BigGreen at SemEval-2021 Task 1: Lexical Complexity Prediction with Assembly Models

Aadil Islam, **Weicheng Ma**, **Soroush Vosoughi**
Department of Computer Science
Dartmouth College
{aadil.islam.21,weicheng.ma.gr,soroush.vosoughi}@dartmouth.edu

Abstract

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 with a deep neural network model founded 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 phonological measures. Visualizations of BERT attention maps offer insight into potential features that Transformers models may learn when fine-tuned for lexical complexity prediction. Our ensembled predictions score reasonably well for the single word subtask, and we demonstrate how they can be harnessed to perform well on the multi word expression subtask too.

1 Introduction

Lexical simplification (LS) is the task of replacing difficult words in text with simpler alternatives. It is relevant in reading comprehension, where early studies have shown infrequent words lead to more time spent by a reader fixated on it, and that ambiguity in a word's meaning adds to comprehension time (Rayner and Duffy, 1986). Complex word identification (CWI) is believed to be a fundamental step in the automation of lexical simplification (Shardlow, 2014). Early techniques for conducting CWI suffer from a lack of robustness, from simplifying all words to *then* study its efficacy (Devlin, 1998), to applying thresholds on features like word frequency (Zeng et al., 2005).

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 complexity on a continuous scale. This choice is intriguing as it mitigates a dilemma with previous approaches of having to treat words extremely close to a decision boundary (suppose a threshold deems a word's difficulty) 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). 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 available on GitHub.[1]

2 Related Work

Previous studies have looked at estimating the readability of a given text at the sentence-level. Mc Laughlin (1969) regresses the number of polysyllabic words in a given lesson against the mean score for students quizzed on said 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 in reading, which they find correlates with passage difficulty. An issue with traditional readability metrics seems to be the 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 a non-complex word does not require further simplification. They also try assigning a familiarity score to a word, and determining whether the word is complex or not by applying a threshold.

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

Proceedings of the 15th International Workshop on Semantic Evaluation (SemEval-2021), pages 667–677
Bangkok, Thailand (online), August 5–6, 2021.

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

Table 1: LCP train, trial, and test sets.

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 from CWI at SemEval-2016 (Zampieri et al., 2017) suggest vote ensembling predictions of the best performing models as an effective strategy, while several top-performing models (Paetzold and Specia, 2016b; Ronzano et al., 2016; Mukherjee et al., 2016) appear to use linguistic information beyond just word frequency. This inspires our use of ensemble techniques, and a foray into phonological features as a new point of research. Results from CWI at SemEval-2018 show feature engineering-based models outperforming deep learning-based counterparts, despite the latter having generally better performances since SemEval-2016.

3 Data

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 lies in range 0-1, and represents the (normalized) average score given by employed crowd workers who record an expression's difficulty on a 5-point Likert scale. We define a sample's *class* as the bin to which its complexity label belongs, where bins are formed 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 it was extracted from, drawn from one of three corpora (Bible, Biomed, and Europarl). A summary of the train, trial,[2] and test set samples is provided in Table 1.

3.2 External Datasets

We use four additional corpora to extract term frequency-based features from:

- **English Gigaword Fifth Edition** (Gigaword): this comprises articles from seven English newswires (Parker et al., 2011).
- **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).
- **British National Corpus, version 3** (BNC): this is a collection of written and spoken English text (Consortium et al., 2007).
- **SUBTLEXus**: this consists of American English subtitles, offering a multitude of word frequency lists (Brysbaert and New, 2009).

4 BigGreen System & Approaches

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

4.1 Feature Engineering-based Approach

4.1.1 Feature Extraction

We aim to capture a breadth of information pertaining to the target word and its context. Most features follow heavily right-skewed distributions, prompting us to also consider the log-transformed version of each feature. For the MWE subtask, features are extracted independently for head and tail words.

4.1.1.1 Lexical Features

These are features based on lexical information about the target word:

- **Word length**: length of the target word.
- **Number of syllables**: number of syllables in the target word, via the Syllables library.[3]
- **Is acronym**: whether the target word is a sequence of capital letters.

[2] 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).

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

4.1.1.2 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 pre-trained on Wikipedia-2014 and Gigaword (Pennington et al., 2014) for each (lowercased) target word.
- **ELMo word embeddings**: we extract for each target word a 1024-dimension contextualized embedding pre-trained on the One Billion Word Benchmark (Peters et al., 2018).
- **GloVe context embeddings**: we obtain the average 300-dimension GloVe word embedding over each token in the given sentence.
- **InferSent context embeddings**: we obtain 4096-dimension InferSent embeddings (Conneau et al., 2017) for each sentence.

4.1.1.3 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/max/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.

4.1.1.4 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), summed frequency of the target word's byte pair encodings (BPEs), as well as summed frequencies of bigrams and trigrams. We complement these features with their IDF-based analogues. Lastly, we use the GBND, BNC, and SUBTLEXus corpora to extract secondary word frequency, bigram, and trigram measures.

4.1.1.5 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)**: tag is assigned using 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 target word and parse tree's root node.
- **Is proper**: whether the target word is a proper noun/adjective, detected using capitalization.

4.1.2 Training

Prior to training, we Z-score standardize all features. For the single word subtask, we fit Linear, Lasso (Tibshirani, 1996), Elastic Net (Zou and Hastie, 2005), Support Vector Machine (Platt et al., 1999), 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.

4.2 Approach based on Feature Learning

Our handcrafted feature set relies heavily on target word-specific features. Beyond N-gram and syntactic features, it is a cursory analysis of the context surrounding the target word. We seek an alternative, automated approach using feature learning.

4.2.1 Architecture

LSTM-based approaches have been used to model the 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 as they can be interpreted as learned relationships between words. BERT (Devlin et al., 2018) is one such 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. We are able to initialize MT-DNN's shared text encoding layers with a pre-trained BERT base model (cased), and fine-tune its later layers for 5 epochs, using a mean squared error loss function and default hyperparameters. Such hyperparameter settings are provided in Appendix B. Note that the model is fine-tuned on only the CompLex corpus.

4.2.2 Input Layer

Data is fed to the model's input layer in *Premise-AndOneHypothesis* format, premise and hypothesis being sentence and target word/MWE, respectively. The data is preprocessed by a BERT tokenizer, backed by Hugging Face (Wolf et al., 2020).

4.2.3 Output Layer

Our model's output layer produces the predicted lexical complexity for a given target word/MWE. Additionally, we extract *attention maps* across each of the model's attention heads, for each test set sample. These will be assessed in Section 6.3.

4.3 Ensembling

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

5 Results

We present the performances of `BigGreen`'s system on each subtask in Tables 2 and 3.

Model	Pearson	Rank
XGBoost$_{full}$	0.7589	-
XGBoost$_{reduced}$	0.7456	-
XGBoost$_{full+reduced}$	0.7576	-
MT-DNN	0.7484	-
Ensemble (submission)	0.7749	8/54
Best competition results	0.7886	

Table 2: Test set results for single word subtask.

Model	Pearson	Rank
XGBoost$_{full+red.}$(head)	0.7164	-
XGBoost$_{full+red.}$(tail)	0.7188	-
MT-DNN	0.7890	-
Ensemble (submission)	0.7898	25/37
Ensemble (improved)	0.8290	*14/37
Best competition results	0.8612	

Table 3: MWE subtask test set results. (*projection)

6 Analysis

6.1 Performance

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 wrapper/embedded methods and a variety of regressors for model comparison. We find an XGBoost regressor (with hyperparameters tuned via grid search) to excel consistently for the single word subtask. As shown in Table 2, we rank in the top 15% by Pearson correlation.

For the MWE subtask, performances are reported in Table 3. Note that our submitted predictions differ from post-competition predictions. We *previously* used a training procedure resembling that for the single word subtask: (1) filter methods for feature selection, (2) XGBoost for regression, (3) ensembling 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 this subtask contributed to the prior 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 head and tail words. This gives us predicted complexities for the head and tail words each, 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.

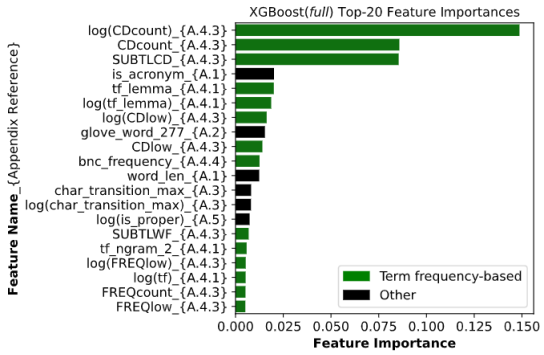

Figure 1: Feature importances for XGBoost$_{full}$. Definitions of the features are shown in Appendix A.

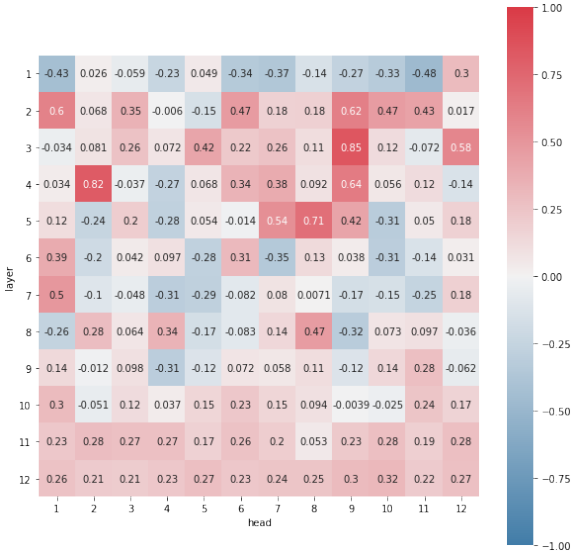

Figure 2: Attention head correlation between word frequency and total attention received by word, averaged across 100 random test set samples.

6.2 Feature Contribution

In total we consider 110 features, in addition to our 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 1). This raises concern for whether the MT-DNN model too relies on term frequencies to make *its* predictions, and if not, the linguistic features it may have learned upon fine-tuning. Of the remaining features having non-zero feature importances, most appear to be dimensions of target word-based semantic features (ie. GloVe or ELMo embeddings).

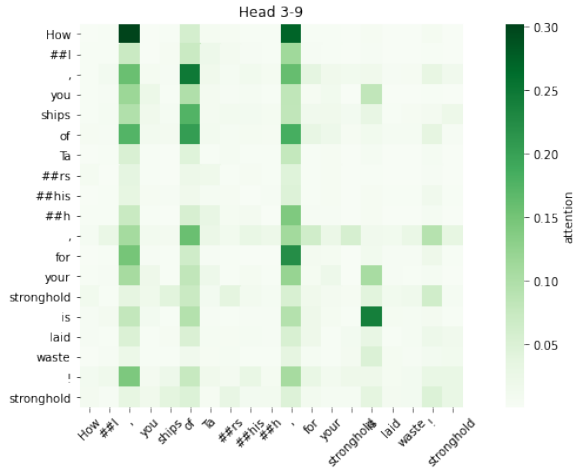

Figure 3: Head 3-9 attention map for a random sample.

6.3 BERT Attention

Attention maps have in previous works been assessed to demonstrate linguistic phenomena learned by a Transformer's specialized attention heads (Voita et al., 2019; Clark et al., 2019). We extract attention maps from MT-DNN's underlying fine-tuned BERT architecture. For each sample in the single word test set, we obtain an attention map from each of the BERT base model's 144 attention heads (ie. 12 heads per 12 layers).

Based on the precedence given to term frequency features by the XGBoost$_{full}$ model, we hypothesize that for certain attention heads, the degree to which BPEs attend to one another varies relative to their word's rarity in lexicon. This follows the findings of Voita et al. (2019), who identify heads in which lesser frequent tokens are attended to semi-uniformly by a majority of sentence tokens.

To test our hypothesis, we estimate for each attention head the Pearson correlation between word frequency and average attention given to each word in the context.[4] As illustrated in Figure 2, we find multiple attention heads appearing to specialize at directing attention towards the most or least frequent words (depending on sign of the correlation). Vertical stripe patterns like that in Figure 3 emerge as a result of attention originating from a spectrum of tokens. The findings seem to affirm the fundamental relevancy of word frequency to lexical complexity prediction, corroborating our intuition.

[4]We compute attention given to a *word* as the sum of attention given to its constituent BPEs. We use the GBND corpus to extract word frequencies, though any large corpora would suffice.

7 Conclusion

In this paper, we report inspirations for a system submitted by `BigGreen` to LCP SharedTask 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 complementary 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.

Avenues for improvement include better data aggregation, as a relative lack of class 4,5 samples seems to hurt Pearson correlation across extremely complex samples. An approach may involve synthetic data generation using SMOGN (Branco et al., 2017). Shardlow et al. (2020) acknowledge a reader's familiarity with a genre may affect perceived word complexity. However, the CompLex dataset lacks information on each annotator's expertise or background, which may offer valuable new insights.

References

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*.

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.

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.

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.

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)*.

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*.

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.

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.

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.

A.1 Lexical Features

`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.

A.2 Semantic Features

`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, pre-trained on Wikipedia-2014 and Gigaword. Target word is lowercased for ease.

`elmo_word`

- 1024-dimension embedding for each target word, pre-trained on the One Billion Word Benchmark corpus.

`glove_context`

- 300-dimension average of GloVe word embeddings (see `glove_word` above) for each word in the given context. Each word is lowercased for simplicity.

`infersent_embeddings`

- 4096-dimension embedding for the context.

A.3 Phonetic Features

`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.[5]

`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.

A.4 Word Frequency & N-gram Features

A.4.1 Gigaword-based

`tf`

- Target word term frequency. Note that all term frequency-based features are computed using 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.

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

`tf_summed_bpe`

- Sum of term frequencies of each BPE in 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

`google_ngram_1`

- Term frequency of the target word.

`google_ngram_2_head`

- Term frequency of leading bigram in the context containing the target word.

`google_ngram_2_tail`

- Term frequency of trailing bigram in the context containing the target word.

`google_ngram_2_min`

- Minimum of the set of term frequencies of bigrams 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 the context containing the target word.

`google_ngram_3_mid`

- Term frequency of middle trigram in the context containing the target word.

`google_ngram_3_tail`

- Term frequency of trailing trigram in the context containing the target word.

`google_ngram_3_min`

- Minimum of set of term frequencies of trigrams in the context containing 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

`FREQcount`

- Number of times target word appears in corpus.

`CDcount`

- Number of films in which 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`

- Percent of films in which target word appears.

A.4.4 BNC-based

`bnc_frequency`: Target word term frequency.

A.5 Syntactic Features

`parse_tree_depth`

- Height of context's constituency parse tree. Parse trees are obtained using a 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 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 indicating the target word's part-of-speech tag. Tags considered are those used in the Penn Treebank Project.[6] Tags are estimated using NLTK's `pos_tag` method.

A.6 Readability Features

`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 Textstat library implementations, most being readability metrics.

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

A.7 Other Features

`ppl`

- Perplexity metric, as defined by the Hugging Face library.[7] 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.

B Model Hyperparameters

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

B.1 XGBoost

Below are tuned parameters used for all of our XGBoost models. Parameters not listed are given default values as specified in documentation:[8]

`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

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

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

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

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.

Weighted average ensemble (single word subtask):
- Weight for XGBoost$_{full+reduced}$ prediction: 0.5
- Weight for MT-DNN prediction: 0.5

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