Spring 6-1-2021

Lexical Complexity Prediction with Assembly Models

Aadil Islam
Dartmouth College, aadil.islam.21@dartmouth.edu

LEXICAL COMPLEXITY PREDICTION WITH ASSEMBLY MODELS

A Thesis

Submitted to the Faculty

in partial fulfillment of the requirements for the

degree of

Bachelor of Arts

in

Computer Science

by

Aadil Islam

DARTMOUTH COLLEGE

Hanover, New Hampshire

June 1, 2021

Advised by

Professor Soroush Vosoughi

Department of Computer Science

Acknowledgments

This past year, I was fortunate to have been guided by graduate student Weicheng Ma and Professor Soroush Vosoughi. I owe everything to their invaluable guidance and encouragement that inspired me during these turbulent times, which has motivated me to pursue future academic research. Given the ongoing pandemic that has transformed how we live our lives on campus and off, I'm especially thankful to have been able to conduct my research fully remotely. This would not have been possible without Weicheng and Professor Vosoughi's open arms, in addition to all the educators at Dartmouth College's departments of Computer Science and Mathematics who have made my undergraduate education possible. I sincerely could not have dreamed for a better situation. Finally, I'd like to thank Dartmouth College's Research Information, Technology and Consulting (RITC) department for providing me the computational resources necessary to complete my work.

Preface

This paper serves as my senior thesis for the undergraduate major in Computer Science at Dartmouth College, as part of the requirements for the departmental Honors Program. It concludes two semesters of research from Fall 2020 to Winter 2021 in the class COSC 99 (Thesis Research). This paper differs from that published in proceedings of the 15th International Workshop on Semantic Evaluation (SemEval) (Islam et al., 2021), which was a system description summarizing my submission to the Lexical Complexity Prediction (LCP) shared task hosted at SemEval-2021. In contrast, this paper expands on data exploration that was conducted, describes our system's underlying feature set in finer detail, and critically analyzes model performances.

Abstract

Tuning the complexity of one's writing is essential to presenting ideas in a logical, intuitive manner to audiences. This paper describes a system submitted by team `BigGreen` to LCP 2021 for predicting the lexical complexity of English words in a given context. We assemble a feature engineering-based model and a deep neural network model with an underlying Transformer architecture based on BERT. While BERT itself performs competitively, our feature engineering-based model helps in extreme cases, eg. separating instances of easy and neutral difficulty. Our handcrafted features comprise a breadth of lexical, semantic, syntactic, and novel phonetic measures. Visualizations of BERT attention maps offer insight into potential features that Transformers models may implicitly learn when fine-tuned for the purposes of lexical complexity prediction. Our assembly technique performs reasonably well at predicting the complexities of single words, and we demonstrate how such techniques can be harnessed to perform well when on multi word expressions (MWEs) too.

Contents

Chapter 1

Introduction

Lexical simplification (LS) is the task of replacing difficult words in a text with simpler alternatives. It is relevant in reading comprehension, where early studies have shown infrequent words to lead to more time a reader spends fixated on it, and that ambiguity in a word's meaning further adds to comprehension time (Rayner and Duffy, 1986). Devlin (1999) demonstrate that a combined approach studying both the *syntactic* structure of a context (where certain compositions may be more difficult to understand than others) and the *lexical* characteristics of each word (certain words are more frequent in language than other) can be used to simplify challenging texts for individuals suffering from aphasia. Complex word identification (CWI) is believed to be a fundamental step in the automation of lexical simplification (Shardlow, 2014). Early techniques for conducting CWI, however, lack in robustness at the word level, from initially simplifying all words in given a sentence to then observe incurred change in meaning (Devlin, 1998), to applying simple thresholds on discriminative features like word frequency (Zeng et al., 2005).

The recent CWI shared task at SemEval-2016 (Paetzold and Specia, 2016a) studied the annotations of 400 non-native speakers on English target words, labeled as either simple or complex. The SemEval-2018 CWI shared task (Yimam et al., 2018) extended their study to data across four languages, while also introducing a probabilistic component to

the binary classification task. This year's Lexical Complexity Prediction (LCP) shared task (Shardlow et al., 2021) forgoes the treatment of word difficulty as a binary classification task (Paetzold and Specia, 2016a; Yimam et al., 2018) and instead measures degree of difficulty on a continuous scale. This choice is intriguing as it mitigates a dilemma with previous approaches that treat words close to a decision boundary (suppose a threshold decides whether a word is considered 'difficult') identically to those that are far away, ie. extremely easy or extremely difficult.

Teams are asked to submit predictions on unlabeled test sets for two subtasks: predicting on English single word and multi word expressions (MWEs). The Pearson correlation coefficient is used to evaluate how closely submitted predictions associate with ground truth labels. For each subtask, `BigGreen` presents a machine learning-based approach that fuses the predictions of a feature engineering-based regressor with those of a feature learning-based deep neural network model founded on BERT (Devlin et al., 2018). Our code is made fully available on GitHub.[1]

In Sections 2 and 3 of this paper, we overview related work that inspires our experiments and describe the data used to train `BigGreen`'s models. For the feature engineering-based model, Section 4 explains six categories of linguistic features experimented with, in addition to feature selection techniques; for the feature learning-based model, we describe its training procedure here instead. Sections 5 and 6 show how our approaches performed in competition, as well as analysis on worked and what did not.

[1] https://github.com/Aadil101/BigGreen-at-LCP-2021

Chapter 2

Related Work

Previous studies have looked at estimating the readability of a given text, though at the sentence-level rather than word-level. Namely, Mc Laughlin (1969) regresses the number of polysyllabic words in a given lesson against the mean score for students quizzed on information pertaining to the lesson, yielding the SMOG Readability Formula. Dale and Chall (1948) offer a list of 768 (later updated to 3,000) words familiar to grade-school students, such that passages containing members of this list tend to be reported as having lesser reading difficulty. Yet, an issue with traditional readability metrics seems to be a loss of generality at the word-level.

Shardlow (2013) tries a brute force approach where a simplification algorithm is applied to each word of a given text, deeming a word complex only if it is simplified. However, this suffers from the assumption that non-complex words do not require further simplification. The author also tries assigning a familiarity score to the target word, and then determining whether the word is complex or not through the use of a threshold. We avoid thresholding our features in this study as we find it unnecessary, since raw familiarity scores can be used as features in regression-based tasks.

Results for the SemEval-2016 task (Zampieri et al., 2017) suggest vote ensembling predictions of one's best performing models to be an effective strategy, while several

top-performing models appear to use linguistic information beyond just word frequency (Paetzold and Specia, 2016b; Ronzano et al., 2016; Mukherjee et al., 2016); such models harness a variety of lexical, semantic, syntactic, and psycholinguistic features. This inspires our use of ensemble techniques, as well us our consideration of phonetic features as a new area of research. Moreover, Zampieri et al. (2017) emphasize the effectiveness of Decision Trees and Random Forests across top performing models, perhaps due to the importance of having features across a variety of categories (eg. lexical, morphological, syntactic, etc.) as opposed to features exclusive to a certain category. Note that the success of tree-based approaches is quantified by the G-score (which measures the harmonic mean between Accuracy and Recall). Interestingly, tree-based approaches tend to be outperformed by simpler threshold-based strategies based on F1-score; namely, Wróbel (2016) learns a threshold over word frequencies that maximizes F1-score over the CWI training dataset. Finally, evident is the underperformance of neural network and/or word embedding-based models in competition, potentially explained by the limited size of the training data.

Results from the SemEval-2018 task (Yimam et al., 2018) show progress in CWI research with regards to cross-lingual prediction, demonstrated by successful modeling of lexical complexity by numerous systems on a per-language basis. Surprisingly, this does *not* require models necessarily be trained on data in the language of interest; training upon data available in one or more foreign languages is sufficient for predicting upon an 'unseen' language, perhaps suggesting the universality of certain predictors of lexical complexity. Feature engineering-based systems appear to outperform competing deep learning-based systems, despite the latter having generally better performances since the CWI shared task 2016.

Chapter 3

Data Collection

Section 3.1

CompLex Dataset

Shardlow et al. (2020) present CompLex, a novel dataset in which each target expression (a single word or two-token MWE) is assigned a continuous label denoting its lexical complexity. Each label falls in range 0-1, and represents the (normalized) average score given by employed crowd workers who report the given expression's difficulty on a 5-point Likert scale. Note that all crowd workers originate from English speaking countries (UK, USA, and Australia). We define a sample's *class* as the bin to which its complexity

Corpus	**Subtask**	**Train**	**Trial**	**Test**
Bible	Single Word	2574	143	283
	Multi Word	505	29	66
Biomed	Single Word	2576	135	289
	Multi Word	514	33	53
Europarl	Single Word	2512	143	345
	Multi Word	498	37	65
Total	Single Word	7662	421	917
	Multi Word	1517	99	184

Table 3.1: LCP train, trial, and test sets.

label belongs, where bins are created using the following mapping of complexity ranges: $[0, 0.2) \rightarrow 1$, $[0.2, 0.4) \rightarrow 2$, $[0.4, 0.6) \rightarrow 3$, $[0.6, 0.8) \rightarrow 4$, $[0.8, 1] \rightarrow 5$. Target expressions in CompLex have 0.395 average complexity and 0.115 standard deviation, reflecting an imbalance in favor of class 2 and 3 samples.

Each target expression is accompanied by the sentence (ie. context) it was extracted from, where certain target words appear multiple times in the dataset (across different contexts). A target expression is drawn from one of three sources (Bible, Biomed, and Europarl) in an effort to motivate study of domain-specific linguistic features. As noted by Shardlow et al. (2020), Biomed samples appear to be on average 1.7 and 2.2 percent higher in lexical complexity than Bible and Europarl samples, respectively. This we believe may reflect the elevated diction exhibited in biomedical research and generally in academic writing, a style that differs from typical colloquial dialogue used in biblical scripture and parliamentary debate. Although an annotator's predisposition to theology, academia, and/or politics may affect his/her labeling of samples, this aspect is sadly beyond the scope of this study. Nonetheless, a summary of train, trial,[1] and test set samples is given in Table 3.1.

Section 3.2

External Datasets

In this study, we use four additional corpora to extract a breadth of term frequency-based features for training our feature engineering-based model on:

- **English Gigaword Fifth Edition** (Gigaword): this comprises articles from seven prominent international English newswires, acquired by the Linguistic Data Consortium (LDC) (Parker et al., 2011). Newswires include the Agence France Press English Service (AFE), Associated Press Worldstream English Service (APW), The

[1] In our study we avoid the trial set as we find it to be less representative of the training data, opting instead for training set cross-validation (stratified by corpus and complexity label).

Newswire	Files	Articles	Total Words
AFE	44	656269	170969
APW	91	1477466	539665
NYT	96	1298498	914159
XIE	83	679007	131711
Total	314	4111240	1756504

Table 3.2: Breakdown of Gigaword articles by newswire. Note that for each newswire, articles are grouped together in certain files. **Total Words** denotes the total number of whitespace-separated tokens across all articles of a given newswire.

New York Times Newswire Service (NYT), and The Xinhua News Agency English Service (XIE). Table 3.2 provides a breakdown of each newswire for reference.

- **Google Books Ngrams, version 2** (GBND): this is used to count occurences of phrases across a corpus of books, accessed via the PhraseFinder API (Trenkmann). The full corpus spans about 8 million books (ie. about 6% of books ever published), and is constructed using improved OCR technology and metadata extraction compared to its prior version (Lin et al., 2012).

- **British National Corpus, version 3** (BNC): this is a 100 million word collection of British written and spoken English text (Consortium et al., 2007). Particularly intruiging is its consideration of spoken English text, for this may potentially account for intricacies of language in speech versus in writing.

- **SUBTLEXus**: this comprises American English movie subtitles totaling 51 million in words. The creators of SUBTLEXus (Brysbaert and New, 2009) offer multiple ready-made word frequency lists; one list considers the number of movies containing a given word, whereas another measures how many times the word occurs specifically in its lowercase form. Please see Appendix A.4.3 for further details.

Chapter 4

Data Exploration

Section 4.1

Assessing Baseline Models

We focus initially on the single word subtask by experimenting with a baseline model inspired by that of the authors of CompLex (Shardlow et al., 2020). Our goal with this baseline model is to closely examine successful predictors, and to understand the shortcomings of said feature set, guiding the construction of `BigGreen`'s own system. The original baseline model comprises a series of handcrafted features (HC), GloVe word embeddings (Pennington et al., 2014), GloVe context embeddings (kindly see Section 5.1.1 for more on

Corpus	All	Handcrafted	GloVe word+context	InferSent
All	0.1399	0.0964	0.0749	0.1585
Bible	0.1133	0.0986	0.0884	-
Biomed	-	0.1094	0.0852	0.2060
Europarl	-	0.0762	0.0698	-

Table 4.1: Performances of linear regression models fitted on different baseline feature subsets. The single word training set is split into smaller training and development subsets (stratified by corpus and class); all models are fitted and evaluated on the former and latter, respectively. All scores are reported in terms of mean absolute error (MAE). Note that **GloVe word+context** denotes the concatenation of 300-dim GloVe word and 300-dim GloVe context embeddings. A hyphen '-' denotes an extremely high MAE.

Corpus	Class	HC	Glove word+context	InferSent
Bible	1	0.1354	0.0931	-
	2	0.0504	0.0690	-
	3	0.1605	0.0681	-
	4	0.3219	0.2319	-
	5	0.5588	0.1693	-
Biomed	1	0.1467	0.0903	0.2341
	2	0.0586	0.0619	0.1812
	3	0.1473	0.0910	0.1995
	4	0.3004	0.1371	0.3437
	5	0.4070	0.2334	0.3614
Europarl	1	0.1098	0.0720	-
	2	0.0417	0.0523	-
	3	0.1564	0.0737	-
	4	0.3156	0.2200	-

Table 4.2: Performances (organized by class) of linear regression models fitted on different baseline feature subsets. Note that we did not try the **All** feature subset shown in Table 4.1 due to time constraints. All scores are reported in terms of mean absolute error (MAE). A hyphen '-' denotes an extremely high MAE.

Corpus	Class	HC	Glove word+context	InferSent
Bible	1	0.1377	0.0909	0.1449
	2	0.0549	0.0845	0.1146
	3	0.1528	0.0634	0.1942
	4	0.3126	0.1922	0.3348
	5	0.5630	0.1420	0.5505
Biomed	1	0.1513	0.0893	-
	2	0.0693	0.0718	-
	3	0.1275	0.0927	-
	4	0.2765	0.1243	-
	5	0.3721	0.2184	-
Europarl	1	0.1078	0.0662	0.1556
	2	0.0434	0.0610	0.1252
	3	0.1483	0.0759	0.1645
	4	0.3082	0.1880	0.2344

Table 4.3: Performances (organized by class) of linear regression models fitted on different baseline feature subsets. This is analogous to Table 4.2, but here models were fitted on a reduced training set containing only half the available class 2 samples.

this) and InferSent sentence embeddings (Conneau et al., 2017). The handcrafted features include (1) target word frequency, courtesy of the Wordfreq library (Speer et al., 2018), (2) word length, and (3) syllable count, via the Syllables library.[1]

As shown in Table 4.1, performances of variant baseline models (*variant* denoting a fitting over a subset of the aforementioned features) show promise in the use of all handcrafted lexical features, and successes with certain semantic features (eg. GloVe word embeddings, but not InferSent embeddings). InferSent embeddings appear to not generalize well for the task at hand, which may suggest a general need for fine-tuning upon whatever pre-trained models we end up extracting semantic representations of sentences from.

Performances of the variant baseline models can be further examined, namely at the class-level (ie. how well a given model predicts lexical complexity across specific classes of difficulty). Observe that in Table 4.2, it appears models based on HC and Glove word+context do best at predicting on class 2 samples, regardless of corpora. This is likely due to there being more class 1-3 samples available in the CompLex corpus to train on, causing models to perform better on lower-rating samples. To try and boost scores across higher-rating samples, we refit our variant baseline models, but this time training upon a reduced training set containing only half the available class 2 samples. As shown in Table 4.3, models based on HC and Glove word+context indeed perform slightly better on classes 3-5 than previously, across all corpora. Performance on class 2 samples is sacrificed, however. Nonetheless, we hope to reuse this strategy for boosting performance across class 4, 5 samples later.

Section 4.2

Examining Challenging Samples

We manually assess training set samples that multiple variant baseline models struggled with, samples that are perhaps naturally difficult to predict on; Table 4.4 shows some of these.

[1] https://github.com/prosegrinder/python-syllables

ID	Corpus	Context	Complexity
1	Bible	Now God made Daniel to find kindness and compassion in the sight of the prince of the **eunuchs**.	0.632
2	Bible	he sent Hadoram his son to king David, to Greet him, and to bless him, because he had fought against Hadadezer and struck him; (for Hadadezer had wars with **Tou**;) and he had with him all kinds of vessels of gold and silver and brass.	0.825
3	Biomed	During budding morphogenesis, intersecting signaling networks from the epithelium and mesenchyme govern transcriptional, adhesive, polarity, and **motility** programs in these select groups of cells.	0.714
4	Biomed	Future fine mapping experiments can be designed to randomize the influences of any contaminating donor **alleles** and environmental differences, as well as test for maternal genotype effects.	0.625
5	Biomed	In the development of the mammalian retina, a diverse **range** of cell types is generated from a pool of multipotent retinal progenitor cells.	0.050
6	Biomed	In the mouse model of **RA**, small genetic contributions are also often observed.	0.813
7	Biomed	As in our tet-off **APP** mice, SantaCruz et al. found that tau neurofibrillary tangles, like amyloid plaques, are not cleared efficiently following transgene suppression.	0.692
8	Biomed	These mice were mated with a strain carrying **Cre** recombinase under the control of the human Keratin 14 (K14) promoter, which is active in basal cells of epidermis and other stratified epithelia.	0.783
9	Europarl	Once there is a statute based on equality, Madam President, the rules for travel and **subsistence** expenses can also be changed.	0.486
10	Europarl	Mobilisation of the European **Globalisation** Adjustment Fund: Ireland - SR Technics (	0.611

Table 4.4: Ten of the top-50 samples predicted with large error margins (MAE) by multiple variant baseline models. Each sample's target word is bolded.

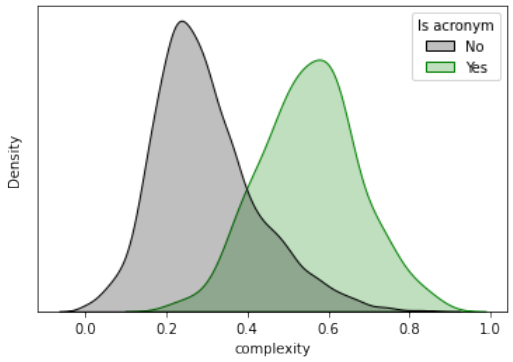


Figure 4.1: Probability density functions for acronym vs. non-acronym target words in single word training set.

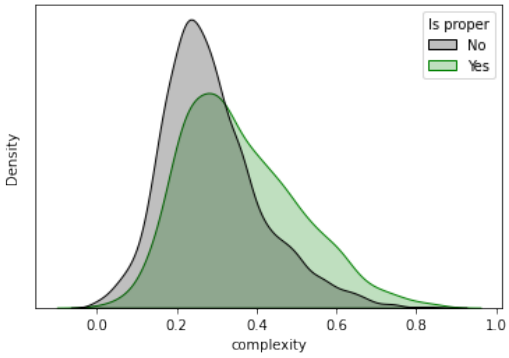


Figure 4.2: Probability density functions for proper vs. improper noun/adjective target words in single word training set.

Observe that samples 6 and 7 bear target words that are acronyms, terms like 'RA'and 'APP' whose meanings are perhaps assumed to be common knowledge to academic audiences. Samples 2, 8, and 10 have target words that are proper nouns and adjectives, usually referring to domain-specific entities that cannot necessarily be learned from context clues (eg. samples 2 and 8 seem to expect the reader to know what 'Tou' and 'Cre' are). Figures 4.1 and 4.2 illustrate the effects of acronymity and propriety to perceived target word complexity.

Finally, we notice that target words used rarely in language (eg. Samples 1, 3, 4, and 9) need to be better considered by future systems. Moreover, we hypothesize that N-grams comprising a target word affect its perceived complexity; the target word in sample 5 is often used in the phrase '**range** of,' whereas the target word in sample 4 rarely ever arises in a niche noun phrase like 'contaminating donor **alleles**.'

Section 4.3

Character Transition Probabilities

While prior top-performing approaches harness a variety of linguistic information across lexical, semantic, syntactic, and psycholinguistic features (Paetzold and Specia, 2016b; Ronzano et al., 2016; Mukherjee et al., 2016), there appears to be a lack of consideration of

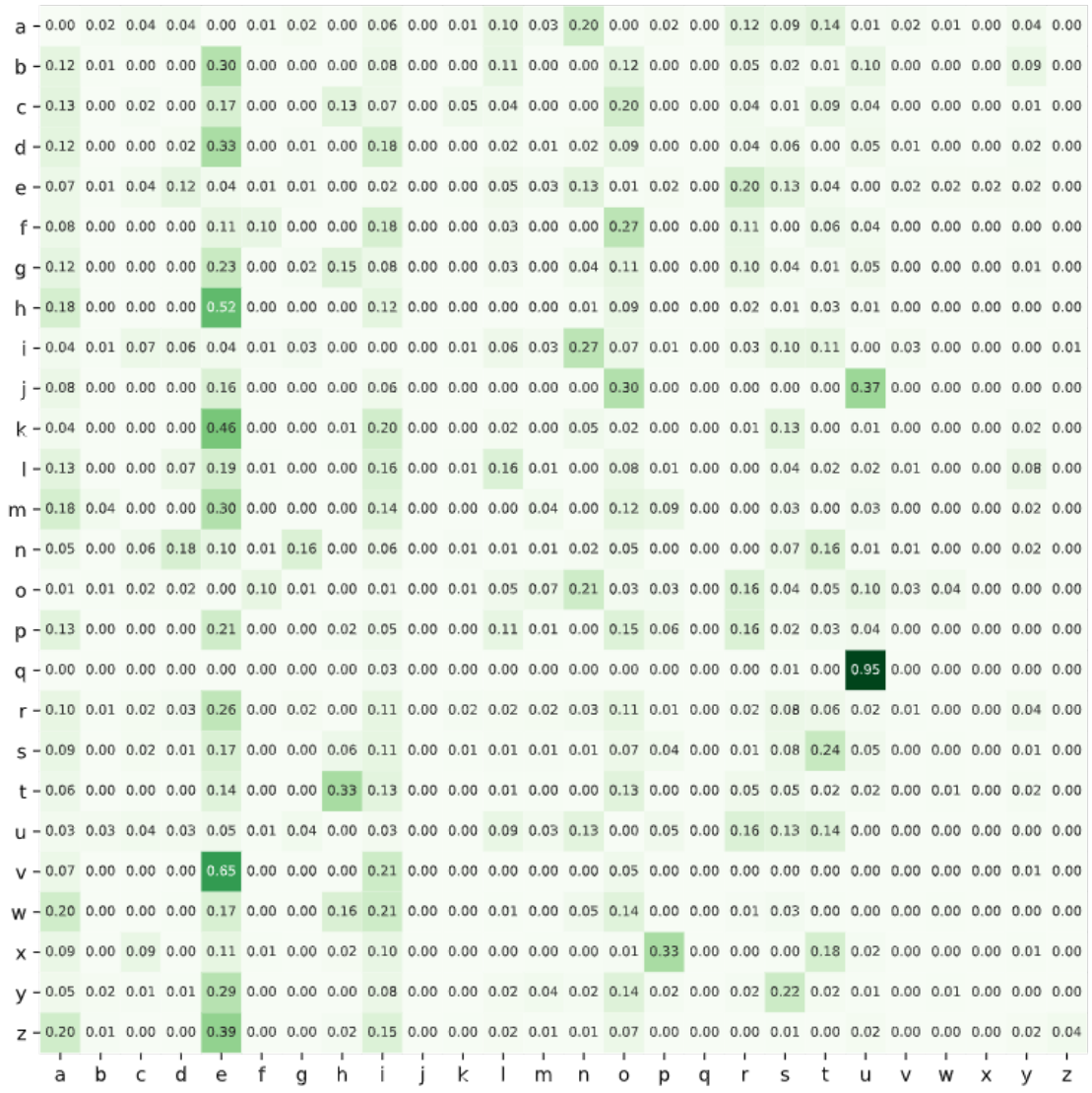


Figure 4.3: Lowercase ASCII transition probabilities, estimated over Gigaword.

Char. Transition	Character Transition Probability			T-Statistic	P-Value
	Gigaword Samples	Class 1 Samples	Class 4, 5 Samples		
$i \rightarrow d$	0.0597	0.0331	0.0588	-7.3532	4.5127e-10
$o \rightarrow s$	0.0402	0.0200	0.0976	-4.4796	2.7255e-05
$o \rightarrow r$	0.1568	0.1233	0.1463	-2.1475	1.6440e-02
$y \rightarrow c$	0.0129	0.0500	0.3333	-1.8898	5.8693e-02
$p \rightarrow h$	0.0206	0.0156	0.4500	-1.5407	6.8241e-02
$o \rightarrow n$	0.2079	0.2700	0.1707	-1.4155	7.8776e-02
$i \rightarrow t$	0.1084	0.0783	0.2353	-1.4036	8.1026e-02
$e \rightarrow s$	0.1255	0.2019	0.2143	-1.2893	9.8986e-02

Table 4.5: Character transitions that are more frequent in higher complexity target words (at 10% significance level).

phonetic features. Hayden (1950) suggests that the relative frequencies of phonemes can potentially be used to identify important phonemes to tutor foreign language students with who exhibit numerous phonemic difficulties. While we assume that each crowd annotator of CompLex understands at least one dialect of English, we also assume that annotators share similar literacy levels and familiarities with the English language. We hypothesize that because certain English phonemes (ie. soundable segments or character n-grams) are evidently more common in usage than others (Hayden, 1950), a target word's complexity could be impacted by an annotator's familiarity with its constituent phonemes.

For a given target word w, consider the transition probability from character $w[i]$ to the character $w[i+1]$, that is, the expected probability of the $(i+1)$th character succeeding the ith character in language; we call this a *character* transition probability. To understand whether this may be a useful feature for our task, for each target word w of class c, we define the random variable ${}_xT_y \in \mathbb{N}$ as the number of occurences of character transition $x \rightarrow y$ in the target word, where $x, y \in \{\text{ASCII character set}\}$. For each possible transition $x \rightarrow y$, we ask whether the distribution of ${}_xT_y$ over class 1 samples (ie. very easy target words) is different from that over class 4, 5 samples (ie. difficult target words).[2] Let μ_1 and

[2]Here, we combine class 4 and 5 samples together because so few samples exist in each individual class.

$\mu_{4,5}$ be the average number of occurences of the given transition $x \rightarrow y$ over class 1 and class 4, 5 target words, respectively. We conduct an unpaired lower-tailed t-test using the null hypothesis that H_0: $\mu_1 = \mu_{4,5}$, and alternative hypothesis that H_a: $\mu_1 < \mu_{4,5}$. Table 4.5 presents only the character transitions that yield statistically significant p-values at the 10% confidence level, implying that these character transitions are slightly more common in higher complexity target words than in lower complexity target words. Note that we also conducted t-tests with instead the alternative hypothesis H_a: $\mu_1 > \mu_{4,5}$, finding that *no* character transition yields a statistically significant p-value at the 10% confidence level.

We attempt to intuit the statistically significant character transitions shown in Table 4.5. Notice that character transition probabilities estimated over target word classes lie roughly in the same ballpark as that estimated over Gigaword in Figure 4.3. Deviation from the Gigaword estimates is somewhat expected, considering the relatively small size of the CompLex corpus. Seven of the eight character transitions (ie. all except $o \rightarrow n$) bear higher transition probabilities over class 4, 5 samples than over class 1 samples, perhaps suggesting that target word complexity is correlated with the existence of transitions between certain characters. Based on Figure 4.3, observe that y often transitions to an e, s, o, etc. whereas rather rarely does it transition to c. Yet, we find class 4, 5 samples tending to towards this rarity nonetheless, often occuring in scientific words with the root 'cycl,' such as 'cycle,' 'cyclone,' and 'doxycycline.' Other examples of this apparent tendency for higher-difficulty samples to exhibit rarer transitions include $i \rightarrow d$, $o \rightarrow s$, and $p \rightarrow h$. This subtle pattern suggests that the transition probabilities between characters (and possibly even phonemes) in a target word could serve as a discriminative feature for predicting lexical complexity.

Chapter 5

BigGreen System & Approaches

In this section, we overview information fed to the feature engineering-based model, as well as training techniques for the feature learning-based model. We describe our features in finer detail in Appendix A. Note that the fitted models for the single word subtask are later harnessed for the MWE subtask.

Section 5.1

Feature Engineering-based Approach

5.1.1. Feature Extraction

We aim to capture a breadth of information pertaining to the target word and its context. The majority of features appear to follow heavily right-skewed distributions, which we attribute to Zipf's law manifesting over our plethora of word frequency-based measures. This prompts us to also consider the $\log$-transformed version of each feature. For the MWE subtask, features are extracted independently for the head and tail words, with they and their *sums* being included in the final feature set.

Lexical Features. These features capture lexical information pertaining to the target word:

- **Word length**: length of the target word.

- **Number of syllables**: number of syllables in the target word, via the Syllables library.
- **Is acronym**: whether the target word is a sequence of capital letters.

Semantic Features. These features capture the target word's meaning:

- **WordNet features**: the number of hyponyms and hypernyms associated with the target word in WordNet (Fellbaum, 2010).
- **GloVe word embeddings**: we extract 300-dimension embeddings trained on Wikipedia-2014 and Gigaword (Pennington et al., 2014) for each (lowercased) target word.
- **ELMo word embeddings**: we extract 1024-dimension embeddings trained on the One Billion Word Benchmark corpus (Peters et al., 2018) for each target word. Observe that these are *contextualized* embeddings, unlike our GloVe word embeddings.
- **GloVe context embeddings**: we obtain the average 300-dimension GloVe word embedding across all words in the given sentence.
- **InferSent context embeddings**: we obtain 4096-dimension InferSent embeddings (Conneau et al., 2017) for each sentence.

Phonetic Features. These features compute the likelihood that soundable portions of the target word would arise in English language. We estimate ground truth transition probabilities between any two units (phonemes or characters) using Gigaword:

- **Phoneme transition probability**: we consider the min/maximum/mean/standard deviation over the set of transition probabilities for the target word's phoneme bigrams.
- **Character transition probability**: analogous to that above, over *character* bigrams.

Word Frequency & N-gram Features. These features are expressly included due to their expected importance as features (Zampieri et al., 2017). Gigaword is the main corpus from which we extract word frequency measures (for both lemmatized and unlemmatized versions of the target word), the summed frequency of a target word's byte pair encodings (BPEs), as well as summed frequencies of bigrams and trigrams containing the target word. We complement these features with their IDF-based analogues. Finally, we use the GBND, BNC, and SUBTLEXus corpora to extract secondary unigram, bigram, and trigram measures.

Syntactic Features. These are features that assess the syntactic structure of the target word's context. We construct the constituency parse tree for each sentence using a Stanford CoreNLP pipeline (Manning et al., 2014).

- **Part of speech (POS)**: predicted via NLTK's `pos_tag` method (Bird et al., 2009).
- **Depth of parse tree**: the parse tree's height.
- **Depth of target word**: distance (in edges) between the target word and the parse tree's root node.
- **Number of words at target depth**: number of words at the same depth in the parse tree as the target word.
- **Is proper**: whether target word is a proper noun/adjective, detected via capitalization.

Readability Metrics. These comprise a variety of tests applied on the target word's context, using low-level traits such as total word count and total syllable count. Interestingly, certain readability metrics count the difficult words in a given sentence by assuming rules for what makes a given word *complex* (eg. the Dale-Chall readability formula (Dale and Chall, 1948) checks a given word against a predetermined list of 3,000 familiar words). This inspires

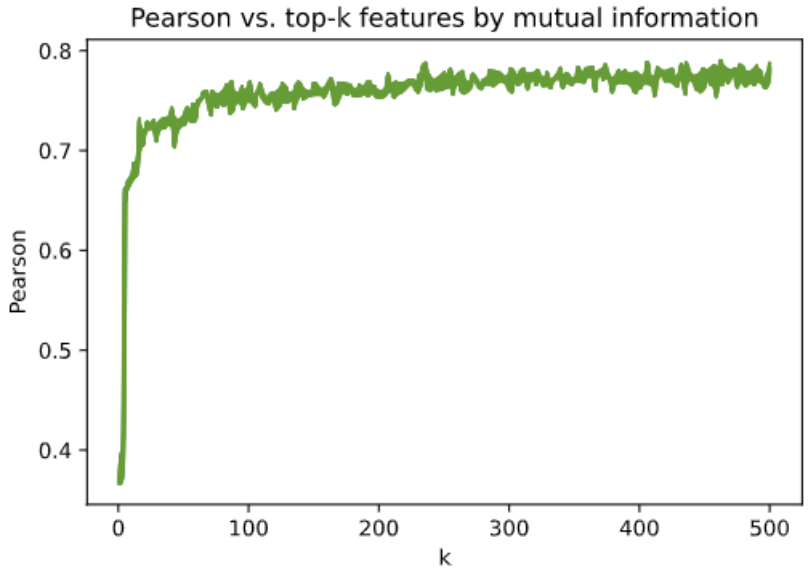


Figure 5.1: Performances of linear regression models fitted (with 5-fold cross-validation) on top-k features by mutual information.

us to try multiple readabililty measures via the Textstat library,[1] including Flesch-Kincaid grade level, Gunning Fog index, and SMOG index.

5.1.2. Feature Selection

For the single word subtask, we select features using a combination of filter and wrapper methods. Our intention is to leverage successful techniques in the MWE subtask as well, where we extract head and tail-specific features.

Filter Methods.

- **Variance**: Features are screened by variance of their distributions, with those lower than 0.01 deemed quasi-constant and subsequently removed.
- **Mutual Information**: Features are ranked by mutual dependence with lexical complexity, and only the top-k features are selected. We tune k by fitting linear regression

[1] https://github.com/shivam5992/textstat

models on the top-k features. Figure 5.1 shows diminishing improvement in Pearson correlation beyond $k = 300$.

- **Variance Inflation Factor (VIF)**: This is computed for each feature to measure contributed multicollinearity. Note that we omit this particular filter method for our submitted model, in order to optimize our model's Pearson correlation coefficient.

Wrapper Methods.

- **Forward Feature Selection (FFS)**: Beginning with an empty feature set, each subsequent iteration appends a feature to the existing feature set offering the best Pearson correlation. The algorithm exits when no feature sufficiently improves correlation.

Embedded Methods.

- **Lasso & Elastic Nets**: We consider these linear models during the subsequent training phase, which use L1 and L1/L2 regularization, respectively, to shrink regression coefficients of lesser important features *during* fitting. Lasso (Tibshirani, 1996) is intriguing due to its ability to reduce the dimensionality of our feature set during fitting. We try Elastic Net (Zou and Hastie, 2005) as it may succeed particularly in the presence of highly intercorrelated features.

5.1.3. Training

Prior to training, we Z-score standardize all features to have approximately zero mean and unit variance. For the single word subtask, we fit Linear, Lasso, Elastic Net, Support Vector Machine (Platt et al., 1999) (with linear kernel), Support Vector Machine (with radial basis function kernel), K-Nearest Neighbors (Wikipedia, 2021), and XGBoost (Chen and Guestrin, 2016) regression models. After identifying the best performing model by Pearson correlation, we seek to mitigate the imbalanced nature of the target variable, ie. multitude of

class 1, 2, 3 and lack of class 4, 5 samples: we devise a sister version of our top-performing model, fit upon a *reduced* training set. For the *reduced* set, we tune percentages removed from classes 1-3 by performing cross-validation on the full training set.

Section 5.2

Feature Learning-based Approach

Our handcrafted feature set provides a cursory analysis of the context surrounding the target word. We seek an alternative, automated approach using feature learning.

5.2.1. Background

LSTM-based approaches have been used to model contexts of target words in past works (Hartmann and Dos Santos, 2018; De Hertog and Tack, 2018). An issue with a single LSTM is its ability to read tokens of an input sentence sequentially only in a single direction (eg. left-to-right). It inspires us to try a Transformer-based approach (Vaswani et al., 2017), architectures that process sentences as a whole (instead of word-by-word) by applying *attention* mechanisms upon them. Attention weights are useful, even interpretable as learned relationships between words in a given text. BERT (Devlin et al., 2018) is one such language representation model used for a variety of natural language understanding (NLU) tasks.

Multi-Task Deep Neural Network (MT-DNN) proposed by Liu et al. (2019) offers state-of-the-art results for multiple NLU tasks by incorporating benefits of both multi-task learning and language model pre-training. Multi-task learning (MTL) is the process of applying knowledge learned from prior, related tasks to help learn to predict on a new task. Liu et al. (2019) notes an advantage of MTL that is being able to conduct supervised learning in the presence of relatively few training samples, which is notable considering the limited size of the CompLex corpus (about 8000 single word training samples). Language model pre-training leverages universal representations learned by neural network language

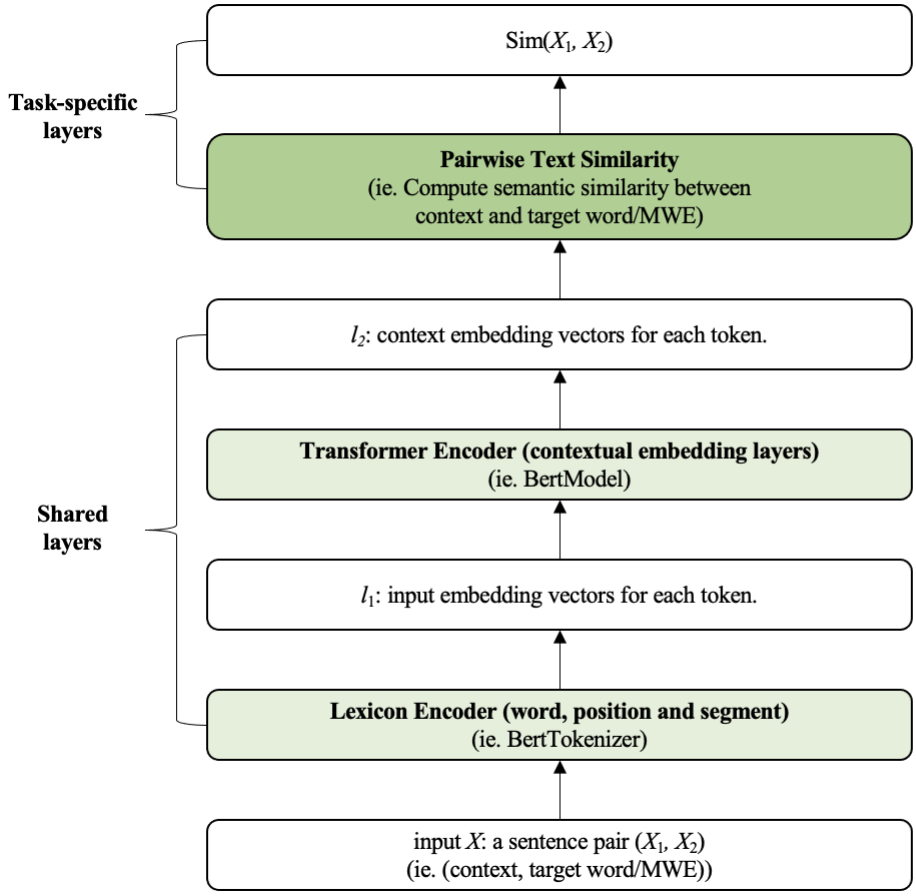


Figure 5.2: Architecture of MT-DNN specifically for a regression-based task like STS-B. This diagram is inspired by that shown by Liu et al. (2019). Note that we are able to refer to this architecture for the purposes of lexical complexity prediction because the training procedures for both tasks are quite similar.

models that have been fitted on large amounts of unlabeled data; notable examples of such models include BERT (Devlin et al., 2018) and ELMo (Peters et al., 2018). In this section, we describe the structure of the general-purpose MT-DNN model when harnessed for lexical complexity prediction (a regression-based task, mind you). Since lexical complexity prediction is a regression-based task, we apply a procedure similar to that used by Liu et al. (2019) to fine-tune MT-DNN for another similar task: STS-B from the General Language Understanding Evaluation (GLUE) benchmark (Wang et al., 2018). STS-B asks to predict a continuous value in range 0-1 indicating the semantic similarity between two given sentences. We will see that the architectures for STS-B (shown in Figure 5.2) and for the task of lexical complexity prediction are analogous to one another.

5.2.2. Lexicon Encoder

Samples are fed to MT-DNN's input layer in *PremiseAndOneHypothesis* format; this means the model expects each input to be a pair of sentences: premise and hypothesis. We let hypothesis and premise be the target word/MWE and its context, respectively. Inputs are preprocessed by a lexicon encoder to convert text to vectorized sequences of BPE tokens; our lexicon encoder is a BERT Tokenizer, backed by Hugging Face (Wolf et al., 2020).

5.2.3. Transformer Encoder

The BERT base model (cased) is one such Transformer whose weights can be used to initialize MT-DNN's shared text encoding layers. We choose the base (rather than large) model due to limited computational power of our machine, and we use its cased (rather than uncased) version due to experimentation in Section 4.2 showing a word's lexical complexity being affected by whether it is 'proper' (which we identify based on capitalization). These layers comprise what is called a *transformer encoder*, which maps input representations vectors (obtained from the lexicon encoder) to contextualized embedding vectors.

5.2.4. Task-Specific Prediction

Recall that the transformer encoder produces a semantic representation for each training set sentence pair (X_1, X_2) (where X_1 and X_2 are a context and target word/MWE, respectively). This semantic representation can be used to compute a similarity score $\text{Sim}(X_1, X_2)$ (Liu et al., 2019). Given the ground truth lexical complexity of the training sample, we propose using mean squared error as the objective function for propagating error between predicted and ground truth complexities; note that this procedure is nearly identical to that used for STS-B, the exceptions being (1) the training data used, and (2) the target word/MWE itself serves as what would have been the second sentence for semantic similarity prediction. By fine-tuning MT-DNN, we aim to repurpose its existing output layer to produce predicted lexical complexity values, ie. continuous values in range 0-1. Note that hyperparameters used for fine-tuning MT-DNN are listed in Appendix B.2. Additionally, we extract *attention maps* across each of the model's attention heads, for each test set sample; in other words, $(12 \text{ layers}) \times (12 \text{ heads per layer}) \implies 144$ attention maps for each test set sample.

Section 5.3

Ensembling

Recall that our best performing feature engineering-based regression model yields two sets of predictions (from fitting on *full* and *reduced* training sets, respectively). We default to using the *full* predictions, then tune a threshold, where predictions higher than the threshold (likely of class 4, 5 samples) are overwritten with the *reduced* predictions. We compute a weighted average ensemble of these predictions with those of our MT-DNN model to obtain a final set of predictions for the single word subtask.

For the MWE subtask, fitted models from the previous subtask are used to predict lexical complexities for constituent head and tail words. We compute a weighted average ensemble of the predicted complexities *and* predictions of an MT-DNN model trained on MWEs.

Chapter 6

Results

We present performances of `BigGreen`'s system on each subtask in Tables 6.1 and 6.2.

Model	**Pearson**	**Rank**	**Spearman**	**MAE**
Linear Regression	0.7347	-	0.6993	0.0669
Forward Feature Selection (FFS)	0.7313	-	0.7053	0.0671
Lasso (alpha=0.0001)	0.7352	-	0.7042	0.0667
ElasticNet (alpha=0.001)	0.7396	-	0.7122	0.0662
SVM (kernel=linear, C=0.001)	0.7254	-	0.6970	0.0678
SVM (kernel=rbf, C=1, gamma=0.001)	0.7392	-	0.7066	0.0668
KNN (k=20, weights=distance)	0.7156	-	0.6914	0.0710
XGBoost$_{full}$	0.7589	-	0.7220	0.0645
XGBoost$_{reduced}$	0.7456	-	0.7157	0.0751
XGBoost$_{full+reduced}$	0.7576	-	0.7220	0.0646
MT-DNN	0.7484	-	0.7044	0.0664
Ensemble (submission)	**0.7749**	**8 of 54**	**0.7294**	**0.0629**
Best competition results	0.7886		0.7425	0.0609

Table 6.1: Test set results for single word subtask.

Model	**Pearson**	**Rank**	**Spearman**	**MAE**
XGBoost$_{full+reduced}$ (head)	0.7164	-	0.7305	0.1281
XGBoost$_{full+reduced}$ (tail)	0.7188	-	0.7416	0.1306
MT-DNN	0.7890	-	0.7649	**0.0766**
Ensemble (submission)	0.7898	25 of 37	0.7769	0.0903
Ensemble (post-competition)	**0.8290**	***14 of 37**	**0.8120**	0.0857
Best competition results	0.8612		0.8548	0.0616

Table 6.2: Test set results for MWE subtask. (* indicates a projection)

Chapter 7

Analysis

Section 7.1

Performances

For feature selection, we find success in selecting the top-300 features by mutual information and removing quasi-constant features. The pruned feature set is passed to both our wrapper/embedded methods and a variety of regressors for model comparison, where we find an XGBoost regressor (with hyperparameters tuned using grid search, as listed in Appendix B.1) to excel consistently for the single word subtask. As shown in Table 6.1, our system ranks in the top 15% by Pearson correlation.

For the MWE subtask, performances are reported in Table 6.2. Note that our submitted predictions differ from post-competition predictions. We *previously* used a training procedure resembling that used for the single word subtask: (1) filter methods for feature selection, (2) XGBoost for regression, (3) assembly with MT-DNN. We had passed the entire MWE as input to our XGBoost and MT-DNN models. We hypothesize that the fewer number of training samples available for the MWE subtask contributed to the previous procedure's lackluster performance. This inspired us to incorporate the predictive capabilities of our fitted single word subtask models by applying them *independently* on the MWE's constituent

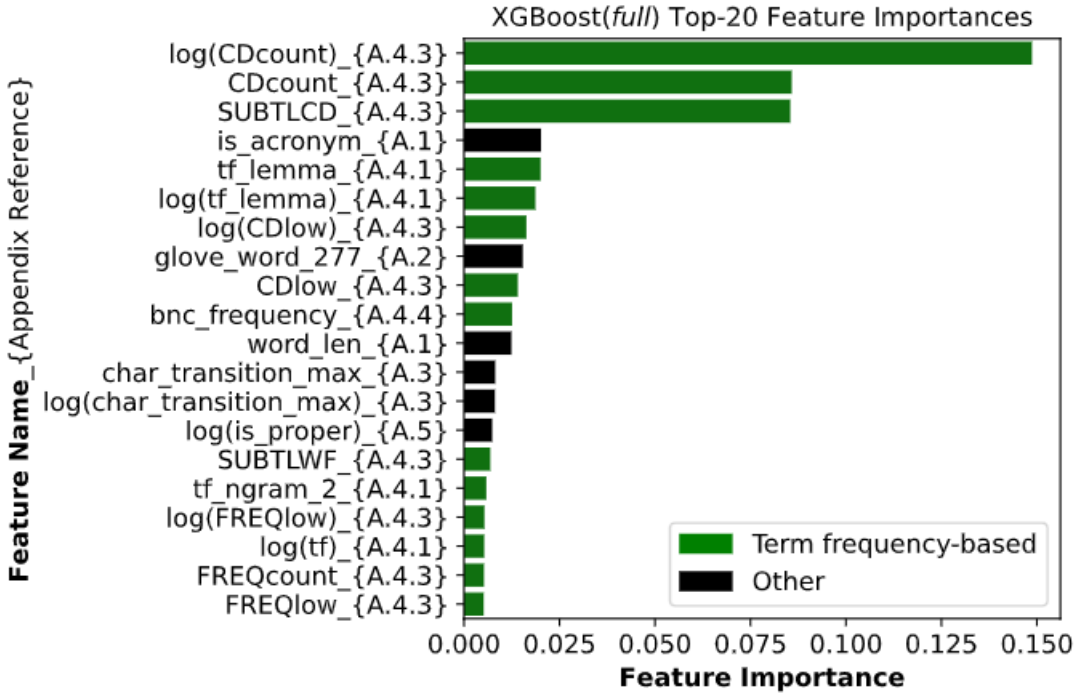


Figure 7.1: Feature importances for XGBoost$_{full}$. Definitions of features are in Appendix A.

head and tail words. This gives us predicted complexities for each of the head and tail words, which when ensembled with the predictions of our MT-DNN model (that, mind you, is trained on the *entire* MWE) yields superior results to those submitted to competition.

Section 7.2

Feature Contribution

In total we consider 110 features, in addition to multidimensional embedding-based features and log-transformed features. We inspect the estimated feature importance scores produced by the XGBoost$_{full}$ model to find that term frequency-based features (eg. unigrams, bigrams, trigrams) are of overwhelming importance (see Figure 7.1). This raises concern over whether the MT-DNN model also relies on term frequencies to make its predictions. Note that of the remaining features with non-zero feature importances (not seen in Figure 7.1), most appear to be dimensions of a target word-based semantic feature (ie. GloVe or ELMo embeddings).

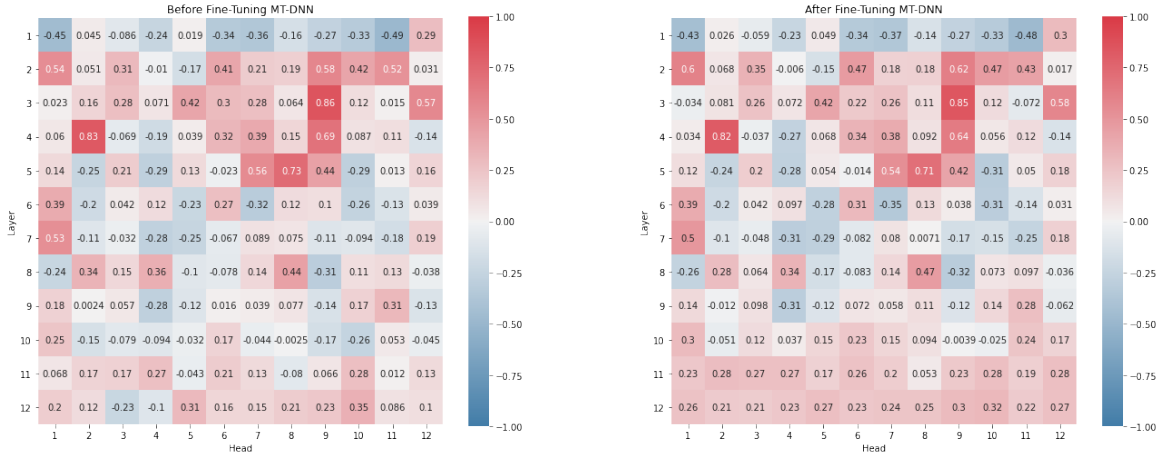


Figure 7.2: Correlation heatmaps before and after fine-tuning MT-DNN, showing strength of association between between word frequency and total attention received by word (computed over 100 random test set samples) at each attention head.

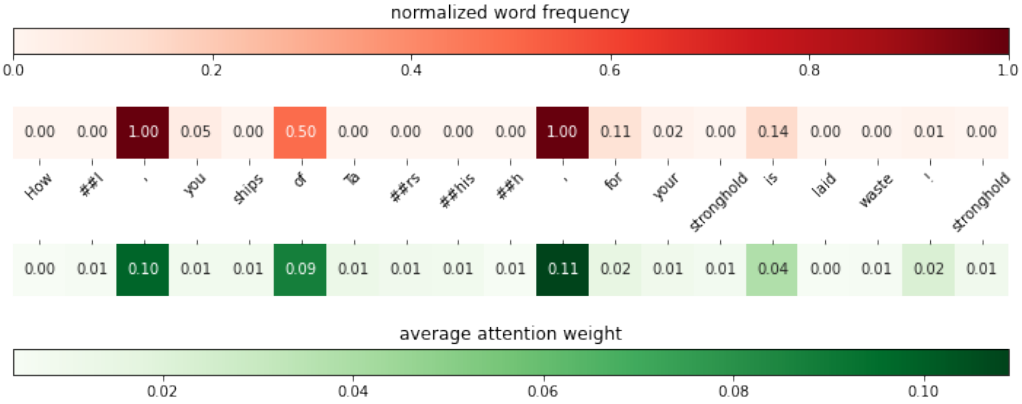
Section 7.3

BERT Attention

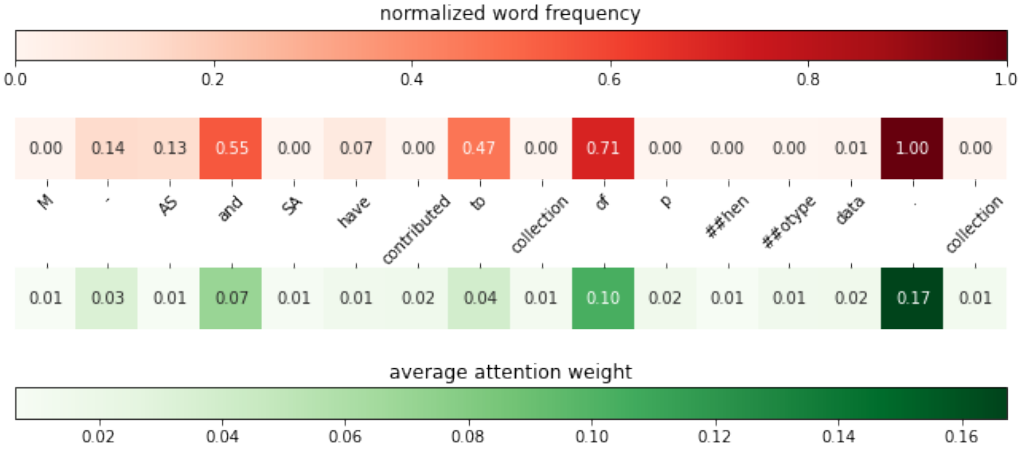
Attention maps of Transformers have in previous works been assessed to expose linguistic phenomena learned by specialized attention heads (Voita et al., 2019; Clark et al., 2019), and to measure the relative contribution of each attention head towards making task-specific predictions (Voita et al., 2019; Michel et al., 2019). We extract attention maps from our fine-tuned MT-DNN model's underlying BERT architecture. For each sample amongst 100 random samples from the single word test set, we obtain an attention map from each of the fine-tuned MT-DNN model's shared text encoding layers' 144 attention heads.

Based on our prior findings on the potential importance of term frequency-based features towards the performance of the XGBoost$_{full}$ model, we hypothesize that at certain attention heads, the average attention given to a word varies relative to the word's rarity in lexicon. This follows the findings of Voita et al., 2019, who identify heads at which lesser frequent tokens are attended to semi-uniformly by all other sentence tokens.

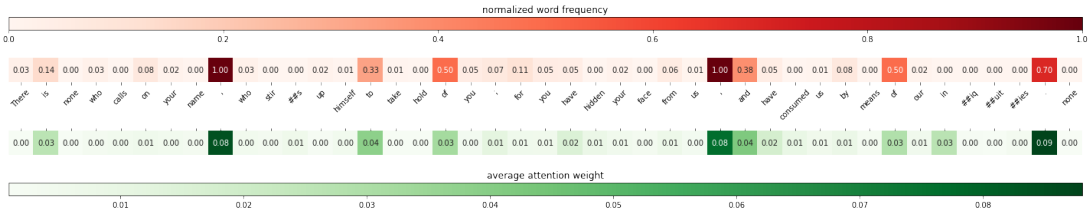
To test this hypothesis, we estimate for each attention head the Pearson correlation



(a) Test set sample 279

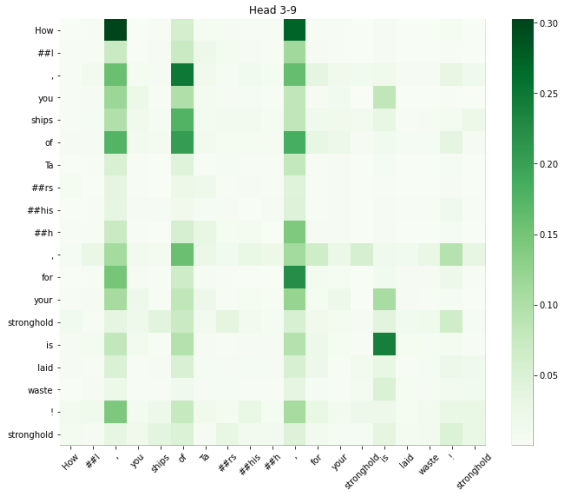


(b) Test set sample 334

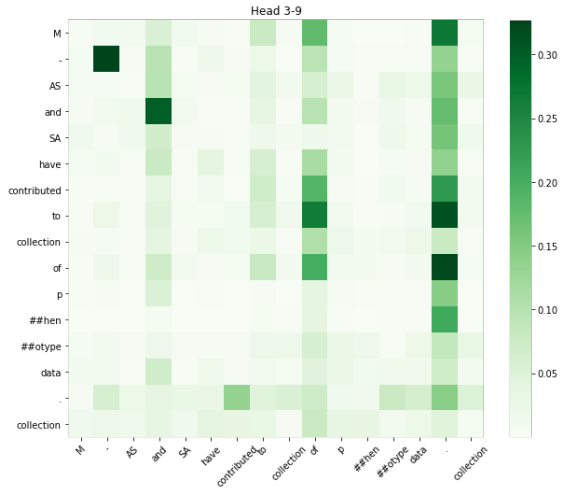


(c) Test set sample 244

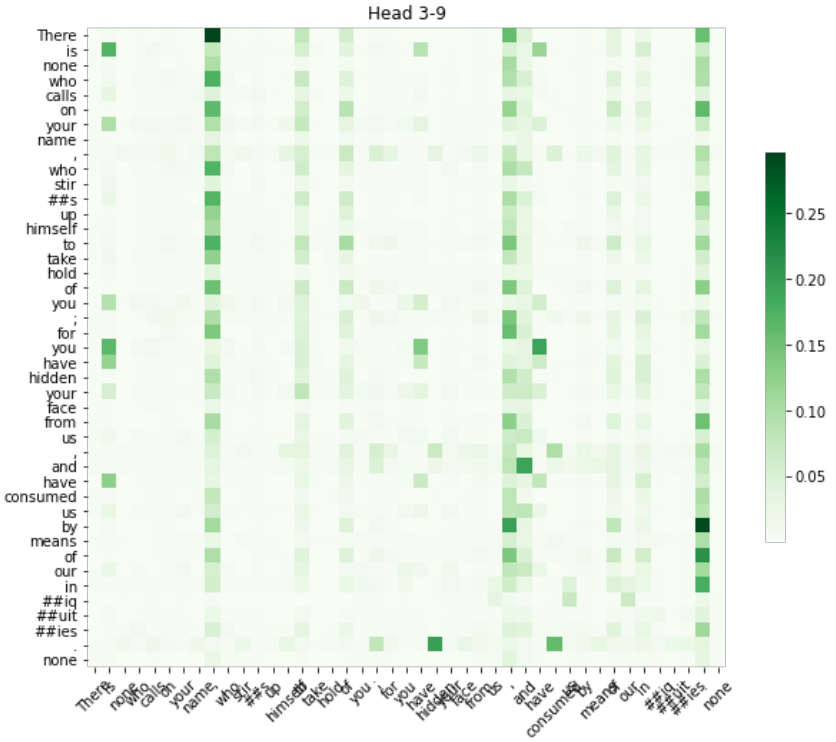
Figure 7.3: Here, we consider three arbitrary single word test set samples: 279, 334, and 244. For each sample, we compute the word frequency of each word in the sentence, relative to the GBND corpus. For simplicity, we assume the word frequency of any BPE is identical to the word frequency of its parent word. We then compute the average attention weight directed to each token in the context at head 3-9 of our fine-tuned MT-DNN model.



(a) Test set sample 279



(b) Test set sample 324



(c) Test set sample 244

Figure 7.4: Here, we once again study test set samples 279, 334, and 244. For each sample, we visualize the head 3-9 attention weight directed between any two tokens in the context.

between word frequency and the average attention weight directed to each word in the given context. Note that a cell (i, j) of any attention map contains the attention weight directed from the ith token to the jth token in the sentence, by the given head.[1] As illustrated in Figure 7.2, there appear to be multiple attention heads specializing at directing attention towards either the most or least (which we disambiguate via the sign of the correlation) frequent words in language. It seems as though the standalone MT-DNN model (ie. before fine-tuning) already possesses such specialized heads. Fine-tuning seems to generally improve correlations for heads in later layers (namely 11th and 12th) by approximately $+0.1$ points on average. Both of these findings are a testament to the versatility of MT-DNN's underlying Transformer-based architecture. In Figure 7.3, we examine sentences from three arbitrary single word test set samples, namely by visualizing attention maps for these sentences at a particularly intruiging head: 3-9. For each token in the given sentence, we compare the average attention directed by head 3-9 towards the token versus the token's corresponding word frequency. One may notice disproportionate attention directed to punctuation (eg. periods, commas) as well as stopwords (eg. 'and,' 'is,' 'of'). Such characters and words are relatively common in corpora, resulting in exceptionally high word frequencies. In contrast, little to no attention is directed towards rare nouns (eg. 'stronghold,' 'collection'), obscure acronyms (eg. 'AS,' 'SA') and rare past tense verbs (eg. 'consumed,' 'contributed').

Vertical stripe patterns like that in Figure 7.4 emerge as a result of attention originating from a spectrum of tokens. A shortcoming of this examination is that the roles played by tokens responsible for the apparent vertical stripe patterns over particular tokens (such patterns generally occur over punctuation or stopwords, as discussed previously) remains unclear. Nonetheless, these findings seem to affirm the fundamental relevancy of word frequency to lexical complexity prediction, corroborating our intuitions.

[1] We define attention directed to a *word* as the sum of attention weights directed to its constituent BPEs. For example, 'Howl' and 'Tarshish' from the sentence shown in Figure 7.3a have their total attention weights broken across multiple BPEs.

Chapter 8

Conclusion

In this paper, we report inspirations for a system submitted by `BigGreen` to LCP Shared-Task 2021, performing reasonably well for the single word subtask by adapting ensemble methods upon feature engineering and feature learning-based models. We see potential in future deep learning approaches, acknowledging the need for strong word frequency-based handcrafted features for the time being. We surpass our submitted results for the MWE subtask by utilizing the predictive capabilities of our single word subtask models, under the assumption that MWE complexity is compositional with respect to its constituent tokens.

Avenues for improvement include better data aggregation, as a relative lack of class 4, 5 samples hurts Pearson correlation across samples of especially extreme complexity. Such an approach may involve synthetic data generation using SMOGN (Branco et al., 2017), for instance. Although Shardlow et al. (2020) acknowledge that a reader's familiarity with a genre may affect his/her perceived complexity of a word, the CompLex corpus unfortunately lacks details on each annotator's expertise or background, which may have offered valuable new insights. Future studies may even consider extracting multidimensional embeddings from the later layers of deep learning models, potentially feeding these embeddings directly into one's feature engineering-based models.

Appendix A

Feature Descriptions

Here, we describe in greater detail the various features that were experimented with for our feature engineering-based model. Note that while this discussion regards the single word subtask, for the MWE subtask we compute the same features but for each of the head and tail words, respectively.

Section A.1

Lexical Features

Feature	Description
word_len	Character length of the target word.
num_syllables	Number of syllables in the target word, via the Syllables library.
is_acronym	Boolean for whether the target word is all capital letters.

Section A.2

Semantic Features

Feature	Description
num_hyperyms	Number of hyperyms associated with the target word. The target word is initially disambiguated using NLTK's implementation of the Lesk algorithm for Word Sense Disambiguation (WSD) (Lesk, 1986), which finds the WordNet Synset with the highest number of overlapping words between the context and different definitions of each Synset.
num_hyponyms	Number of hyponyms associated with the target word. Procedure for finding this is analogous to that for num_hyperyms.
glove_word	300-dimension embedding for each target word, pretrained on Wikipedia-2014 and Gigaword. The target word is lowercased for simplicity.
elmo_word	1024-dimension embedding for each target word, pretrained on the One Billion Word Benchmark corpus.
glove_context	300-dimension average of GloVe word embeddings (see glove_word) for each word in the given context. Each word is lowercased for simplicity.
infersent_embeddings	4096-dimension embedding for the context.

Section A.3

Phonetic Features

Feature	Description
char_transition_min	Minimum of the set of character transition probabilities for each character bigram in the target word. Ground truth character transition probabilities between any two English characters are estimated over Gigaword.
char_transition_max	Maximum of the set described above.
char_transition_mean	Mean of the set described above.
char_transition_std	Standard deviation of the set described above.
phoneme_transition_min	Minimum of the set of phoneme transition probabilities for each character bigram in the target word. Ground truth phoneme transition probabilities between any two phonemes are estimated over the Gigaword corpus. The phoneme set considered is that of the CMU Pronouncing Dictionary.[1]
phoneme_transition_max	Maximum of the set described above.
phoneme_transition_mean	Mean of the set described above.
phoneme_transition_std	Standard deviation of the set described above.

[1] http://www.speech.cs.cmu.edu/cgi-bin/cmudict

Section A.4

Word Frequency & N-gram Features

A.4.1. Gigaword-based

Feature	Description
`tf`	Term frequency of the target word. Note that all term frequency-based features are computed using the Scikit-learn library's `CountVectorizer` (Pedregosa et al., 2011).
`tf_lemma`	Term frequency of the lemmatized target word. Lemmatization is performed using NLTK's WordNet Lemmatizer.
`tf_summed_bpe`	Sum of the term frequencies of each BPE of the target word. BPE tokenization is performed using Hugging Face's BERT Tokenizer.
`tf_ngram_2`	Sum of the term frequencies of each bigram in the context containing the target word.
`tf_ngram_3`	Sum of the term frequencies of each trigram in the context containing the target word.
`tfidf`	Term frequency-inverse document frequency.
`tfidf_ngram_2`	Sum of the term frequency-inverse document frequencies of each bigram in the context containing the target word.
`tfidf_ngram_3`	Sum of the term frequency-inverse document frequencies of each trigram in the context containing the target word.

A.4.2. Google N-gram-based

Feature	Description
google_ngram_1	Term frequency of the target word.
google_ngram_2_head	Term frequency of leading bigram in context containing the target word.
google_ngram_2_tail	Term frequency of trailing bigram in context containing the target word.
google_ngram_2_min	Minimum of the set of term frequencies of each bigram in context containing the target word.
google_ngram_2_max	Maximum of the set described above.
google_ngram_2_mean	Average of the set described above.
google_ngram_2_std	Standard deviation of the set described above.
google_ngram_3_head	Term frequency of leading trigram in context containing the target word.
google_ngram_3_mid	Term frequency of middle trigram in context containing the target word.
google_ngram_3_tail	Term frequency of trailing trigram in context containing the target word.
google_ngram_3_min	Minimum of the set of term frequencies of each trigram in context containing the target word.
google_ngram_3_max	Maximum of the set described above.
google_ngram_3_mean	Average of the set described above.
google_ngrams_3_std	Standard deviation of the set described above.

A.4.3. SUBTLEXus-based

Feature	Description
FREQcount	Number of times the target word appears in corpus.
CDcount	Number of films in which the target word appears.
FREQlow	Number of times the lowercased target word appears in corpus.
CDlow	Number of films in which the lowercased target word appears.
SUBTLWF	Number of times the target word appears per million words.
SUBTLCD	Percentage of films in which the target word appears.

A.4.4. BNC-based

Feature	Description
bnc_frequency	Term frequency of the target word.

Section A.5

Syntactic Features

Feature	Description
parse_tree_depth	Height of context's constituency parse tree. We get tree using Stanford CoreNLP pipeline.
token_depth	Depth of the target word with respect to root node of the context's constituency parse tree.
num_words_at_depth	Number of words at the depth of the target word (see token_depth above) in the context's constituency parse tree.
is_proper	Boolean for whether the target word is a proper noun/adjective, based on capitalization.
POS_{CC, CD, DT, EX, FW, IN, JJ, JJR, JJS, LS, MD, NN, NNP, NNPS, NNS, PDT, POS, PRP, PRP$, RB, RBR, RBS, RP, SYM, TO, UH, VB, VBD, VBG, VBN, VBP, VBZ, WDT, WP, WP$, WRB}	Booleans for whether the target word's part-of-speech tag is such. Tags considered are those used in the Penn Treebank Project.[2] Tags are estimated using NLTK's pos_tag method.

[2] https://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html

Section A.6

Readability Features

Feature	Description
automated_readability_index, avg_character_per_word, avg_letter_per_word, avg_syllables_per_word, char_count, coleman_liau_index, crawford, fernandez_huerta, flesch_kincaid_grade, flesch_reading_ease, gutierrez_polini, letter_count, lexicon_count, linsear_write_formula, lix, polysyllabcount, reading_time, rix, syllable_count, szigriszt_pazos, SMOGIndex, DaleChallIndex	Algorithms applied using the Textstat library implementations, most of whom are readability metrics.

Section A.7

Other Features

Feature	Description
`ppl`	Perplexity metric, as defined by the Hugging Face library.[3] For each token in the context, we use a pre-trained GPT-2 model to estimate the log-likelihood of the token occurring *given its preceding tokens*. A sliding-window approach is used to handle the large number of tokens in a context. The log-likelihoods are averaged, and then exponentiated.
`ppl_aspect_only`	Similar approach to that described above, where only log-likelihoods of tokens comprising the target word are averaged.
`num_OOV`	Number of words in the context that do not exist in the vocabulary of Gigaword.
`corpus_bible`, `corpus_biomed`, `corpus_europarl`	Booleans indicating the sample's domain.

[3] https://huggingface.co/transformers/perplexity.html

Appendix B

Model Hyperparameters

Here, we provide optimized hyperparameter settings that may help future developers with reproducing our results, namely for training our models.

Section B.1

XGBoost

Below are tuned parameters used for all of our XGBoost models. Parameters not listed are given their default values as specified in XGBoost documentation:[1]

`colsample_bytree`: 0.7

`learning_rate`: 0.03

`max_depth`: 5

`min_child_weight`: 4

`n_estimators`: 225

`nthread`: 4

`objective`: 'reg:linear'

`silent`: 1

`subsample`: 0.7

[1] https://xgboost.readthedocs.io/en/latest/

Section B.2

MT-DNN

MT-DNN uses `yaml` as its config file format. Below are the contents of our task config file:

`data_format`: PremiseAndOneHypothesis

`enable_san`: false

`metric_meta`:

- Pearson

- Spearman

`n_class`: 1

`loss`: MseCriterion

`kd_loss`: MseCriterion

`adv_loss`: MseCriterion

`task_type`: Regression

Section B.3

Ensemble

Threshold above which a sample is assigned its *reduced* prediction (ie. XGBoost$_{reduced}$ prediction) instead of its *full* prediction (ie. XGBoost$_{full}$ prediction): 0.59. Note that this threshold is used to compute our XGBoost$_{full+reduced}$ prediction.

Regarding weighted average ensemble (single word subtask):

- Weight for XGBoost$_{full+reduced}$ prediction: 0.5
- Weight for MT-DNN prediction: 0.5

Regarding weighted average ensemble (MWE subtask):

- Weight for XGBoost$_{full+reduced}$(head): 0.28

- Weight for XGBoost$_{full+reduced}$(tail): 0.17
- Weight for MT-DNN prediction: 0.55

Bibliography

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural language processing with Python: analyzing text with the natural language toolkit*. O'Reilly Media, Inc.

Paula Branco, Luís Torgo, and Rita P Ribeiro. 2017. SMOGN: A pre-processing approach for imbalanced regression. In *First international workshop on learning with imbalanced domains: Theory and applications*, pages 36–50. PMLR.

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.

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.

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D Manning. 2019. What does BERT look at? An analysis of BERT's attention. *arXiv preprint arXiv:1906.04341*.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loic Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. *arXiv preprint arXiv:1705.02364*.

BNC Consortium et al. 2007. British national corpus. *Oxford Text Archive Core Collection*.

Edgar Dale and Jeanne S Chall. 1948. A formula for predicting readability: Instructions. *Educational research bulletin*, pages 37–54.

Dirk De Hertog and Anaïs Tack. 2018. Deep Learning Architecture for ComplexWord Identification. In *Thirteenth Workshop of Innovative Use of NLP for Building Educational Applications*, pages 328–334. Association for Computational Linguistics (ACL); New Orleans, Louisiana.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Siobhan Devlin. 1998. The use of a psycholinguistic database in the simplification of text for aphasic readers. *Linguistic databases*.

Siobhan Lucy Devlin. 1999. *Simplifying natural language for aphasic readers.* Ph.D. thesis, University of Sunderland.

Christiane Fellbaum. 2010. Wordnet. In *Theory and applications of ontology: computer applications*, pages 231–243. Springer.

Nathan Hartmann and Leandro Borges Dos Santos. 2018. NILC at CWI 2018: Exploring feature engineering and feature learning. In *Proceedings of the Thirteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 335–340.

Rebecca E Hayden. 1950. The relative frequency of phonemes in General-American English. *Word*, 6(3):217–223.

Aadil Islam, Weicheng Ma, and Soroush Vosoughi. 2021. BigGreen at SemEval-2021 Task 1: Lexical Complexity Prediction with Assembly Models.

Michael Lesk. 1986. Automatic sense disambiguation using machine readable dictionaries: how to tell a pine cone from an ice cream cone. In *Proceedings of the 5th annual international conference on Systems documentation*, pages 24–26.

Yuri Lin, Jean-Baptiste Michel, Erez Lieberman Aiden, Jon Orwant, William Brockman, and Slav Petrov. 2012. Syntactic annotations for the google books ngram corpus.

Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 2019. Multi-task deep neural networks for natural language understanding. *arXiv preprint arXiv:1901.11504*.

Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*, pages 55–60.

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

Paul Michel, Omer Levy, and Graham Neubig. 2019. Are sixteen heads really better than one? *arXiv preprint arXiv:1905.10650*.

Niloy Mukherjee, Braja Gopal Patra, Dipankar Das, and Sivaji Bandyopadhyay. 2016. JUNLP at SemEval-2016 Task 11: Identifying complex words in a sentence. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 986–990.

Gustavo Paetzold and Lucia Specia. 2016a. Semeval 2016 task 11: Complex word identification. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 560–569.

Gustavo Paetzold and Lucia Specia. 2016b. SV000gg at Semeval-2016 Task 11: Heavy

gauge complex word identification with system voting. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 969–974.

R Parker, D Graff, J Kong, K Chen, and K Maeda. 2011. English Gigaword Fifth Edition LDC2011T07 (tech. rep.). Technical report, Technical Report. Linguistic Data Consortium, Philadelphia.

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 D 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.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

John Platt et al. 1999. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in large margin classifiers*, 10(3):61–74.

Keith Rayner and Susan A Duffy. 1986. Lexical complexity and fixation times in reading: Effects of word frequency, verb complexity, and lexical ambiguity. *Memory & cognition*, 14(3):191–201.

Francesco Ronzano, Luis Espinosa Anke, Horacio Saggion, et al. 2016. TALN at SemEval-2016 Task 11: Modelling complex words by contextual, lexical and semantic features. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 1011–1016.

Matthew Shardlow. 2013. A Comparison of Techniques to Automatically Identify Complex Words. In *51st Annual Meeting of the Association for Computational Linguistics Proceedings of the Student Research Workshop*, pages 103–109.

Matthew Shardlow. 2014. Out in the Open: Finding and Categorising Errors in the Lexical Simplification Pipeline. In *LREC*, pages 1583–1590.

Matthew Shardlow, Michael Cooper, and Marcos Zampieri. 2020. CompLex: A New Corpus for Lexical Complexity Predicition from Likert Scale Data. In *Proceedings of the 1st Workshop on Tools and Resources to Empower People with REAding DIfficulties (READI)*.

Matthew Shardlow, Richard Evans, Gustavo Paetzold, and Marcos Zampieri. 2021. SemEval-2021 Task 1: Lexical Complexity Prediction. In *Proceedings of the 14th International Workshop on Semantic Evaluation (SemEval-2021)*.

Robyn Speer, Joshua Chin, Andrew Lin, Sara Jewett, and Lance Nathan. 2018. Luminosoinsight/wordfreq: v2. 2.

Robert Tibshirani. 1996. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288.

Martin Trenkmann. PhraseFinder – Search millions of books for language use. https://phrasefinder.io. Accessed: 2021-02-08.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *arXiv preprint arXiv:1706.03762*.

Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. 2019. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. *arXiv preprint arXiv:1905.09418*.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*.

Wikipedia. 2021. K-nearest neighbors algorithm — Wikipedia, the free encyclopedia. http://en.wikipedia.org/w/index.php?title=K-nearest%20neighbors%20algorithm&oldid=1008084290. [Online; accessed 02-April-2021].

Thomas Wolf, Julien Chaumond, Lysandre Debut, Victor Sanh, Clement Delangue, Anthony Moi, Pierric Cistac, Morgan Funtowicz, Joe Davison, Sam Shleifer, et al. 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.

Krzysztof Wróbel. 2016. PLUJAGH at SemEval-2016 Task 11: Simple system for complex word identification. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 953–957.

Seid Muhie Yimam, Chris Biemann, Shervin Malmasi, Gustavo H Paetzold, Lucia Specia, Sanja Štajner, Anaïs Tack, and Marcos Zampieri. 2018. A report on the complex word identification shared task 2018. *arXiv preprint arXiv:1804.09132*.

Marcos Zampieri, Shervin Malmasi, Gustavo Paetzold, and Lucia Specia. 2017. Complex word identification: Challenges in data annotation and system performance. *arXiv preprint arXiv:1710.04989*.

Qing Zeng, Eunjung Kim, Jon Crowell, and Tony Tse. 2005. A text corpora-based estimation of the familiarity of health terminology. In *International Symposium on Biological and Medical Data Analysis*, pages 184–192. Springer.

Hui Zou and Trevor Hastie. 2005. Regularization and variable selection via the elastic net. *Journal of the royal statistical society: series B (statistical methodology)*, 67(2):301–320.