Pushing on Text Readability Assessment: A Transformer Meets Handcrafted Linguistic Features

Bruce W. Lee[1,3]
Univ. of Pennsylvania[1]
PA, USA
brucelws@seas.upenn.edu

Yoo Sung Jang[2,3]
Univ. of Wisconsin-Madison[2]
WI, USA
yjang43@wisc.edu

Jason Hyung-Jong Lee[3]
LXPER AI[3]
Seoul, South Korea
jasonlee@lxper.com

Abstract

We report two essential improvements in readability assessment: 1. three novel features in advanced semantics and 2. the timely evidence that traditional ML models (e.g. Random Forest, using handcrafted features) can combine with transformers (e.g. RoBERTa) to augment model performance. First, we explore suitable transformers and traditional ML models. Then, we extract 255 handcrafted linguistic features using self-developed extraction software. Finally, we assemble those to create several hybrid models, achieving state-of-the-art (SOTA) accuracy on popular datasets in readability assessment. The use of handcrafted features help model performance on smaller datasets. Notably, our RoBERTA-RF-T1 hybrid achieves the near-perfect classification accuracy of 99%, a 20.3% increase from the previous SOTA.

1 Introduction

The long quest for advancing readability assessment (RA) mostly centered on handcrafting the linguistic features that affect readability (Pitler and Nenkova, 2008). RA is a time-honored branch of natural language processing (NLP) that quantifies the difficulty with which a reader understands a text (Feng et al., 2010). Being one of the oldest systematic approaches to linguistics (Collins-Thompson, 2014), RA developed various linguistic features. These range from simple measures like the average count of syllables to those as sophisticated as semantic complexity (Buchanan et al., 2001).

Perhaps due to the abundance of dependable linguistic features, an overwhelming majority of RA systems are Support Vector Machines (SVM) with handcrafted features (Hansen et al., 2021). Such traditional machine learning (ML) methods were linguistically explainable, expandable, and most importantly, competent against the modern neural models. As a fragmentary example, Filighera et al. (2019) reports that a large ensemble of 6 BiLSTMs with BERT (Devlin et al., 2019), ELMo (Peters et al., 2018), Word2Vec (Mikolov et al., 2013), and GloVe (Pennington et al., 2014) embeddings showed only $\sim$1% accuracy improvement from a single SVM model developed by Xia et al. (2016).

Even though deep neural networks have achieved state-of-the-art (SOTA) performance in almost all semantic tasks where sufficient data were available (Collobert et al., 2011; Zhang et al., 2015), neural models started showing promising results in RA only quite recently (Martinc et al., 2021). A known challenge for the researchers in RA is the lack of large public datasets – with the unique exception of WeeBit (Vajjala and Meurers, 2012). Technically speaking, even WeeBit is not entirely public since it has to be directly obtained from the authors.

Martinc et al. (2021) raised the SOTA classification accuracy on the popular WeeBit dataset (Vajjala and Meurers, 2012) by about 4% using BERT. This was the first solid proof that neural models with auto-generated features can show significant improvement compared to traditional ML with handcrafted features. However, neural models, or transformers (which is the interest of this paper), still show not much better performance than traditional ML on smaller datasets like OneStopEnglish (Vajjala and Lučić, 2018), despite the complexity.

From our observations, the reported low performances of transformers on small RA datasets can be accounted for two reasons. 1. Only BERT was applied to RA, and there could be other transformers that perform better, even on small datasets. 2. If a transformer shows weak performance on small datasets, there must be some additional measures done to supply the final model (e.g. ensemble) with more linguistic information, but such a study is rare in RA. Hence, we tackle the abovementioned issues in this paper. In particular, we 1. perform a wide search on transformers, traditional ML models, and handcrafted features & 2. develop a hybrid architecture for SOTA and robustness on small datasets.

However, before we move on to hybrid models, we begin by supplementing an underexplored linguistic branch of handcrafted features. According to survey research on RA (Collins-Thompson, 2014), the study on advanced semantics is scarce. We lack a model to capture how deeper semantic structures affect readability. We attempt to solve this issue by viewing a text as a collection of latent topics and calculating the probability distribution.

Then, we move on to combine traditional ML (w handcrafted features[1]) and transformers. Such a hybrid system is only reported by Deutsch et al. (2020), concluding, "(hybrid models) did not achieve SOTA performance." But we obtain contrary results. Through a large study on the optimal combination, we obtain SOTA results on WeeBit and OneStopEnglish. Also, our BERT-GB-T1 hybrid beats the (previous) SOTA accuracy with only 30% of the full dataset, in section 4.7.

Our main objectives are creating advanced semantic features and hybrid models. But our contributions to academia are not limited to the above-mentioned two. We make the following additions:
1. We numerically represent certain linguistic properties pertaining to advanced semantics.
2. We develop a large-scale, openly available 255 features extraction Python toolkit (which is highly scarce[2] in RA). We name the software **LingFeat**[3].
3. We conduct wide searches and parametrizations on transformers[4] and traditional ML[5] for RA use.
4. We develop hybrid models for SOTA and robustness on small datasets. Notably, RoBERTa-RF-T1 achieves 99% accuracy on OneStopEnglish, 20.3% higher than the previous SOTA (table 5).

2 Advanced Semantics

2.1 Overview

A text is a communication between author and reader, and its readability is affected by the reader having shared world/domain knowledge. According to Collins-Thompson (2014), the features resulting from topic modeling may characterize the deeper semantic structures of a text. These deeper representations accumulate and appear to us in the form of perceivable properties like sentiment and genre. But advanced semantics aims to capture the deeper representation itself.

Among the four branches of linguistic properties (in RA) identified by Collins-Thompson (2014), advanced semantics remain unexplored. Lexico-semantic (Lu, 2011; Malvern and Richards, 2012), syntactic (Heilman et al., 2007; Petersen and Ostendorf, 2009), and discourse-based (McNamara et al., 2010) properties had several notable works but little dealt with advanced semantics as the given definition. The existing examples in higher-level semantics focus on word-level complexity (Collins-Thompson and Callan, 2004; Crossley et al., 2008; Landauer et al., 2011; Nam et al., 2017).

Such a phenomenon is complex. The lack of investigation on advanced semantics could be due to its low correlation with readability. This is plausible because RA studies often test their features on a human-labeled dataset, potentially biased towards easily recognizable surface-level features (Evans, 2006). Such biases could cause low performance.

Further, it must be noted that: 1. world knowledge might not always directly indicate difficulty, and 2. there can be other existing substitute features that capture similar properties on a word level.

S1) *Kindness is good.*

S2) *Christmas is good.*

S3) *I return with the stipulation to dismiss Smith's case; the same being duly executed by me.*

S2 above seems to require more world knowledge than S1. However, "Christmas", as a familiar entity, seems to have no apparent contribution to increased difficulty. If any, similar properties can be captured by word frequency/familiarity measures (lexico-semantics) in a large representative corpus (Leroy and Kauchak, 2013). Also, it seems that S3 is the most difficult, and this can be easily deduced using entity counts (discourse). Entities mostly introduce conceptual information (Feng et al., 2010).

Our key objective in studying advanced semantics is to identify features that add orthogonal information. In other words, we hope to see a performance increase in our overall RA model rather than specific features' high correlations with readability.

Given the considerations, we draw two guidelines: 1. develop passage-level features since most word-level attributes are captured by existing features, and 2. value the orthogonal addition of information, not individual feature's high correlation.

[1]We use "handcrafted features" and "linguistic features" interchangeably throughout this paper.

[2]An exception is Dr. Vajjala's Java toolkit, available at bitbucket.org/nishkalavallabhi/complexity-features.

[3]github.com/brucewlee/lingfeat

[4]github.com/yjang43/pushingonreadability_transformers

[5]github.com/brucewlee/pushingonreadability_traditional_ML

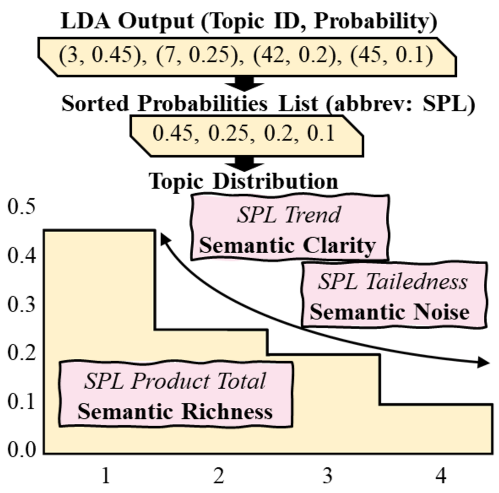

Figure 1: Graphical representation. Semantic Richness, Clarity, and Noise. abbrev: abbreviation.

2.2 Approach

Topics convey text meaning on a global level (Holtgraves, 1999). In order to capture the deeper structure of meaning (advanced semantics), we hypothesized that calculating the distribution of document-topic probabilities from latent dirichlet allocation (LDA) (Blei et al., 2003) could be helpful.

Moreover, domain/world knowledge can be accounted for in LDA-resulting measures since LDA can be trained on various data. As explored in Qumsiyeh and Ng (2011), it may seem sensible to use the count of discovered topics as the measure of required knowledge. However, such measures can be extremely sensitive to passage length. Along with the count of discovered topics, we develop three others that consider how these topics are distributed: semantic richness, clarity, and noise.

Fig. 1 depicts the steps: 1. obtain output from a trained LDA model, 2. ignore topic ID and create a sorted probabilities list, and 3. calculate semantic richness, clarity, and noise. We model "how" the topics are distributed, not "what" the topics are.

2.3 Semantic Richness

Traditionally, semantic richness is quantified according to word usage (Pexman et al., 2008). In a high-dimensional model of semantic space (Li et al., 2000), co-occurring words clustered as semantic neighbors, quantifying semantic richness. As such, the previous models of semantic richness were often studied for word-level complexity and made no explicit connection with readability on a global level. Also, they were often subject-dependent (Buchanan et al., 2001). As concluded by Pexman et al. (2008), semantic richness is defined in several ways. We propose a novel variation.

We apply the similar co-occurrence concept but on the passage level using LDA. Here, semantic richness is the measure of how "largely" populated the topics are. In fig. 1, we approximately define richness as the product total of SPL, which measures the count of discovered topics (n) and topic probability (p). Additionally, we multiply index (i) to reward longer n so that the overall richness increases faster with more topics. See eqn. 1.

$$\text{Semantic Richness} = \sum_{i=1}^{n} p_i \cdot i \qquad (1)$$

2.4 Semantic Clarity

Semantic clarity is critical in understanding text (Peabody and Schaefer, 2016). Likewise, complex meaning structures lead to comprehension difficulty (Pires et al., 2017). Some existing studies quantify semantic complexity (or clarity) through various measures, but most on the fine line between lexical and semantic properties (Collins-Thompson, 2014). They rarely deal with the latent meaning representations or the clarity of the main topic.

For semantic clarity, we quantify how the probability distribution (fig. 1) is focused (skewed) towards the largest discovered topic. In other words, we hope to see how easily identifiable the main topic is. We wanted to adopt the standard skewness equation from statistics, but we developed an alternative (eqn. 2) because the standard equation failed to capture the anticipated trends in appendix A.

$$\text{Semantic Clarity} = \frac{1}{n} \cdot \sum_{i=1}^{n} (max(p) - p_i) \qquad (2)$$

2.5 Semantic Noise

Semantic noise is the measure of the less-important topics (those with low probability), also the "tailedness" of sorted probability lists (fig. 1). A sorted probability list that resembles a (right-halved) leptokurtic curve would have higher semantic noise. In comparison, a (right-halved) platykurtic curve of similar length would have low semantic noise. We adopt the kurtosis equation under Fisher definition (Kokoska and Zwillinger, 2000). See eqn. 3.

$$\text{Semantic Noise} = n \cdot \frac{\sum_{i=1}^{n} (p_i - \bar{p})^4}{(\sum_{i=1}^{n} (p_i - \bar{p})^2)^2} \qquad (3)$$

3 Covered Features

We study 255 linguistic features. For the already existing features, we add variations to widen coverage. The full list of features, feature codes, and definition are provided in appendix B. Also, we classify features into 14 subgroups. External dependencies (e.g. parser) are reported in appendix D.

3.1 Advanced Semantic Features (AdSem)

Here, we follow the methods provided in section 2.

1∼3) Wikipedia (WoKF), WeeBit (WBKF), & OneStop Knowledge Features (OSKF). Each subgroup name represents the respective training data. We train Online LDA (Hoffman et al., 2010) with the 20210301 dump[6] from English Wikipedia for WoKF. The others are trained on two popular corpora in RA: WeeBit and OneStopEnglish.

For each training set, four variations of 50, 100, 150, 200 topics models are trained. Four features – semantic richness, clarity, noise, and the total count of discovered topics – are extracted per model.

3.2 Discourse-Based Features (Disco)

A text is more than a series of random sentences. It indicates a higher-level structure of dependencies.

4) Entity Density Features (EnDF). Conceptual information is often introduced by entities. Hence, the count of entities affects the working memory burden (Feng et al., 2009). We bring entity-related features from Feng et al. (2010).

5) Entity Grid Features (EnGF) Coherent texts are easy to comprehend. Thus, we measure coherence through entity grid, using the 16 transition pattern ratios approach by Pitler and Nenkova (2008) as features. Also, we adopt local coherence scores (Guinaudeau and Strube, 2013), using the code implemented by Palma and Atkinson (2018).

3.3 Syntactic Features (Synta)

Syntactic complexity is associated with longer processing times (Gibson, 1998). Such syntactic properties also affect the overall complexity of a text (Hale, 2016), an important indicator of readability.

6) Phrasal Features (PhrF). Ratios involving clauses correlate with learners' abilities to read (Lu, 2010). We implement several variations, including the counts of noun, verb, and adverb phrases.

7) Tree Structure Features (TrSF). We deal with the structural shape of parsed trees, inspired by the work on average parse tree height by Schwarm and Ostendorf (2005). On a constituency parser (appendix D) output, NLTK (Loper and Bird, 2002) is used for the final calculation of features.

8) Part-of-Speech Features (POSF). Several studies report the effectiveness of using POS counts as features (Tonelli et al., 2012; Lee and Lee, 2020a). We count based on Universal POS tags[7].

3.4 Lexico-Semantic Features (LxSem)

Perhaps the most explored, lexico-semantics capture the attributes associated with the difficulty or unfamiliarity of words (Collins-Thompson, 2014).

9) Variation Ratio Features (VarF) Lu (2011) reports noun, verb, adjective, and adverb variations, which represent the proportion of the respective category's words to total. We implement the features with variants from Vajjala and Meurers (2012).

10) Type Token Ratio Features (TTRF). TTR has been widely used as a measure of lexical richness in language acquisition studies (Malvern and Richards, 2012). We bring five variations of TTR from Vajjala and Meurers (2012). For MTLD (McCarthy and Jarvis, 2010), we default TTR to 0.72.

11) Psycholinguistic Features (PsyF) As explored in Vajjala and Meurers (2016), we implement various Age-of-Acquisition features from Kuperman study database Kuperman et al. (2012).

12) Word Familiarity Features (WorF) Word frequency in a large representative corpus often represents lexical difficulty (Collins-Thompson, 2014) due to unfamiliarity. We use SubtlexUS database (Brysbaert and New, 2009) to measure familiarity.

3.5 Shallow Traditional Features (ShaTr)

Classic readability formulas (e.g. Flesch-Kincaid Grade) (Kincaid et al., 1975) or shallow measures often do not represent a specific linguistic branch.

13) Shallow Features (ShaF) These features capture surface-level difficulty. Our measures include the average count of tokens and syllables.

14) Traditional Formulas (TraF). For Flesh-Kincaid Grade Level, Automated Readability, and Gunning Fog, we follow the "new" formulas in Kincaid et al. (1975). We follow Si and Callan (2001) for Smog Index (Mc Laughlin, 1969). And we follow Eltorai et al. (2015) for Linsear Write.

[6]dumps.wikimedia.org/enwiki

[7]universaldependencies.org/u/pos

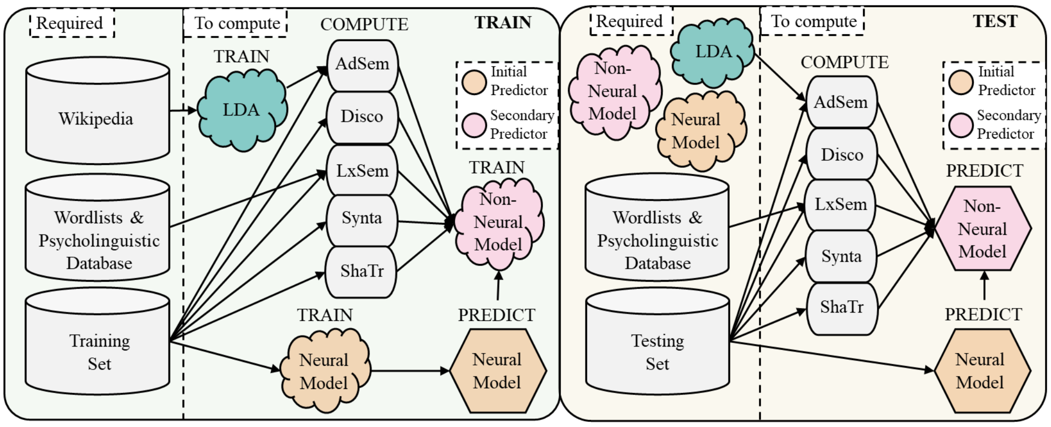

Figure 2: Hybrid model. AdSem, Disco, LxSem, Synta, and ShaTr show handcrafted features' linguistic branches.

4 Hybrid Model

4.1 Overview

As shown in section 3, myriad linguistic properties affect readability. Despite the continual effort at handcrafting features, they lack coverage. Deutsch et al. (2020) hint neural models can better model the linguistic properties for RA task. But the performance/flexibility of neural models could improve.

In our hybrid model, we take a simple approach of joining the soft label predictions of a neural model (e.g. BERT) with handcrafted features and wrapping it with a non-neural model (e.g. SVM).

In fig. 2, the non-neural model (i.e. *secondary predictor*) learns 1. predictions/outputs of the neural model and 2. handcrafted features. The addition of handcrafted features supplements what neural models (i.e. *initial predictor*) might miss, reinforcing performance on the secondary prediction.

4.2 Finding Best Combination

Our hybrid architecture (fig. 2) is simple; Deutsch et al. (2020) explored a similar concept but did not achieve SOTA. But the benefits (section 4.1) from its simplicity are critical for RA, which has a lacking number/size of public datasets, wide educational use, and diverse handcrafted features. We obtain SOTA with a wider search on combinations.

4.2.1 Datasets and Evaluation Setups

WeeBit. Perhaps the most widely-used, WeeBit is often considered the gold standard in RA. It was first created as an expansion of the famous Weekly Reader corpus (Feng et al., 2009). To avoid classification bias, we downsample classes to equalize the number of items (passages) in each class to 625. It is common practice to downsample WeeBit.

Properties	WeeBit	OneStopEng	Cambridge
Target Audience	General	L2	L2
Covered Age	7∼16	Adult	A2∼C2 (CEFR)
Curriculum-Based?	No	No	Yes
Class-Balanced?	No	Yes	No
# of Classes	5	3	5
# of Items/Class	625	189	60
# of Tokens/Item	217	693	512
Accessibility	Author	Public	Public

Table 1: Statistics for datasets.

OneStopEnglish. OneStopEnglish is an aligned passage corpus developed for RA and simplification studies. A passage is paraphrased into three readability classes. OneStopEnglish is designed to be a balanced dataset. No downsampling is needed.

Cambridge. Cambridge (Xia et al., 2016) categorizes articles based on Cambridge English Exam levels (KET, PET, FCE, CAE, CPE). These five exams are targeted at learners at A2–C2 levels of the Common European Framework of Reference (Xia et al., 2016). We downsample to 60 items/class.

For evaluation, we calculate accuracy, weighted F1 score, precision, recall, and quadratic weighted kappa (QWK). The use of QWK is inspired by Chen et al. (2016); Palma et al. (2019). We use stratified k-fold (k=5, train=0.8, val=0.1, test=0.1) and average the results for reliability. We use SciKit-learn (Pedregosa et al., 2011) for metrics.

4.2.2 Search on Neural Model

Extending from the existing use of BERT on RA (Deutsch et al., 2020; Martinc et al., 2021), we explore RoBERTa, (Liu et al., 2019), BART (Lewis et al., 2020), and XLNet (Yang et al., 2019). We use base models for all (details in appendix D). For each of the four models (table 2), we perform grid searches on WeeBit validation sets to identify

Corpus		BERT	RoBERTa	BART	XLNet
WeeBit	Accuracy	0.893	**0.900**	0.889	0.881
	F1	0.893	**0.900**	0.889	0.880
	Precision	0.896	**0.902**	0.892	0.881
	Recall	0.896	**0.902**	0.892	0.881
	QWK	0.966	**0.970**	0.963	0.966
OneStopE	Accuracy	0.801	0.965	**0.968**	0.804
	F1	0.793	0.965	**0.968**	0.794
	Precision	0.815	0.968	**0.970**	0.810
	Recall	0.814	0.968	**0.970**	0.810
	QWK	0.840	0.942	**0.952**	0.845
Cambridge	Accuracy	0.573	**0.680**	0.620	0.573
	F1	0.517	**0.658**	0.598	0.554
	Precision	0.528	**0.693**	0.643	0.591
	Recall	0.525	**0.693**	0.643	0.591
	QWK	0.809	**0.881**	0.835	0.832

Table 2: Best performances, neural models.

Corpus		SVM	RandomF	XGBoost	LogR
WeeBit	Accuracy	**0.679**	0.638	0.638	0.622
	F1	**0.672**	0.626	0.627	0.615
	Precision	**0.696**	0.645	0.656	0.676
	Recall	**0.679**	0.638	0.638	0.622
	QWK	**0.716**	0.703	0.692	0.640
OneStopE	Accuracy	0.737	0.709	0.719	**0.778**
	F1	0.730	0.706	0.701	**0.770**
	Precision	0.751	0.726	0.734	**0.778**
	Recall	0.737	0.709	0.719	**0.778**
	QWK	0.400	0.434	0.363	**0.486**
Cambridge	Accuracy	0.627	0.673	**0.685**	0.680
	F1	0.613	0.663	**0.681**	0.657
	Precision	0.660	0.696	**0.701**	0.694
	Recall	0.627	0.673	**0.674**	0.680
	QWK	0.857	0.880	**0.852**	0.855

Table 3: Best performances, non-neural models.

Subgr	Model	
	LogR	SVM
EnDF	0.442	0.374
ShaF	0.404	0.409
TrSF	0.396	0.360
POSF	0.394	0.513
WorF	0.391	0.387
PsyF	0.378	0.437
WoKF	0.367	0.369

(a) *WeeBit*

Subgr	Model	
	LogR	SVM
TraF	0.513	0.620
PsyF	0.437	0.696
PhrF	0.429	0.608
VarF	0.409	0.626
TrSF	0.391	0.614
WorF	0.387	0.637
OSKF	0.359	0.605

(b) *OneStopEnglish*

Subgr	Model	
	LogR	SVM
TraF	0.640	0.593
WorF	0.613	0.593
ShaF	0.600	0.587
VarF	0.600	0.533
PsyF	0.593	0.620
POSF	0.553	0.407
WoKF	0.540	0.433

(c) *Cambridge*

Table 4: Top 7 Feature Subgroups.

the well-performing hyperparameters based on 5-fold mean accuracy. Once identified, we used the same configuration for all the other corpora and performed no corpus-specific tweaking. We search the learning rates of [1e-5, 2e-5, 4e-5, 1e-4] and the batch sizes of [8, 16, 32]. The input sequence lengths are all set at 512, and we used Adam optimizer. Last, we fine-tuned the model for three epochs. Full hyperparameters are in appendix F.

In table 2, RoBERTa & BART beat BERT & XLNet on most metrics. Martinc et al. (2021) reports that transformers are weak on parallel datasets (OneStopEnglish) due to their reliance on semantic information. However, RoBERTa & BART show great performances on OneStopEnglish as well. Such a phenomenon likely derives from numerous aspects of the architecture. We carefully posit that the varying pretraining steps could be a reason.

BERT uses two objectives, masked language model (MLM) and next sentence prediction (NSP). The latter was included to capture the relation between sentences for natural language inference. Thus, sentence/segment-level input is used. Likewise, XLNet adopts a similar idea, limiting input to sentence/segment-level. But RoBERTa disproved the efficiency of NSP, adopting document-level inputs. Similarly, BART, via random shuffling of sentences and in-filling scheme, does not limit itself to a sentence/segment size input. As in section 3, "readability" is possibly a global-level representation (accumulated across the whole document). Thus, the performance differences could stem from the pretraining input size; sentence/segment-level input likely loses the global-level information.

4.2.3 Search on Non-Neural Model

We explored SVM, Random Forest (RandomF), Gradient Boosting (XGBoost) (Chen and Guestrin, 2016), and Logistic Regression (LogR). With the exception of XGBoost, the chosen models are frequently used in RA but rarely go through adequate hyperparameter optimization steps (Ma et al., 2012; Yaneva et al., 2017; Mohammadi and Khasteh, 2020). We perform a randomized search to first identify the sensible range of hyperparameters to search. Then, we apply grid search to specify the optimal values. The parameters are in appendix F.

In table 3, we report the performances of the parameter-optimized models trained with all 255 handcrafted features. Compared to transformers, these non-neural models show lower accuracy in general. Even on the smallest Cambridge dataset, non-neural models do not necessarily show higher performances than transformers. But it is important to note that they managed to show fairly good, "expectable" performances on a much smaller dataset.

4.2.4 Search on Handcrafted Features

We start by ranking performances of the feature subgroups. In table 4, we report the top 7 (upper half) by accuracy on WeeBit. The result is obtained

Corpus		Model															
		Baselines, Previous Studies				BERT			RoBERTa			BART			XLNet		
		Xia-16	Fili-19	Mar-21		hybrid	Δ	Δ	hybrid	Δ	Δ	hybrid	Δ	Δ	hybrid	Δ	Δ
		SVM	LSTM	BERT	HAN	GB-T1	BERT	GB	RF-T1	RBRT	RF	RF-T1	BART	RF	RF-P3	XLNet	RF
WeeBit	Accuracy	0.803	0.813	**0.857**	0.752	0.895	0.002	0.257	0.902	0.002	0.264	**0.905**	0.016	0.267	0.892	0.011	0.254
	F1	-	-	**0.866**	0.753	0.895	0.002	0.268	0.902	0.002	0.276	**0.905**	0.016	0.279	0.892	0.012	0.266
	Precision	-	-	**0.857**	0.752	0.897	0.001	0.241	0.903	0.001	0.258	**0.905**	0.013	0.260	0.893	0.012	0.248
	Recall	-	-	**0.858**	0.752	0.897	0.001	0.259	0.903	0.001	0.265	**0.904**	0.012	0.266	0.892	0.011	0.254
	QWK	-	-	**0.953**	0.886	0.969	0.001	0.277	0.971	0.001	0.268	**0.968**	0.005	0.265	0.966	0.000	0.263
OneStopE	Accuracy	-	-	0.674	**0.787**	0.982	0.181	0.263	**0.990**	0.025	0.281	0.971	0.003	0.262	0.848	0.044	0.139
	F1	-	-	0.740	**0.798**	0.982	0.189	0.281	**0.995**	0.030	0.289	0.971	0.003	0.265	0.848	0.050	0.142
	Precision	-	-	0.674	**0.787**	0.983	0.168	0.249	**0.995**	0.027	0.269	0.972	0.002	0.246	0.852	0.042	0.126
	Recall	-	-	0.677	**0.789**	0.982	0.168	0.263	**0.996**	0.028	0.287	0.971	0.001	0.262	0.848	0.038	0.139
	QWK	-	-	0.708	**0.825**	0.973	0.133	0.610	**0.996**	0.054	0.562	0.952	0.000	0.518	0.855	0.010	0.369
Cambridge	Accuracy	0.786**	-	-	-	0.687	0.114	0.002	**0.763**	0.083	0.090	0.727	0.107	0.054	0.687	0.114	0.014
	F1	-	-	-	-	0.682	0.165	0.001	**0.752**	0.094	0.089	0.727	0.129	0.064	0.676	0.122	0.013
	Precision	-	-	-	-	0.732	0.204	0.031	**0.792**	0.099	0.096	0.760	0.117	0.064	0.710	0.119	0.014
	Recall	-	-	-	-	0.687	0.162	0.013	**0.753**	0.060	0.080	0.727	0.084	0.054	0.687	0.096	0.014
	QWK	-	-	-	-	0.873	0.064	0.021	**0.919**	0.038	0.039	0.889	0.054	0.009	0.888	0.056	0.008

** *Xia-16 (Cambridge) uses semi-supervised learning (self-training) on a larger corpus to increase performance.*

Table 5: Best performances, hybrid models.

Set	Features	LogR
T1	AdSem+Disco+Synta+LxSem+ShaTr	0.622
P3	ShaTr+EnDF+TrSF+POSF+WorF+PsyF+TraF+VarF	0.647

* *Note: 5 letters (e.g. AdSem) mean linguistic branch. 4 letters (e.g. PhrF) mean subgroup. We report accuracy on WeeBit.*

Table 6: Best feature sets.

after training the respective model using the specified feature subgroup. Importantly, the advanced semantic features show good performance in all measures. WorF and PsyF, features calculated from external databases, rank in the top 7 for all corpora, hinting they are strong measures of readability.

Moving on, we constructed several types of feature combinations with varying aims. These include: 1. **T-type** to thoroughly capture linguistic properties and 2. **P-type** to collect features by performance. We tested the variations on LogR and SVM to determine the optimal. Two sets (table 6) performed well. Appendix G reports all tested variations. We highlight that both advanced semantics and discourse added distinct (orthogonal) information, which was evidenced by performance change.

4.3 Assembling Hybrid Model

Based on the exploration so far, we assemble our hybrid model. We perform a brute-force grid search on four neural models (table 2), four non-neural models (table 3), and 14 feature sets (table 24).

To interweave the model, we followed the steps of 1: obtain soft labels (probabilities that a text belongs to the respective readability class) from a neural model by softmax layer, 2: append the soft labels to handcrafted features (create a dataframe), 3. train non-neural model on the dataframe. As in fig 2, the neural models performed a sort of re-prediction to the data used for training to match the dataframe dimensions in training and test stages.

Table 5 reports the best performing combination per respective neural model. Under "hybrid" column are code names (e.g. GB-T1 under BERT = XGBoost trained with handcrafted feature set T1 and BERT outputs). Under "Δ" column, we report differences with the respective single model performance. We also include SOTA baseline results Xia-16 → Xia et al. (2016), Fili-19 → Filighera et al. (2019), Mar-21 → Martinc et al. (2021).

4.4 Hybrid Model Results

In table 5, our hybrid models achieve SOTA performances on WeeBit (BART-RF-T1) and OneStopEnglish (RoBERTa-RF-T1). With the exception of Xia et al. (2016) which uses extra data to increase accuracy, we also achieve SOTA on Cambridge: 76.3% accuracy on a small dataset of only 60 items/class. Among the hybrids, RoBERTa-RF-T1 showed consistently high performance on all metrics. But all hybrid models beat previous SOTA results by a large margin. Notably, we achieve the near-perfect accuracy of 99% on OneStopEnglish, a massive 20.3% increase from the previous SOTA (Martinc et al., 2021) by HAN (Meng et al., 2020).

Both neural and non-neural models benefit from the hybrid architecture. This is explicitly shown in BERT-GB-T1 performance on OneStopEnglish, achieving 98.2% accuracy. This is an 18.1% increase from BERT and a 26.3% increase from XGBoost. However, BART did not benefit much from the hybrid architecture on WeeBit and OneStopEnglish, meaning that hybrid architectures do not augment model performance at all conditions.

Along similar lines, the hybrid architecture performance on the larger WeeBit dataset showed only a small improvement from the transformer-only result. On the other hand, the hybrid architecture performance on the smaller Cambridge dataset was consistently better than the transformer-only performance. The hybrid shows ∼10% improvement in accuracy on average for Cambridge. On the smallest dataset (Cambridge), the hybrid architecture benefited more from a non-neural, handcrafted features-based model like RF (Random Forest) and GB (XGBoost). On the largest dataset (WeeBit), the hybrid benefited more from a transformer.

Our explanation is that the handcrafted features do not add much, at the data size of WeeBit. But the handcrafted features could be a great help where data is insufficient like they did for the Cambridge dataset. OneStopEnglish, being the medium-sized parallel dataset, could have hit the sweet spot for the hybrid architecture. But it must be noted that the data size is not the only determining factor as to which model (neural or non-neural) the hybrid architecture benefits more from. It must also be questioned if the max performance (∵ label noise induced by subjectivity) (Frénay et al., 2014) is already achieved on WeeBit (Deutsch et al., 2020).

Also, it seems that the hybrid architecture benefits when each model (neural and non-neural) already shows considerably good performance. This is plausible as the neural model outputs are considered features for the non-neural model. Including more "fairly" well-performing features only creates extra distractions. The hybrid architecture's limit is that it gets a model from "good" to "great," not "fair" to "good." But determining the definition of "fair" performance is a difficult feat as it likely depends on the dataset and a researcher's intuition from the empirical experience of the model. Hence, the hybrid architecture's limit is that one must test several combinations to pick the effective one.

4.5 Why Not Directly Append Features?

Regarding the model architecture, we examined appending the handcrafted features to transformer embeddings without the use of a secondary predictor like SVM. But an existing example of ReadNet (Meng et al., 2020) hints that such a model is not robust to small datasets. ReadNet reports 52.8% accuracy on Cambridge, worse than *any* of our tested models (table 2, 3, 5). Besides, ReadNet claims to have achieved 91.7% accuracy on WeeBit, without reports on downsampling. Many studies, like Deutsch et al. (2020), report that the model accuracy can increase ∼4% on the full, class-imbalanced WeeBit. Hence, ReadNet is not directly comparable. We omitted ReadNet from table 5.

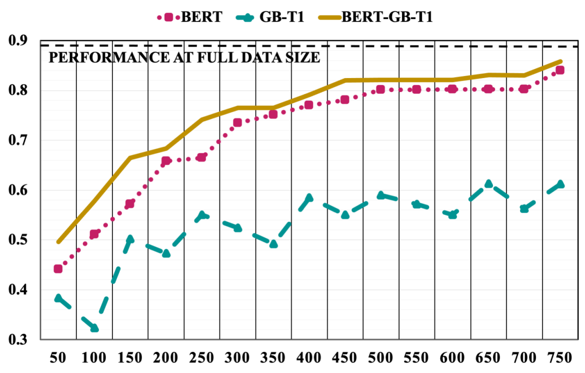

Figure 3: Performance Change, WeeBit Data Size

4.6 Why Was Our BERT Better?

Noticeable in table 2 and table 5 is that our BERT implementation performed much better on WeeBit than what was reported. The dataset preparation methods and overall evaluation settings are the same or very similar across ours (accuracy: 89.3%), Deutsch et al. (2020)'s (accuracy: 83.9%), and Martinc et al. (2021)'s (accuracy: 85.7%). We believe that the differences stem from the hyperparameters.

Notably, Deutsch et al. (2020) uses 128 input sequence length. This is ineffective as the downsampled WeeBit has 2374 articles of over 128 tokens but only 275 articles of over 512 tokens (which was our input sequence length). Hence, we can reasonably think that much semantic information was lost in Deutsch et al. (2020)'s implementation. Martinc et al. (2021) uses 512 input sequence length but lacks a report on other possibly critical hyperparameters, and we cannot compare in detail.

4.7 Data Size Effect

In table 5, our hybrid architecture generally did not contribute much to the classification on WeeBit. But we argue that it has much to do with data size.

To model how data size affects the accuracies of 1. hybrid model, 2. transformer, and 3. traditional ML, we conducted an additional experiment using the same test data (10% of WeeBit) explained in section 4.2.1. However, we random sampled the train data (80% of WeeBit) into the smaller sizes of from 50 to 750, with 50 passages increase each set. We sampled with equal class weights, meaning that a 250 passages train set has 50 from each readability class. We trained BERT-GB-T1 (table 5) on the sampled data and evaluated on the same test data throughout. We also recorded BERT and XGBoost (with T1 features) performances in fig. 3.

In fig. 3, the hybrid model performs consistently better than transformer (+0.01 $\sim$ 0.05) at all sizes. But the difference gap gets smaller as the train data size increases. The hybrid model does help the efficiency of learning RA linguistic properties.

Contrary to the conventional beliefs, the transformer (BERT) performed better than our expectations, even on smaller data sizes. BERT always outperformed XGBoost. The traditional ML performance was arguably more consistent but never better than a transfomer's.

BERT-GB-T1's trend line seemed like the mixture of GB-T1's and BERT's. Notably, BERT-GB-T1 achieves 85.8% accuracy on WeeBit using only 750 passages, 30% of the original train data. For comparison, 85.7% was the past SOTA (table 5).

5 Cross Domain Evaluation

99% accuracy on OneStopEnglish (table 5) shows that our model is capable of almost perfectly capturing the linguistic properties relating to readability on certain datasets. This is a positive and abnormally quick improvement, considering that the reported RA accuracies have never exceeded 90% on popular datasets (Vajjala and Meurers, 2012; Xu et al., 2015; Xia et al., 2016; Vajjala and Lučić, 2018) until 2021. Since the reported in-domain accuracies in RA had much room for improvement, we were not at the stage to be seriously concerned about cross-domain evaluation (Štajner and Nisioi, 2018) in this paper.

It would be very favorable to run an extra cross-domain evaluation (which we believe to be a next-level topic). But realistically, performing a cross-domain evaluation requires a thorough study on at least two datasets, which is potentially out of scope in this research. The readability classes/levels are labeled by a few human experts, making the standards vary among datasets. To make two datasets suitable for cross-domain evaluation, much effort is needed to connect the two, such as the class mapping used in Xia et al. (2016). However, it should be noted for future researchers that the notion of domain overfitting is indeed a common problem faced in RA, which often uses one dataset for train/test/validation. Without a new methodology to connect several datasets or a new large public dataset for RA, it will forever be challenging to develop a RA model for general use (Vajjala, 2021).

6 Conclusion

We have reported the four contributions mentioned in section 1. We checked that the new advanced semantic features add orthogonal information to the model. Further, we created hybrid models (table 5) that achieved SOTA results. RoBERTA-RF-T1 achieved 99% accuracy on OneStopEnglish, and BERT-GB-T1 beat the previous SOTA on WeeBit using only 30% of the original train data.

7 Acknowledgements

We wish to thank Dr. Inhwan Lee, Dongjun Lee, Sangjo Park, Donghyun Lee, and Eunsoo Shin. Partly funded by the 4th Industrial Revolution R&D Program, Min. of SMEs & Startups, Rep. of Korea.

References

Sandra Aluisio, Lucia Specia, Caroline Gasperin, and Carolina Scarton. 2010. Readability assessment for text simplification. In *Proceedings of the NAACL HLT 2010 Fifth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 1–9.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022.

OV Blinova, Tarasov NA, Modina VV, and IS Blekanov. 2020. Modeling lemma frequency bands for lexical complexity assessment of russian texts1. In *Computational Linguistics and Intellectual Technologies: Proceedings of the International Conference "Dialogue 2020", Moscow, June 17–20, 2020*, pages 76–92.

Leo Breiman. 2001. Random forests. *Machine learning*, 45(1):5–32.

Marc Brysbaert and Boris New. 2009. Moving beyond kučera and francis: A critical evaluation of current word frequency norms and the introduction of a new and improved word frequency measure for american english. *Behavior research methods*, 41(4):977–990.

Lori Buchanan, Chris Westbury, and Curt Burgess. 2001. Characterizing semantic space: Neighborhood effects in word recognition. *Psychonomic Bulletin & Review*, 8(3):531–544.

Chih-Chung Chang and Chih-Jen Lin. 2011. Libsvm: a library for support vector machines. *ACM transactions on intelligent systems and technology (TIST)*, 2(3):1–27.

Jing Chen, James H Fife, Isaac I Bejar, and André A Rupp. 2016. Building e-rater® scoring models using machine learning methods. *ETS Research Report Series*, 2016(1):1–12.

Tianqi Chen and Carlos Guestrin. 2016. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794.

Alina Maria Ciobanu, Liviu P Dinu, and Flaviu Pepelea. 2015. Readability assessment of translated texts. In *Proceedings of the International Conference Recent Advances in Natural Language Processing*, pages 97–103.

Kevyn Collins-Thompson. 2014. Computational assessment of text readability: A survey of current and future research. *ITL-International Journal of Applied Linguistics*, 165(2):97–135.

Kevyn Collins-Thompson and James P Callan. 2004. A language modeling approach to predicting reading difficulty. In *Proceedings of the human language technology conference of the North American chapter of the association for computational linguistics: HLT-NAACL 2004*, pages 193–200.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(76):2493–2537.

Scott A Crossley, Jerry Greenfield, and Danielle S McNamara. 2008. Assessing text readability using cognitively based indices. *Tesol Quarterly*, 42(3):475–493.

Patcharanut Daowadung and Yaw-Huei Chen. 2011. Using word segmentation and svm to assess readability of thai text for primary school students. In *2011 Eighth International Joint Conference on Computer Science and Software Engineering (JCSSE)*, pages 170–174. IEEE.

Aaron Defazio, Francis Bach, and Simon Lacoste-Julien. 2014. Saga: A fast incremental gradient method with support for non-strongly convex composite objectives. *arXiv preprint arXiv:1407.0202*.

Tovly Deutsch, Masoud Jasbi, and Stuart Shieber. 2020. Linguistic features for readability assessment. In *Proceedings of the Fifteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 1–17, Seattle, WA, USA Online. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 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)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Adam EM Eltorai, Syed S Naqvi, Soha Ghanian, Craig P Eberson, Arnold-Peter C Weiss, Christopher T Born, and Alan H Daniels. 2015. Readability of invasive procedure consent forms. *Clinical and translational science*, 8(6):830–833.

Vyvyan Evans. 2006. *Cognitive linguistics*. Edinburgh University Press.

Johan Falkenjack, Katarina Heimann Mühlenbock, and Arne Jönsson. 2013. Features indicating readability in swedish text. In *Proceedings of the 19th Nordic Conference of Computational Linguistics (NODALIDA 2013)*, pages 27–40.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. Liblinear: A library for large linear classification. *the Journal of machine Learning research*, 9:1871–1874.

Lijun Feng, Noémie Elhadad, and Matt Huenerfauth. 2009. Cognitively motivated features for readability assessment. In *Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009)*, pages 229–237.

Lijun Feng, Martin Jansche, Matt Huenerfauth, and Noémie Elhadad. 2010. A comparison of features for automatic readability assessment.

Anna Filighera, Tim Steuer, and Christoph Rensing. 2019. Automatic text difficulty estimation using embeddings and neural networks. In *European Conference on Technology Enhanced Learning*, pages 335–348. Springer.

Paul R Fitzsimmons, BD Michael, Joane L Hulley, and G Orville Scott. 2010. A readability assessment of online parkinson's disease information. *The journal of the Royal College of Physicians of Edinburgh*, 40(4):292–296.

Thomas François. 2014. An analysis of a french as a foreign language corpus for readability assessment. In *Proceedings of the third workshop on NLP for computer-assisted language learning*, pages 13–32.

Benoît Frénay, Ata Kabán, et al. 2014. A comprehensive introduction to label noise. In *ESANN*. Citeseer.

Edward Gibson. 1998. Linguistic complexity: Locality of syntactic dependencies. *Cognition*, 68(1):1–76.

Camille Guinaudeau and Michael Strube. 2013. Graph-based local coherence modeling. In *Proceedings of the 51st Annual Meeting of the Association for*

Computational Linguistics (Volume 1: Long Papers), pages 93–103.

John Hale. 2016. Information-theoretical complexity metrics. *Language and Linguistics Compass*, 10(9):397–412.

Hieronymus Hansen, Adam Widera, Johannes Ponge, and Bernd Hellingrath. 2021. Machine learning for readability assessment and text simplification in crisis communication: A systematic review. In *Proceedings of the 54th Hawaii International Conference on System Sciences*, page 2265.

Marti A. Hearst. 1998. Support vector machines. *IEEE Intelligent Systems*, 13(4):18–28.

Michael Heilman, Kevyn Collins-Thompson, Jamie Callan, and Maxine Eskenazi. 2007. Combining lexical and grammatical features to improve readability measures for first and second language texts. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 460–467, Rochester, New York. Association for Computational Linguistics.

Matthew Hoffman, Francis R Bach, and David M Blei. 2010. Online learning for latent dirichlet allocation. In *advances in neural information processing systems*, pages 856–864. Citeseer.

Thomas Holtgraves. 1999. Comprehending indirect replies: When and how are their conveyed meanings activated? *Journal of Memory and Language*, 41(4):519–540.

Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. 2020. spaCy: Industrial-strength Natural Language Processing in Python.

J. Kincaid, R. P. Fishburne, R. L. Rogers, and B. S. Chissom. 1975. Derivation of new readability formulas (automated readability index, fog count and flesch reading ease formula) for navy enlisted personnel.

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

Stephen Kokoska and Daniel Zwillinger. 2000. *CRC standard probability and statistics tables and formulae*. Crc Press.

Victor Kuperman, Hans Stadthagen-Gonzalez, and Marc Brysbaert. 2012. Age-of-acquisition ratings for 30,000 english words. *Behavior research methods*, 44(4):978–990.

Thomas K Landauer, Kirill Kireyev, and Charles Panaccione. 2011. Word maturity: A new metric for word knowledge. *Scientific Studies of Reading*, 15(1):92–108.

Bruce W. Lee and Jason Lee. 2020a. LXPER index 2.0: Improving text readability assessment model for L2 English students in Korea. In *Proceedings of the 6th Workshop on Natural Language Processing Techniques for Educational Applications*, pages 20–24, Suzhou, China. Association for Computational Linguistics.

Bruce W Lee and Jason Hyung-Jong Lee. 2020b. Lxper index: a curriculum-specific text readability assessment model for efl students in korea. *arXiv preprint arXiv:2008.01564*.

Gondy Leroy and David Kauchak. 2013. The effect of word familiarity on actual and perceived text difficulty. *Journal of the American Medical Informatics Association*, 21(e1):e169–e172.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Ping Li, Curt Burgess, and Kevin Lund. 2000. The acquisition of word meaning through global lexical co-occurrences. In *Proceedings of the thirtieth annual child language research forum*, pages 166–178. Citeseer.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Edward Loper and Steven Bird. 2002. Nltk: The natural language toolkit. *arXiv preprint cs/0205028*.

Xiaofei Lu. 2010. Automatic analysis of syntactic complexity in second language writing. *International journal of corpus linguistics*, 15(4):474–496.

Xiaofei Lu. 2011. A corpus-based evaluation of syntactic complexity measures as indices of college-level esl writers' language development. *TESOL quarterly*, 45(1):36–62.

Yi Ma, Eric Fosler-Lussier, and Robert Lofthus. 2012. Ranking-based readability assessment for early primary children's literature. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 548–552.

David Malvern and Brian Richards. 2012. Measures of lexical richness. *The encyclopedia of applied linguistics*.

Matej Martinc, Senja Pollak, and Marko Robnik-Šikonja. 2021. Supervised and Unsupervised Neural Approaches to Text Readability. *Computational Linguistics*, pages 1–39.

G Harry Mc Laughlin. 1969. Smog grading-a new readability formula. *Journal of reading*, 12(8):639–646.

Philip M McCarthy and Scott Jarvis. 2010. Mtld, vocd-d, and hd-d: A validation study of sophisticated approaches to lexical diversity assessment. *Behavior research methods*, 42(2):381–392.

Danielle S McNamara, Max M Louwerse, Philip M McCarthy, and Arthur C Graesser. 2010. Coh-metrix: Capturing linguistic features of cohesion. *Discourse Processes*, 47(4):292–330.

Changping Meng, Muhao Chen, Jie Mao, and Jennifer Neville. 2020. Readnet: A hierarchical transformer framework for web article readability analysis. In *Advances in Information Retrieval*, pages 33–49, Cham. Springer International Publishing.

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

Hamid Mohammadi and Seyed Hossein Khasteh. 2020. A machine learning approach to persian text readability assessment using a crowdsourced dataset. In *2020 28th Iranian Conference on Electrical Engineering (ICEE)*, pages 1–7. IEEE.

SungJin Nam, Gwen Frishkoff, and Kevyn Collins-Thompson. 2017. Predicting short-and long-term vocabulary learning via semantic features of partial word knowledge. *International Educational Data Mining Society*.

Diego Palma and John Atkinson. 2018. Coherence-based automatic essay assessment. *IEEE Intelligent Systems*, 33(5):26–36.

Diego Palma, Christian Soto, Mónica Veliz, Bernardo Riffo, and Antonio Gutiérrez. 2019. A data-driven methodology to assess text complexity based on syntactic and semantic measurements. In *International Conference on Human Interaction and Emerging Technologies*, pages 509–515. Springer.

Mary Anne Peabody and Charles E Schaefer. 2016. Towards semantic clarity in play therapy. *International Journal of Play Therapy*, 25(4):197.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830.

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

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

S. E. Petersen and Mari Ostendorf. 2009. A machine learning approach to reading level assessment. *Comput. Speech Lang.*, 23:89–106.

Penny M Pexman, Ian S Hargreaves, Paul D Siakaluk, Glen E Bodner, and Jamie Pope. 2008. There are many ways to be rich: Effects of three measures of semantic richness on visual word recognition. *Psychonomic Bulletin & Review*, 15(1):161–167.

Carla Pires, Afonso Cavaco, and Marina Vigário. 2017. Towards the definition of linguistic metrics for evaluating text readability. *Journal of Quantitative Linguistics*, 24(4):319–349.

Emily Pitler and Ani Nenkova. 2008. Revisiting readability: A unified framework for predicting text quality. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 186–195, Honolulu, Hawaii. Association for Computational Linguistics.

John C. Platt. 1999. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In *ADVANCES IN LARGE MARGIN CLASSIFIERS*, pages 61–74. MIT Press.

Rani Qumsiyeh and Yiu-Kai Ng. 2011. Readaid: a robust and fully-automated readability assessment tool. In *2011 IEEE 23rd International Conference on Tools with Artificial Intelligence*, pages 539–546. IEEE.

Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta. ELRA.

Mark Schmidt, Nicolas Le Roux, and Francis Bach. 2017. Minimizing finite sums with the stochastic average gradient. *Mathematical Programming*, 162(1-2):83–112.

Sarah E. Schwarm 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, page 523–530, USA. Association for Computational Linguistics.

Luo Si and Jamie Callan. 2001. A statistical model for scientific readability. In *Proceedings of the tenth international conference on Information and knowledge management*, pages 574–576.

Sanja Štajner and Sergiu Nisioi. 2018. A detailed evaluation of neural sequence-to-sequence models for in-domain and cross-domain text simplification. In *Proceedings of the eleventh international conference on language resources and evaluation (LREC 2018)*.

Kumiko Tanaka-Ishii, Satoshi Tezuka, and Hiroshi Terada. 2010. Sorting texts by readability. *Computational Linguistics*, 36(2):203–227.

Sara Tonelli, Ke M Tran, and Emanuele Pianta. 2012. Making readability indices readable. In *Proceedings of the First Workshop on Predicting and Improving Text Readability for target reader populations*, pages 40–48.

Sowmya Vajjala. 2021. Trends, limitations and open challenges in automatic readability assessment research. *arXiv preprint arXiv:2105.00973*.

Sowmya Vajjala and Ivana Lučić. 2018. Onestopenglish corpus: A new corpus for automatic readability assessment and text simplification. In *Proceedings of the thirteenth workshop on innovative use of NLP for building educational applications*, pages 297–304.

Sowmya Vajjala and Detmar Meurers. 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*, pages 163–173, Montréal, Canada. Association for Computational Linguistics.

Sowmya Vajjala and Detmar Meurers. 2013. On the applicability of readability models to web texts. In *Proceedings of the Second Workshop on Predicting and Improving Text Readability for Target Reader Populations*, pages 59–68.

Sowmya Vajjala and Detmar Meurers. 2016. Readability-based sentence ranking for evaluating text simplification. *arXiv preprint arXiv:1603.06009*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Menglin Xia, Ekaterina Kochmar, and Ted Briscoe. 2016. Text readability assessment for second language learners. In *Proceedings of the 11th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 12–22, San Diego, CA. Association for Computational Linguistics.

Wei Xu, Chris Callison-Burch, and Courtney Napoles. 2015. Problems in current text simplification research: New data can help. *Transactions of the Association for Computational Linguistics*, 3:283–297.

Victoria Yaneva, Constantin Orăsan, Richard Evans, and Omid Rohanian. 2017. Combining multiple corpora for readability assessment for people with cognitive disabilities. Association for Computational Linguistics.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in Neural Information Processing Systems*, 32:5753–5763.

Shuyu Zhang, Xuanyu Zhou, Huifeng Pan, and Junyi Jia. 2019. Cryptocurrency, confirmatory bias and news readability–evidence from the largest chinese cryptocurrency exchange. *Accounting & Finance*, 58(5):1445–1468.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc.

Yu Zhang, Houquan Zhou, and Zhenghua Li. 2020. Fast and accurate neural CRF constituency parsing. In *Proceedings of IJCAI*, pages 4046–4053.

Ciyou Zhu, Richard Byrd, Jorge Nocedal, and Jose Luis Morales. 2011. L-bfgs-b. *Retrieved Feb*, 18:2012.

Zhemin Zhu, Delphine Bernhard, and Iryna Gurevych. 2009. *A multi-dimensional model for assessing the quality of answers in social Q&A sites*. Ph.D. thesis.

A Trend, Advanced Semantic Features

Sorted Probability List	**R.**	**out**	**C.**	**out**	**N.**	**out**
9, 0.5, 0.5	Low	115	High	56.7	H-M	30.0
6, 2, 1, 0.5, 0.3, 0.2	L-M	177	H-M	43.3	High	48.1
4, 4, 1, 1	Mid	190	L-M	15.0	L-M	18.5
4, 2, 1, 1, 0.6, 0.4	H-M	204	Mid	25.0	Mid	35.3
2.5, 1.5, 1.5, 1.5, 1.5, 1.5	High	325	Low	8.34	Low	13.3

Table 7: Trends. Richness, Clarity, Noise. All numbers $\times 10$ for conciseness. L-M: Low-Mid. H-M: High-Mid.

In table 7, we name each list as $1 \sim 5$ from top to bottom. "out" refers to raw output from equations. See what the sorted probabilities list is in fig. 1.

Semantic Richness. List 4 and list 5 have the same lengths. However, list 5 contains more meaningful topics ($\uparrow p$) throughout the list, resulting in higher overall semantic richness. As such, semantic richness rewards long probability lists ($\uparrow n$) with more meaningful ($\uparrow p$) topics. Similarly, list 3 ($\downarrow n$,$\uparrow p$) has higher richness than list 2 ($\uparrow n$,$\downarrow p$).

Semantic Clarity. List 3 and list 4 have the same $max(p)$ and two other same elements (1). However, the second element in list 3 is the same as the first element, resulting in increased difficulty in identifying the main topic ($max(p)$). Likewise, semantic clarity rewards the deviation between the $max(p)$ and the other elements & short probability lists ($\downarrow n$). Hence, list 1 has the highest clarity.

Semantic Noise. List 2 and list 4 have the same lengths of 6 elements. However, list 2 contains more extraneous topics ($\downarrow p$), resulting in higher semantic noise. As such, semantic noise rewards longer lists ($\uparrow n$) with more extraneous elements ($\downarrow p$). As a result, list 5 has the least semantic noise.

B Features, Codes, and Definitions

idx	Code	Definition
1	WRich05_S	Richness, 50 topics extracted from Wikipedia Dump
2	WClar05_S	Clarity, 50 topics extracted from Wikipedia Dump
3	WNois05_S	Noise, 50 topics extracted from Wikipedia Dump
4	WTopc05_S	# of topics, 50 topics extracted from Wikipedia Dump
5	WRich10_S	Richness, 100 topics extracted from Wikipedia Dump
6	WClar10_S	Clarity, 100 topics extracted from Wikipedia Dump
7	WNois10_S	Noise, 100 topics extracted from Wikipedia Dump
8	WTopc10_S	# of topics, 100 topics extracted from Wikipedia Dump
9	WRich15_S	Richness, 150 topics extracted from Wikipedia Dump
10	WClar15_S	Clarity, 150 topics extracted from Wikipedia Dump
11	WNois15_S	Noise, 150 topics extracted from Wikipedia Dump
12	WTopc15_S	# of topics, 150 topics extracted from Wikipedia Dump
13	WRich20_S	Richness, 200 topics extracted from Wikipedia Dump
14	WClar20_S	Clarity, 200 topics extracted from Wikipedia Dump
15	WNois20_S	Noise, 200 topics extracted from Wikipedia Dump
16	WTopc20_S	# of topics, 200 topics extracted from Wikipedia Dump

Table 8: Wikipedia Knowledge Features (WoKF).

idx	Code	Definition
17	BRich05_S	Richness, 50 topics extracted from WeeBit Corpus
18	BClar05_S	Clarity, 50 topics extracted from WeeBit Corpus
19	BNois05_S	Noise, 50 topics extracted from WeeBit Corpus
20	BTopc05_S	# of topics, 50 topics extracted from WeeBit Corpus
21	BRich10_S	Richness, 100 topics extracted from WeeBit Corpus
22	BClar10_S	Clarity, 100 topics extracted from WeeBit Corpus
23	BNois10_S	Noise, 100 topics extracted from WeeBit Corpus
24	BTopc10_S	# of topics, 100 topics extracted from WeeBit Corpus
25	BRich15_S	Richness, 150 topics extracted from WeeBit Corpus
26	BClar15_S	Clarity, 150 topics extracted from WeeBit Corpus
27	BNois15_S	Noise, 150 topics extracted from WeeBit Corpus
28	BTopc15_S	# of topics, 150 topics extracted from WeeBit Corpus
29	BRich20_S	Richness, 200 topics extracted from WeeBit Corpus
30	BClar20_S	Clarity, 200 topics extracted from WeeBit Corpus
31	BNois20_S	Noise, 200 topics extracted from WeeBit Corpus
32	BTopc20_S	# of topics, 200 topics extracted from WeeBit Corpus

Table 9: WeeBit Knowledge Features (WBKF).

idx	Code	Definition
33	ORich05_S	Richness, 50 topics extracted from OneStop Corpus
34	OClar05_S	Clarity, 50 topics extracted from OneStop Corpus
35	ONois05_S	Noise, 50 topics extracted from OneStop Corpus
36	OTopc05_S	# of topics, 50 topics extracted from OneStop Corpus
37	ORich10_S	Richness, 100 topics extracted from OneStop Corpus
38	OClar10_S	Clarity, 100 topics extracted from OneStop Corpus
39	ONois10_S	Noise, 100 topics extracted from OneStop Corpus
40	OTopc10_S	# of topics, 100 topics extracted from OneStop Corpus
41	ORich15_S	Richness, 150 topics extracted from OneStop Corpus
42	OClar15_S	Clarity, 150 topics extracted from OneStop Corpus
43	ONois15_S	Noise, 150 topics extracted from OneStop Corpus
44	OTopc15_S	# of topics, 150 topics extracted from OneStop Corpus
45	ORich20_S	Richness, 200 topics extracted from OneStop Corpus
46	OClar20_S	Clarity, 200 topics extracted from OneStop Corpus
47	ONois20_S	Noise, 200 topics extracted from OneStop Corpus
48	OTopc20_S	# of topics, 200 topics extracted from OneStop Corpus

Table 10: OneStop Knowledge Features (OSKF).

idx	Code	Definition
49	to_EntiM_C	total number of Entities Mentions
50	as_EntiM_C	average number of Entities Mentions per sentence
51	at_EntiM_C	average number of Entities Mentions per token (word)
52	to_UEnti_C	total number of unique Entities
53	as_UEnti_C	average number of unique Entities per sentence
54	at_UEnti_C	average number of unique Entities per token (word)

Table 11: Entity Density Features (EnDF).

idx	Code	Definition
55	ra_SSToT_C	ratio of SS transitions : total, count from Entity Grid
56	ra_SOToT_C	ratio of SO transitions : total, count from Entity Grid
57	ra_SXToT_C	ratio of SX transitions : total, count from Entity Grid
58	ra_SNToT_C	ratio of SN transitions : total, count from Entity Grid
59	ra_OSToT_C	ratio of OS transitions : total, count from Entity Grid
60	ra_OOToT_C	ratio of OO transitions : total, count from Entity Grid
61	ra_OXToT_C	ratio of OX transitions : total, count from Entity Grid
62	ra_ONToT_C	ratio of ON transitions : total, count from Entity Grid
63	ra_XSToT_C	ratio of XS transitions : total, count from Entity Grid
64	ra_XOToT_C	ratio of XO transitions : total, count from Entity Grid
65	ra_XXToT_C	ratio of XX transitions : total, count from Entity Grid
66	ra_XNToT_C	ratio of XN transitions : total, count from Entity Grid
67	ra_NSToT_C	ratio of NS transitions : total, count from Entity Grid
68	ra_NOToT_C	ratio of NO transitions : total, count from Entity Grid
69	ra_NXToT_C	ratio of NX transitions : total, count from Entity Grid
70	ra_NNToT_C	ratio of NN transitions : total, count from Entity Grid

Table 12: Entity Grid Features (EnDF) Part 1.

idx	Code	Definition
71	LoCohPA_S	Local Coherence for PA score from Entity Grid
72	LoCohPW_S	Local Coherence for PW score from Entity Grid
73	LoCohPU_S	Local Coherence for PU score from Entity Grid
74	LoCoDPA_S	Local Coherence dist. for PA score from Entity Grid
75	LoCoDPW_S	Local Coherence dist. for PW score from Entity Grid
76	LoCoDPU_S	Local Coherence dist. for PU score from Entity Grid

Table 13: Entity Grid Features (EnDF) Part 2.

idx	Code	Definition
77	to_NoPhr_C	total count of Noun phrases
78	as_NoPhr_C	average count of Noun phrases per sentence
79	at_NoPhr_C	average count of Noun phrases per token
80	ra_NoVeP_C	ratio of Noun phrases : Verb phrases count
81	ra_NoSuP_C	ratio of Noun phrases : Subordinate clauses count
82	ra_NoPrP_C	ratio of Noun phrases : Prep phrases count
83	ra_NoAjP_C	ratio of Noun phrases : Adj phrases count
84	ra_NoAvP_C	ratio of Noun phrases : Adv phrases count
85	to_VePhr_C	total count of Verb phrases
86	as_VePhr_C	average count of Verb phrases per sentence
87	at_VePhr_C	average count of Verb phrases per token
88	ra_VeNoP_C	ratio of Verb phrases : Noun phrases count
89	ra_VeSuP_C	ratio of Verb phrases : Subordinate clauses count
90	ra_VePrP_C	ratio of Verb phrases : Prep phrases count
91	ra_VeAjP_C	ratio of Verb phrases : Adj phrases count
92	ra_VeAvP_C	ratio of Verb phrases : Adv phrases count
93	to_SuPhr_C	total count of Subordinate clauses
94	as_SuPhr_C	average count of Subordinate clauses per sentence
95	at_SuPhr_C	average count of Subordinate clauses per token
96	ra_SuNoP_C	ratio of Subordinate clauses : Noun phrases count
97	ra_SuVeP_C	ratio of Subordinate clauses : Verb phrases count
98	ra_SuPrP_C	ratio of Subordinate clauses : Prep phrases count
99	ra_SuAjP_C	ratio of Subordinate clauses : Adj phrases count
100	ra_SuAvP_C	ratio of Subordinate clauses : Adv phrases count
101	to_PrPhr_C	total count of prepositional phrases
102	as_PrPhr_C	average count of prepositional phrases per sentence
103	at_PrPhr_C	average count of prepositional phrases per token
104	ra_PrNoP_C	ratio of Prep phrases : Noun phrases count
105	ra_PrVeP_C	ratio of Prep phrases : Verb phrases count
106	ra_PrSuP_C	ratio of Prep phrases : Subordinate clauses count
107	ra_PrAjP_C	ratio of Prep phrases : Adj phrases count
108	ra_PrAvP_C	ratio of Prep phrases : Adv phrases count
109	to_AjPhr_C	total count of Adjective phrases
110	as_AjPhr_C	average count of Adjective phrases per sentence
111	at_AjPhr_C	average count of Adjective phrases per token
112	ra_AjNoP_C	ratio of Adj phrases : Noun phrases count
113	ra_AjVeP_C	ratio of Adj phrases : Verb phrases count
114	ra_AjSuP_C	ratio of Adj phrases : Subordinate clauses count
115	ra_AjPrP_C	ratio of Adj phrases : Prep phrases count
116	ra_AjAvP_C	ratio of Adj phrases : Adv phrases count
117	to_AvPhr_C	total count of Adverb phrases
118	as_AvPhr_C	average count of Adverb phrases per sentence
119	at_AvPhr_C	average count of Adverb phrases per token
120	ra_AvNoP_C	ratio of Adv phrases : Noun phrases count
121	ra_AvVeP_C	ratio of Adv phrases : Verb phrases count
122	ra_AvSuP_C	ratio of Adv phrases : Subordinate clauses count
123	ra_AvPrP_C	ratio of Adv phrases : Prep phrases count
124	ra_AvAjP_C	ratio of Adv phrases : Adj phrases count

Table 14: Phrasal Features (PhrF)

idx	Code	Definition
125	to_TreeH_C	total parsed Tree Height of all sentences
126	as_TreeH_C	average parsed Tree Height per sentence
127	at_TreeH_C	average parsed Tree Height per token
128	to_FTree_C	total length of Flattened parsed Trees
129	as_FTree_C	average length of Flattened parsed Trees per sentence
130	at_FTree_C	average length of Flattened parsed Trees per token

Table 15: Tree Structural Features (TrSF)

idx	Code	Definition
131	to_NoTag_C	total count of Noun tags
132	as_NoTag_C	average count of Noun tags per sentence
133	at_NoTag_C	average count of Noun tags per token
134	ra_NoAjT_C	ratio of Noun : Adjective count
135	ra_NoVeT_C	ratio of Noun : Verb count
136	ra_NoAvT_C	ratio of Noun : Adverb count
137	ra_NoSuT_C	ratio of Noun : Subordinating Conj. count
138	ra_NoCoT_C	ratio of Noun : Coordinating Conj. count
139	to_VeTag_C	total count of Verb tags
140	as_VeTag_C	average count of Verb tags per sentence
141	at_VeTag_C	average count of Verb tags per token
142	ra_VeAjT_C	ratio of Verb : Adjective count
143	ra_VeNoT_C	ratio of Verb : Noun count
144	ra_VeAvT_C	ratio of Verb : Adverb count
145	ra_VeSuT_C	ratio of Verb : Subordinating Conj. count
146	ra_VeCoT_C	ratio of Verb : Coordinating Conj. count
147	to_AjTag_C	total count of Adjective tags
148	as_AjTag_C	average count of Adjective tags per sentence
149	at_AjTag_C	average count of Adjective tags per token
150	ra_AjNoT_C	ratio of Adjective : Noun count
151	ra_AjVeT_C	ratio of Adjective : Verb count
152	ra_AjAvT_C	ratio of Adjective : Adverb count
153	ra_AjSuT_C	ratio of Adjective : Subordinating Conj. count
154	ra_AjCoT_C	ratio of Adjective : Coordinating Conj. count
155	to_AvTag_C	total count of Adverb tags
156	as_AvTag_C	average count of Adverb tags per sentence
157	at_AvTag_C	average count of Adverb tags per token
158	ra_AvAjT_C	ratio of Adverb : Adjective count
159	ra_AvNoT_C	ratio of Adverb : Noun count
160	ra_AvVeT_C	ratio of Adverb : Verb count
161	ra_AvSuT_C	ratio of Adverb : Subordinating Conj. count
162	ra_AvCoT_C	ratio of Adverb : Coordinating Conj. count
163	to_SuTag_C	total count of Subordinating Conj. tags
164	as_SuTag_C	average count of Subordinating Conj. per sentence
165	at_SuTag_C	average count of Subordinating Conj. per token
166	ra_SuAjT_C	ratio of Subordinating Conj. : Adjective count
167	ra_SuNoT_C	ratio of Subordinating Conj. : Noun count
168	ra_SuVeT_C	ratio of Subordinating Conj. : Verb count
169	ra_SuAvT_C	ratio of Subordinating Conj. : Adverb count
170	ra_SuCoT_C	ratio, Subordinating Conj. : Coordinating Conj. count
171	to_CoTag_C	total count of Coordinating Conj. tags
172	as_CoTag_C	average count of Coordinating Conj. per sentence
173	at_CoTag_C	average count of Coordinating Conj. per token
174	ra_CoAjT_C	ratio of Coordinating Conj. : Adjective count
175	ra_CoNoT_C	ratio of Coordinating Conj. : Noun count
176	ra_CoVeT_C	ratio of Coordinating Conj. : Verb count
177	ra_CoAvT_C	ratio of Coordinating Conj. : Adverb count
178	ra_CoSuT_C	ratio, Coordinating Conj. : Subordinating Conj. count
179	to_ContW_C	total count of Content words
180	as_ContW_C	average count of Content words per sentence
181	at_ContW_C	average count of Content words per token
182	to_FuncW_C	total count of Function words
183	as_FuncW_C	average count of Function words per sentence
184	at_FuncW_C	average count of Function words per token
185	ra_CoFuW_C	ratio of Content words to Function words

Table 16: Part-of-Speech Features (POSF)

idx	Code	Definition
186	SimpNoV_S	unique Nouns/total Nouns #Noun Variation
187	SquaNoV_S	(unique Nouns**2)/total Nouns #Squared Noun Variation
188	CorrNoV_S	unique Nouns/sqrt(2*total Nouns) #Corrected Noun Var
189	SimpVeV_S	unique Verbs/total Verbs #Verb Variation
190	SquaVeV_S	(unique Verbs**2)/total Verbs #Squared Verb Variation
191	CorrVeV_S	unique Verbs/sqrt(2*total Verbs) #Corrected Verb Var
192	SimpAjV_S	unique Adjectives/total Adjectives #Adjective Var
193	SquaAjV_S	(unique Adj**2)/total Adj #Squared Adj Variation
194	CorrAjV_S	unique Adj/sqrt(2*total Adj) #Corrected Adj Var
195	SimpAvV_S	unique Adverbs/total Adverbs #Adverb Variation
196	SquaAvV_S	(unique Adv**2)/total Adv #Squared Adv Variation
197	CorrAvV_S	unique Adv/sqrt(2*total Adv) #Corrected Adv Var

Table 17: Variation Ratio Features (VarF)

idx	Code	Definition
198	SimpTTR_S	unique tokens/total tokens #TTR
199	CorrTTR_S	unique/sqrt(2*total) #Corrected TTR
200	BiLoTTR_S	log(unique)/log(total) #Bi-Logarithmic TTR
201	UberTTR_S	$(\log(\text{unique}))^2/\log(\text{total}/\text{unique})$ #Uber
202	MTLDTTR_S	#Measure of Textual Lexical Diversity (TTR, 0.72)

Table 18: Type Token Ratio Features (TTRF)

idx	Code	Definition
203	to_AAKuW_C	total AoA (Age of Acquisition) of words, Kuperman
204	as_AAKuW_C	average AoA of words per sentence, Kuperman
205	at_AAKuW_C	average AoA of words per token, Kuperman
206	to_AAKuL_C	total AoA of lemmas, Kuperman
207	as_AAKuL_C	average AoA of lemmas per sentence, Kuperman
208	at_AAKuL_C	average AoA of lemmas per token, Kuperman
209	to_AABiL_C	total AoA of lemmas, Bird norm
210	as_AABiL_C	average AoA of lemmas, Bird norm per sent
211	at_AABiL_C	average AoA of lemmas, Bird norm per token
212	to_AABrL_C	total AoA of lemmas, Bristol norm
213	as_AABrL_C	average AoA of lemmas, Bristol norm per sent
214	at_AABrL_C	average AoA of lemmas, Bristol norm per token
215	to_AACoL_C	total AoA of lemmas, Cortese and Khanna norm
216	as_AACoL_C	average AoA of lem, Cortese and K norm per sent
217	at_AACoL_C	average AoA of lem, Cortese and K norm per token

Table 19: Psychollinguistic Features (PsyF)

idx	Code	Definition
218	to_SbFrQ_C	total SubtlexUS FREQcount value
219	as_SbFrQ_C	average SubtlexUS FREQcount value per sentence
220	at_SbFrQ_C	average SubtlexUS FREQcount value per token
221	to_SbCDC_C	total SubtlexUS CDcount value
222	as_SbCDC_C	average SubtlexUS CDcount value per sent
223	at_SbCDC_C	average SubtlexUS CDcount value per token
224	to_SbFrL_C	total SubtlexUS FREQlow value
225	as_SbFrL_C	average SubtlexUS FREQlow value per sent
226	at_SbFrL_C	average SubtlexUS FREQlow value per token
227	to_SbCDL_C	total SubtlexUS CDlow value
228	as_SbCDL_C	average SubtlexUS CDlow value per sent
229	at_SbCDL_C	average SubtlexUS CDlow value per token
230	to_SbSBW_C	total SubtlexUS SUBTLWF value
231	as_SbSBW_C	average SubtlexUS SUBTLWF value per sent
232	at_SbSBW_C	average SubtlexUS SUBTLWF value per token
233	to_SbL1W_C	total SubtlexUS Lg10WF value
234	as_SbL1W_C	average SubtlexUS Lg10WF value per sent
235	at_SbL1W_C	average SubtlexUS Lg10WF value per token
236	to_SbSBC_C	total SubtlexUS SUBTLCD value
237	as_SbSBC_C	average SubtlexUS SUBTLCD value per sent
238	at_SbSBC_C	average SubtlexUS SUBTLCD value per token
239	to_SbL1C_C	total SubtlexUS Lg10CD value
240	as_SbL1C_C	average SubtlexUS Lg10CD value per sent
241	at_SbL1C_C	average SubtlexUS Lg10CD value per token

Table 20: Word Familiarity Features (WorF)

idx	Code	Definition
242	TokSenM_S	total count of tokens x total count of sentence
243	TokSenS_S	sqrt(total count of tokens x total count of sentence)
244	TokSenL_S	log(total count of tokens)/log(total count of sent)
245	as_Token_C	average count of tokens per sentence
246	as_Sylla_C	average count of syllables per sentence
247	at_Sylla_C	average count of syllables per token
248	as_Chara_C	average count of characters per sentence
249	at_Chara_C	average count of characters per token

Table 21: Shallow Features (ShaF)

idx	Code	Definition
250	SmogInd_S	Smog Index
251	ColeLia_S	Coleman Liau Readability Score
252	Gunning_S	Gunning Fog Count Score (New, US Navy Report)
253	AutoRea_S	Automated Readability Idx (New, US Navy Report)
254	FleschG_S	Flesch Kincaid Grade Level (New, US Navy Report)
255	LinseaW_S	Linsear Write Formula Score

Table 22: Shallow Features (ShaF)

C Rules Behind Feature Codes

In table 8∼22, "Code" columns show feature codes. The related linguistic features appear with quite a number of variations across academia, without a naming convention (Zhu et al., 2009; Fitzsimmons et al., 2010; Tanaka-Ishii et al., 2010; Daowadung and Chen, 2011; Vajjala and Meurers, 2013; Ciobanu et al., 2015; Zhang et al., 2019; Blinova et al., 2020; Lee and Lee, 2020b). For consistency, we set ourselves a few naming rules.

1. Feature codes consist of 8 letters/numerals, with 1 or 2 underscores depending on feature types.
2. All features classify into either count-based or score-based, following popular convention.
   - Count-based
     - define: final calculation uses simple counts (i.e. total, avg per sent, avg per token, ratio)
     - format: xx_$xxxxx$_C. First two letters are "to" (total), "as" (avg per sent), "at" (avg per token), "ra" (ratio). Five letters in the middle explain what the feature is. Last letter always "C." Two underscores in between.
   - Score-based
     - define: require additional calculation (e.g. log, square), or famous features with predefined names (e.g. Flesch-Kincaid, TTR).
     - format: $xxxxxxx$_S. Seven letters are all dedicated to explaining what the feature is. Last letter always "S." One underscore.
3. For the "explanation" part of each feature code, capital letters denote new words. The small letters that follow are from the same word. (e.g. 1: Coleman Liau → ColeLia, 2: AoA (Age of Acquisition) Kuperman of words → AAKuW)

D Scientific Artifacts

We use Online LDA implemented by Gensim v4.0 (Řehůřek and Sojka, 2010). For most general tasks, including sentence/entity recognition, POS tagging, and dependency parsing, we use spaCy v3.0[8] (Honnibal et al., 2020) with en_core_web_sm pretrained model. For constituency parsing, we use CRF parser (Zhang et al., 2020) in SuPar v1.0 [9].

D.1 Transformers

For transformers, we use the following models from HuggingFace transformers v4.5.0 (Wolf et al., 2020).
1. **bert-base-uncased**
2. **roberta-base**
3. **bart-base**
4. **xlnet-base-cased**

D.2 Non-Neural Models

For non-neural models, we use the following models from from SciKit-Learn v0.24.1.
1. **support vector classifiers** (svm.SVC) (Hearst, 1998; Platt, 1999; Chang and Lin, 2011)
2. **random forest classifiers** (ensemble.RandomForestClassifier) (Breiman, 2001)
3. **logistic regression** (linear_model.LogisticRegression)

For gradient boosting, we use the following from XGBoost v1.4.0 (Chen and Guestrin, 2016).
4. **gradient boosting** (XGBclassifier)

E Preprocessing

Our preprocessing steps are inspired by Martinc et al. (2021) and several other existing RA research. These steps are used only during the extraction of handcrafted features for increased accuracy.
1. remove all special characters
2. remove words less than 3 characters
3. lowercase all
4. tokenize
5. remove NLTK default stopwords

F Full Hyperparameters

F.1 Machine Learning Models

We perform grid search on the hyperparameters (table 3) after performing a large randomized search to identify the sensible range of hyperparameters to tune. In particular, logistic regression solver hyperparameter search include libfgs (Zhu et al., 2011), liblinear (Fan et al., 2008), SAG (Schmidt et al., 2017), and SAGA (Defazio et al., 2014).

In table 3(a), C is the regularization parameter, G is the kernel coefficient (gamma), and K is the kernel. In table 3(b), nEst is the number of trees, MDep is the max depth of a tree, and Mfea is the number of features considered. In table 3(c), eta is the learning rate, G is the minimum loss reduction need to make a further partition on the leaf node (gamma), and MDep is the max depth of a tree. In table 4(d), C is the inverse of the regularization strength, Pen is the norm used in penalization, and Solver is the algorithm used in optimization. The other parameters best performed at default.

[8] github.com/explosion/spaCy
[9] github.com/yzhangcs/parser

Model	Hyperparameter		
	C	G	K
SVM	**1**	**scale**	rbf
	5	auto	linear
	10		**poly**
	50		sigmoid

(a) *SVM*, Best Params

Model	Hyperparameter		
	nEst	MDep	Mfea
RF	600	20	**auto**
	700	60	sqrt
	800	100	log2
	900	**None**	None

(b) *RandomF*, Best Params

Model	Hyperparameter		
	eta	G	MDep
XGBoost	1e-2	0	3
	5e-2	1e-2	6
	1e-1	1e-1	**9**
	2e-1	**1**	12

(c) *XGBoost*, Best Params

Model	Hyperparameter		
	C	Pen	Solver
LR	1e-1	**l1**	lbfgs
	5e-1	l2	l.linear
	1	elastic	newton
	10	none	**saga**

(d) *LogR*, Best Params

Table 23: Hyperparameters, non-neural models.

F.2 Transformers

We use AdamW (optimizer) (Kingma and Ba, 2014), linear (scheduler), 10% (warmup steps), 8 (batch size), 3 (epoch) for all tested transformers. We use the learning rate of 2e-5 for BERT and 3e-5 for the other three transformers.

G Feature Combinations

Set	Features	LogR	SVM
T1	**AdSem + Disco + Synta + LxSem + ShaTr**	**0.622**	**0.679**
T2	Disco + Synta + LxSem + ShaTr	0.528	0.546
T3	AdSem + Synta + LxSem + ShaTr	0.591	0.582
H1	**AdSem + Disco**	**0.463**	**0.513**
L1	Synta + LxSem	0.499	0.561
L2	**Set L1 - PhrF**	**0.539**	**0.577**
L3	Set L1 - VarF	0.529	0.551
L4	Set L1 - POSF	0.449	0.551
E1	AdSem + PsyF + WorF + TraF	0.489	0.473
E2	**AdSem + PsyF + WorF**	**0.490**	**0.479**
E3	PsyF + WorF	0.464	0.459
P1	EnDF + ShaF + TrSF + POSF + WorF + PsyF + TraF	0.608	0.633
P2	Set P1 + TraF	0.629	0.638
P3	**Set P2 + VarF**	**0.647**	**0.674**

[*] *Note: 5 letters (e.g. AdSem) mean linguistic branch. 4 letters (e.g. PhrF) mean subgroup. We report accuracy on WeeBit.*

Table 24: Defining feature sets.

The five types of feature sets have varying aims: 1. **T-type** thoroughly captures linguistic properties, 2. **H-type** captures the high-level properties, 3. **L-type** captures the low, surface-level properties, 4. **E-type** uses features calculated from external data (out-of-model info, i.e. Age-of-Acquisition), and 5. **P-type** collects features by performance. Both advanced semantic and discourse features add distinctive information. This can be evidenced by the performance decreases (T1 $\rightarrow$ T2 and T1 $\rightarrow$ T3). We checked that all measures of F1, precision, recall, and QWK followed the same trend. Similar method was used in Feng et al. (2009); Aluisio et al. (2010); Vajjala and Meurers (2012); Falkenjack et al. (2013); François (2014) to check if a feature added orthogonal information. More linguistic branches generally indicated better performance. We use SciKit-learn (Pedregosa et al., 2011) for metrics.

H Transformers Training Time

All numbers are in seconds. We report in the order of (BERT, RoBERTa, XLNet, BART). These are the average training times for each fold, with 80% of the full dataset used to train. We used an NVIDIA Tesla V100 GPU.
1. **WeeBit** (1546, 1485, 3617, 1202)
2. **OneStopEnglish** (451, 373, 977, 396)
3. **Cambridge** (215, 122, 393, 239)

I LingFeat

Throughout our paper, we mention LingFeat as one of our contributions to academia. This is because a large-scale handcrafted features extraction toolkit is scarce in RA, despite its reliance on the features.

LingFeat is a Python research package for various handcrafted linguistic features. More specifically, LingFeat is an NLP feature extraction software, which currently extracts 255 linguistic features from English string input. The package is available on both PyPI and GitHub.

Due to the wide number of supported features, we had to define subgroups (section 3) for features. Hence, features are not accessible individually. Instead, one has to call the subgroups to obtain the dictionary of the corresponding features. The corresponding code is applicable to LingFeat v.1.0.

```python
"""
Import

this is the only import you need
"""
from lingfeat import extractor


"""
Pass text

here, text must be in string type
"""
text = "..."
LingFeat = extractor.pass_text(text)


"""
Preprocess text

options (all boolean):
- short (def. False): include short words
- see_token (def. False): return token list
- see_sent_token (def. False): return sent

output:
- n_token
- n_sent
- token_list (optional)
- sent_token_list (optional)
"""
LingFeat.preprocess()
# or
# print(LingFeat.preprocess())


"""
Extract features

each method returns a dictionary of
the corresponding features
"""
# Advanced Semantic (AdSem) Features
WoKF=LingFeat.WoKF_() #Wiki Know. Features
WBKF=LingFeat.WBKF_() #WB Knowledge Features
OSKF=LingFeat.OSKF_() #OSE Knowledge Features

# Discourse (Disco) Features
EnDF=LingFeat.EnDF_() #Entity Dens. Features
EnGF=LingFeat.EnGF_() #Entity Grid Features

# Syntactic (Synta) Features
PhrF=LingFeat.PhrF_() #Phrasal Features
TrSF=LingFeat.TrSF_() #(Parse) Tree Features
POSF=LingFeat.POSF_() #POS Features

# Lexico Semantic (LxSem) Features
TTRF=LingFeat.TTRF_() #TTR Features
VarF=LingFeat.VarF_() #Variational Features
PsyF=LingFeat.PsyF_() #Psycholing Difficulty
WoLF=LingFeat.WorF_() #Word Familiarity

# Shallow Traditional (ShTra) Features
ShaF=LingFeat.ShaF_() #Shallow Features
TraF=LingFeat.TraF_() #Traditional Formulas
```