivation of New Readability Formulas (Automated Readability Index, Fog Count and Flesch Reading Ease Formula) for Navy Enlisted Personnel:* Fort Belvoir, VA: Defense Technical Information Center. DOI: `10.21236/ADA006655`. URL: `http://www.dtic.mil/docs/citations/ADA006655` (visited on 05/02/2019).

Klein, Dan and Christopher D. Manning (2003). "Accurate Unlexicalized Parsing". In: *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - ACL '03*. The 41st Annual Meeting. Vol. 1. Sapporo, Japan: Association for Computational Linguistics, pp. 423–430. DOI: `10.3115/1075096.1075150`. URL: `http://portal.acm.org/citation.cfm?doid=1075096.1075150` (visited on 05/01/2019).

Koehrsen, Will (2019). *Beyond Accuracy: Precision and Recall*. URL: `https://towardsdatascience.com/beyond-accuracy-precision-and-recall-3da06bea9f6c` (visited on 12/07/2019).

Kullback, S. and R. A. Leibler (Mar. 1951). "On Information and Sufficiency". In: *The Annals of Mathematical Statistics* 22.1, pp. 79–86. ISSN: 0003-4851, 2168-8990. DOI: `10.1214/aoms/1177729694`. URL: `https://projecteuclid.org/euclid.aoms/1177729694` (visited on 05/01/2019).

Lu, Xiaofei (Nov. 17, 2010). "Automatic analysis of syntactic complexity in second language writing". In: *International Journal of Corpus Linguistics* 15.4, pp. 474–496. ISSN: 1384-6655. DOI: `10.1075/ijcl.15.4.02lu`.

— (2011). "A Corpus-Based Evaluation of Syntactic Complexity Measures as Indices of College-Level ESL Writers' Language Development". In: *TESOL Quarterly* 45.1, pp. 36–62. ISSN: 00398322. URL: `http://www.jstor.org.ezp-prod1.hul.harvard.edu/stable/41307615`.

Magpie (Mar. 4, 2020). inspirehep. URL: `https://github.com/inspirehep/magpie` (visited on 03/06/2020).

Martinc, Matej, Senja Pollak, and Marko Robnik-Šikonja (July 31, 2019). “Supervised and Unsupervised Neural Approaches to Text Readability”. URL: http://arxiv.org/abs/1907.11779 (visited on 11/15/2019).

McCall, William Anderson and Lelah Mae Crabbs (1926). *Standard Test Lessons in Reading*. Vol. Books II, III, IV, and V. New York: Bureau of Publications, Teachers College, Columbia University.

Mikolov, Tomas, Kai Chen, and Corrado (Jan. 16, 2013). “Efficient Estimation of Word Representations in Vector Space”. In: *International Conference on Learning Representations: Workshops Track*. arXiv: 1301.3781 [cs]. URL: http://arxiv.org/abs/1301.3781 (visited on 10/20/2019).

Naidu, Ashwin (Feb. 8, 2020). *Image Filtering*. URL: https://github.com/ashushekar/image-convolution-from-scratch (visited on 02/27/2020).

National Governors Association Center for Best Practices and Council of Chief State School Officers (2010). *Common Core State Standards For English Language Arts & Literacy in History/Social Studies, Science, and Technical Subjects Appendix A*. Washington D.C.: Common Core State Standards Initiative.

Nguyen, Viet (Feb. 28, 2020). *Hierarchical Attention Networks for Document Classification*. URL: https://github.com/uvipen/Hierarchical-attention-networks-pytorch (visited on 03/02/2020).

OECD (2016). *Skills Matter*. URL: https://www.oecd-ilibrary.org/content/publication/9789264258051-en.

Pedregosa, Fabian et al. (Oct. 2011). “Scikit-Learn: Machine Learning in Python”. In: *Journal of Machine Learning Research* 12, 2825-2830. ISSN: 1533-7928. URL: http://jmlr.csail.mit.edu/papers/v12/pedregosa11a.html (visited on 12/07/2019).

Readable Language in Insurance Policies. (1982). URL: http://edocs.dlis.state.fl.us/fldocs/leg/actsflorida/1982/1982V1Pt2.pdf.

Reutzel, D. Ray et al. (2008). “Scaffolded Silent Reading: A Complement to Guided Repeated Oral Reading That Works!” In: *The Reading Teacher* 62.3, pp. 194–207. ISSN: 1936-2714. DOI: 10.1598/RT.62.3.2. URL: http://ila.onlinelibrary.wiley.com/doi/abs/10.1598/RT.62.3.2 (visited on 02/03/2020).

Schwarm, Sarah E. and Mari Ostendorf (2005). “Reading Level Assessment Using Support Vector Machines and Statistical Language Models”. In: *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*. ACL ’05. Ann Arbor, Michigan: Association for Computational Linguistics, pp. 523–530. DOI: 10.3115/1219840.1219905. URL: https://doi.org/10.3115/1219840.1219905 (visited on 04/05/2019).

Smith, D. Randall (1989). *The Lexile Scale in Theory and Practice. Final Report.*

Smith, E. A. and R. J. Senter (May 1967). “Automated Readability Index”. In: *AMRL-TR. Aerospace Medical Research Laboratories (U.S.)*, pp. 1–14. pmid: 5302480.

Trask, Andrew, Phil Michalak, and John Liu (Nov. 19, 2015). “Sense2vec - A Fast and Accurate Method for Word Sense Disambiguation In Neural Word Embeddings”. URL: http://arxiv.org/abs/1511.06388 (visited on 10/21/2019).

Vajjala Balakrishna, Sowmya (Aug. 3, 2015). “Analyzing Text Complexity and Text Simplification: Connecting Linguistics, Processing and Educational Applications”. Dissertation. Universität Tübingen. DOI: http://dx.doi.org/10.15496/publikation-5781. URL: https://publikationen.uni-tuebingen.de/xmlui/handle/10900/64359 (visited on 10/17/2019).

Vajjala, Sowmya and Ivana Lucic (Aug. 2019). “On Understanding the Relation between Expert Annotations of Text Readability and Target Reader Comprehension”. In: *Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications*. Florence, Italy: Association for Computational Linguistics, pp. 349–359. DOI: 10.18653/v1/W19-4437. URL: https://www.aclweb.org/anthology/W19-4437 (visited on 03/06/2020).

Vajjala, Sowmya and Detmar Meurers (June 2012). "On Improving the Accuracy of Readability Classification Using Insights from Second Language Acquisition". In: *Proceedings of the Seventh Workshop on Building Educational Applications Using NLP*. Montréal, Canada: Association for Computational Linguistics, pp. 163–173. URL: https://www.aclweb.org/anthology/W12-2019 (visited on 05/01/2019).

— (Jan. 1, 2014). "Readability Assessment for Text Simplification: From Analyzing Documents to Identifying Sentential Simplifications". In: *International Journal of Applied Linguistics, Special Issue on Current Research in Readability and Text Simplification* 165. DOI: 10.1075/itl.165.2.04vaj.

— (Mar. 18, 2016). *Readability-Based Sentence Ranking for Evaluating Text Simplification.* arXiv: 1603.06009. URL: http://arxiv.org/abs/1603.06009 (visited on 11/13/2019).

Vaswani, Ashish et al. (2017). "Attention Is All You Need". In: *Advances in Neural Information Processing Systems 30*. Ed. by I. Guyon et al. Curran Associates, Inc., pp. 5998–6008. URL: http://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf.

Wilson, Michael (Jan. 1, 1988). "MRC Psycholinguistic Database: Machine-Usable Dictionary, Version 2.00". In: *Behavior Research Methods, Instruments, & Computers* 20.1, pp. 6–10. ISSN: 1532-5970. DOI: 10.3758/BF03202594. URL: https://doi.org/10.3758/BF03202594 (visited on 12/06/2019).

Wolf, Thomas et al. (2019). *HuggingFace's Transformers: State-of-the-Art Natural Language Processing*. URL: https://arxiv.org/abs/1910.03771.

Xia, Menglin, Ekaterina Kochmar, and Ted Briscoe (June 2016). "Text Readability Assessment for Second Language Learners". In: *Proceedings of the 11th Workshop on Innovative Use of NLP for Building Educational Applications*. San Diego, CA: Association for Computational Linguistics, pp. 12–22. DOI: 10.18653/v1/W16-0502. URL: https://www.aclweb.org/anthology/W16-0502 (visited on 12/06/2019).

Xu, Wei, Chris Callison-Burch, and Courtney Napoles (2015). "Problems in Current Text Simplification Research: New Data Can Help". In: *Transactions of the Association for*

Computational Linguistics 3, pp. 283–297. DOI: `10.1162/tacl_a_00139`. URL: `https://www.aclweb.org/anthology/Q15-1021` (visited on 01/24/2020).

Yang, Zichao et al. (June 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*. NAACL-HLT 2016. San Diego, California: Association for Computational Linguistics, pp. 1480–1489. DOI: `10.18653/v1/N16-1174`. URL: `https://www.aclweb.org/anthology/N16-1174` (visited on 01/22/2020).

Zhu, Y. et al. (Dec. 2015). "Aligning Books and Movies: Towards Story-like Visual Explanations by Watching Movies and Reading Books". In: *2015 IEEE International Conference on Computer Vision (ICCV)*, pp. 19–27. DOI: `10.1109/ICCV.2015.11`. URL: `https://arxiv.org/abs/1506.06724`.