IITK@LCP at SemEval-2021 Task 1: Classification for Lexical Complexity Regression Task

Neil Shirude* **Sagnik Mukherjee***
Tushar Shandhilya **Ananta Mukherjee** **Ashutosh Modi**
Indian Institute of Technology Kanpur (IIT Kanpur)
{neilrs, sagnikm, anantam, stushar}@iitk.ac.in
ashutoshm@cse.iitk.ac.in

Abstract

This paper describes our contribution to SemEval 2021 Task 1: Lexical Complexity Prediction. In our approach, we leverage the ELECTRA model and attempt to mirror the data annotation scheme. Although the task is a regression task, we show that we can treat it as an aggregation of several classification and regression models. This somewhat counter-intuitive approach achieved an MAE score of 0.0654 for Sub-Task 1 and MAE of 0.0811 on Sub-Task 2. Additionally, we used the concept of weak supervision signals from Gloss-BERT in our work, and it significantly improved the MAE score in Sub-Task 1.

1 Introduction

With the rapid growth in digital pedagogy, English has become an extremely popular language. Although English is considered an easy language to learn and grasp, a person's choice of words often affects texts' readability. The use of difficult words can potentially lead to a communication gap, thus hampering language efficiency. Keeping these issues in mind, many Natural Language Processing tasks for text simplification have been recently proposed (Paetzold and Specia, 2017; Sikka and Mago, 2020). Our task of lexical complexity prediction is an important step in the process of simplifying texts.

The SemEval 2021 Task 1 (Shardlow et al., 2021) focuses on lexical complexity prediction in English. Given a sentence and a token from it, we have to predict the complexity score of the token. The task has two Sub-Tasks-

Sub-Task 1: complexity prediction of single words
Sub-Task 2: complexity prediction of multi word expressions (MWEs).

A word might seem complex because of 2 major factors-

a) The word is less common or complex in itself.
b) The context in which the word is used makes it hard to comprehend.

Observing the orthogonality of these two reasons, we captured the context-dependent features and independent features separately, trained models on them individually, and then combined the two using ensemble methods. We used the ELECTRA (Clark et al., 2020) model for extracting context-dependent features and GloVe embeddings (Pennington et al., 2014) for representing the word-level features.

Additionally, we propose a classification pipeline that is trained on GloVe embeddings of the tokens. This pipeline can be interpreted as a model for capturing different annotators' thought processes: overconfidence, under-confidence and randomness. We are making our code available for our models and experiments via GitHub[1].

2 Background

This task uses the CompLex dataset (Shardlow et al., 2020), which is a lexical complexity prediction dataset in English for single and multi word expressions (2-grams). Sentences in this task consists of sentences taken from 3 corpora- Bible, Biomed and Europarl. The train, validation and test split of the data was 9179, 520, 1103 respectively. We used the trial data as the validation set.

The aim of the task is to predict how complex a given token in a given sentence is. More mathematically, given a tuple $[s, t, c]$, where $s = [t_1, t_2, ...t_n]$ and $t = t_j$, we have to give an estimate of the function σ, such that $\sigma(s, t) = c$. (s is the sentence, t is the token and c is the complexity score).

The earlier focus on this task has been through

* Authors equally contributed to this work.

[1] https://github.com/neilrs123/Lexical-Complexity-Prediction

Proceedings of the 15th International Workshop on Semantic Evaluation (SemEval-2021), pages 541–547
Bangkok, Thailand (online), August 5–6, 2021.

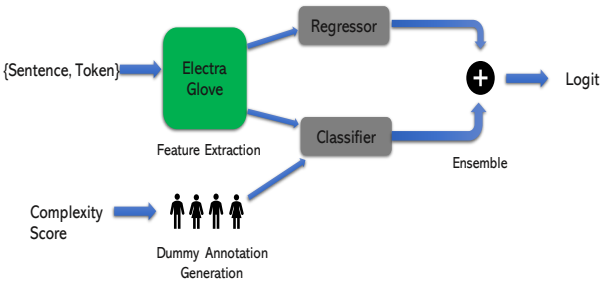

Figure 1: Solution Pipeline

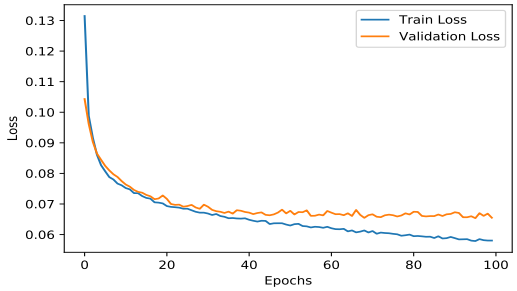

Figure 2: Convergence of losses for finetuning ELECTRA with weak supervision

the SemEval 2016 Task 11 (Paetzold and Specia (2016a)). However, it was a binary classification task. Most of the participating systems used Support Vector Machines such as Kuru (2016) and Choubey and Pateria (2016), decision trees and random forests (Choubey and Pateria (2016), Brooke et al. (2016), Ronzano et al. (2016)), and even basic threshold based approaches (Kauchak (2016), Malmasi et al. (2016)). Very few of them, including Bingel et al. (2016) used neural networks. The system by Wróbel (2016) achieved an F1 score very close to the winning solution using only single feature - word frequency from Wikipedia. Most of these systems use word embeddings, POS information and word frequencies as features. The winning system by Paetzold and Specia (2016b) however uses 69 morphological, semantic and syntactic features.

Another related shared task was presented at the BEA workshop at 2018 (Yimam et al., 2018). It had a probabilistic task as well as a binary classification task. Even there, the organizers conclude that feature engineering has worked better than neural networks. The winning system by Gooding and Kochmar (2018) uses feature engineering and later random forest and linear regression models.

3 System Overview

Our proposed pipeline can be divided into the following 4 main components-

a) Feature Extraction

b) Regression Pipeline

c) Classification Pipeline

d) Ensemble

The pipeline is shown in Figure 3.

3.1 Feature Extraction

ELECTRA is a transformer based model, that is trained like a discriminator and not like generator. And in our case, this model performed exceptionally well on the validation data as compared to BERT (Devlin et al., 2019).

We extracted context-dependent features using embeddings generated from the ELECTRA model and captured context-independent word-level features using static 200-dimensional GloVe embeddings of the tokens.

In order to generate the embeddings of the target word through ELECTRA, we implemented the KMP pattern matching algorithm (Wikipedia, 2021) to find the indices of the sub-tokens of the target token in the tokenized sentence. Subsequently, we calculated an average across these sub-token embeddings generated by ELECTRA.

While using GloVe embeddings, in the case of multi-word expressions in Sub-Task 2, the average of the embeddings of both token words was taken as the feature vector. If a word was not present in the GloVe dictionary, the GloVe embedding was initialized to a 200-dimensional vector consisting of zeros.

3.2 Regression Pipeline

The most natural way to look at the lexical complexity prediction task is to treat it as a regression task. The regression pipeline, a significant component of our system, is based on this idea. For Sub-task 1, in the regression pipeline, a pretrained ELECTRA model was finetuned with a linear layer on top of it. We leveraged the model directly available at the Huggingface library (Wolf et al., 2020). Only the last transformer layer of ELECTRA was kept trainable. The remaining ones were kept frozen. For Sub-task 2, a fixed ELECTRA model (non-trainable weights) was used to generate token embeddings and a linear regression model was trained with these extracted embeddings.

Weak Supervision: In order to have higher attention on the target word, the use of weak supervi-

sion signals proved useful. Inspired by GlossBert (Huang et al., 2019), the target word was wrapped with single inverted commas (' 's) as a weak signal to the transformer (Vaswani et al., 2017) model. This technique significantly improved the results obtained using the regression pipeline in subtask I. However, the same technique applied to subtask II made the scores worse.

Method	Val MAE	Test MAE
+ signal	0.06516	0.06800
- signal	0.06990	0.07118

Table 1: Variation of MAE scores with and without the signalling technique for Sub-task 1: the single word task. ('+ signal' means weak supervision has been used and '- signal' means otherwise.)

3.3 Classification Pipeline

Motivation from Annotation Procedure: Another way to look at the task is via a novel classification pipeline that is inspired from the data annotation process that is explained in Shardlow et al. (2020). Even though the task is a regression task, each data annotator performed a 5 class classification-

Given a sentence and a token in the sentence, each annotator had to select one class from among Very Easy, Easy, Neutral, Difficult and Very Difficult. Each of these classes was mapped to a discrete label between 0 and 1- namely 0, 0.25, 0.5, 0.75 and 1 respectively. The final complexity score was an average of up to 20 such annotations.

The Classification Pipeline aims to model this data annotation procedure. The main idea of this process is to teach classification models how to annotate data tuples. The three main components of this scheme are-

a) Generating dummy annotations from complexity scores **b)** Training classification models on dummy annotations, and **c)** Aggregating all predicted annotations to generate predicted complexity scores.

Generation of Dummy Annotations: A given complexity score can be represented as a weighted average of its lower and upper target classes and the weights can be determined using the magnitude of the complexity score. These weights then determine the proportions of the two classes in the set of dummy annotations for that data tuple. For example, if the number of dummy annotators is $n = 5$ and the complexity score of the training example is $c = 0.2$, the lower and upper target classes are $low = 0$ and $high = 0.25$, respectively. Let α be the proportion of dummy annotations with the lower target class. Correspondingly, $1 - \alpha$ will be the proportion with the upper target class. The number of dummy annotations with $target\_class = low$ are given as $floor(n * \alpha)$ and that with $target\_class = high$ as $n - floor(n * \alpha)$. α can be calculated using the equation-

$$c = \alpha * low + (1 - \alpha) * high$$

We get $\alpha = 0.2$. Hence, we have $floor(n * \alpha) = 1$ dummy annotations with $target\_class = low(0)$ and remaining 4 annotations with $target\_class = high(0.25)$. Hence, the dummy annotations set for $c = 0.2$ is $0, 0.25, 0.25, 0.25, 0.25$. Similarly, the dummy annotations set for $c = 0.8$ is $0.75, 0.75, 0.75, 0.75, 1$.

In this process, we also attempted to capture the impact of intentional human errors made during the data annotation procedure. Just like a weary or uninterested annotator who would have randomly selected for one of the five classes for a certain data tuple, a small fraction of the dummy annotations was assigned random values from the set containing 0, 0.25, 0.5, 0.75 and 1. This modification aims to model the small-scale randomness in annotation procedure.

Using this procedure, dummy annotation sets of size n can be generated for any value of c, where n can be treated as a hyperparameter. The value n can also be interpreted as the number of classification models that are being trained in the next step.

Classification Models: In a diverse set of annotators, there will be over-confident annotators who will select lower classes and there will be under-confident annotators who will select upper classes. Then there will be neutral annotators as well. By ensuring that the dummy annotations are sorted, we can say that the first classifier learns how to annotate like the over-confident annotator, the last classifier learns how to annotate like the under-confident annotator and the classifiers in between model the neutral annotators. We trained SVM classifiers with RBF kernels, using GloVe embeddings of token words as features.

Aggregation of Predicted Annotations: The annotations were aggregated by simply taking the average of all predicted class labels in order to obtain the final predicted complexity scores. Each of these models may have high individual variances,

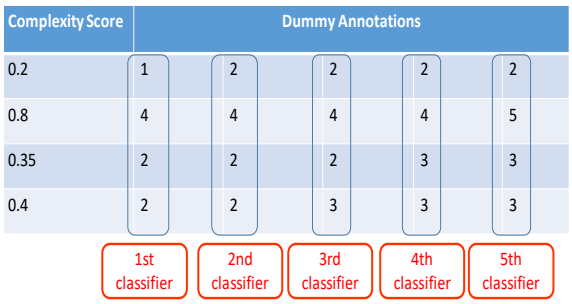

Figure 3: A few worked out examples of generating dummy annotations from complexity scores. For each of these cases, the continuous labels 0,0.25,0.50,0.75 and 1 are mapped to categorical labels 1,2,3,4,5 and then put into SVM. Clearly the labels of the 1st classifier is less that that of the second one. i.e. on a scale of confidence, the first classifier is at a lesser position. So it models a less confident person.

but the ensemble tends to have lower variance and bias. Also, any number of models can be inserted in the ensemble without leading to over-fitting on the train data.

3.4 Ensemble

In order to have a better bias variance trade off and also to exploit the "expertise" of different pipelines, the final approach incorporates both the regression and classification pipelines to form an ensemble. The final predicted complexity was obtained by taking an ensemble of the predictions from the regression and classification pipelines as described above. The classification pipeline for both the Sub-Tasks was based on GloVe embeddings as features and SVM classifiers. The regression pipeline for Sub-Task 1 was based on fine-tuning ELECTRA with weak supervision and that for Sub-Task 2 was based on features collected from ELECTRA model (non-trainable) with a linear regression trained on it.

4 Experimental Setup

The official evaluation metric for both the Sub-Tasks was Pearson Correlation (standard for regression tasks). For both sub-tasks, the train/test/val split as per the official release has been used. The ELECTRA finetuning was done with an NVIDIA GTX 1080 GPU with early stopping (93 epochs). We used the MAE loss function to train the model with an adam optimizer with $lr = 1e^{-}5$, $eps = 1e-08$ and $weightdecay = 0$. Training set was shuffled and the batch size was kept at 64. In the ELECTRA model, the padding parameter was set to True and maximum length was at 140. For the SVM models the value of slack was chosen to be 1 and for SVM and Linear regresion the sklearn (Pedregosa et al., 2011) library was used. All the hyperparameters were tuned with a grid search method.

5 Results

Results on Validation Data: The comparison of the baseline results and our results obtained using the regression pipeline, the classification pipeline and the ensemble of the two models on the validation set (trial data) is given in Table 4.

Task	Baseline	Regression Pipeline	Classification Pipeline
Subtask 1	0.0853	0.0651	0.0641
Subtask 2	-	0.0840	0.0768

Table 2: Results on validation set (Mean Absolute Errors)

Task	MAE	Pearson	MSE
One	0.0623	0.8308	0.0065
Two	0.0727	0.8146	0.0087

Table 3: Results on Vaditation Set for final ensemble

Results on Test Data: Our results on the test data along with the best results obtained for each task are shown in Table 1.
The winning system's pearson and MAE scores on the test data are as follows: 0.7886 and 0.0609 for subtask I(single word expressions), 0.8612 and 0.0616 for subtask II(multi word expressions).

Task	MAE	Pearson	MSE
One	0.0654	0.7511	0.0071
Two	0.0811	0.8277	0.0098

Table 4: Results on Test Set

6 Error Analysis

Analyzing all the experiments and the corresponding results, the following can be concluded: **a)** Word-level features as well as context-dependent features need to be considered while determining

complexity of a token. **b)** Approaches based on the data annotation scheme are well suited to tackle the lexical complexity prediction task. **c)** Ensemble of a large number of simple models is an effective way of tackling this task. **d)** Models with large number of parameters like BERT () suffer heavily due to overfitting, where as ELECTRA base prove to be much better.

The model architectures that were tried out in earlier stages showed similar trends. For example, ELECTRA finetuning produced much better scores than BERT finetuning. Also, simpler models like a simple linear regression on GloVe embeddings showed promise, proving that simpler models with lesser parameters worked better. All these trends across those models are visually shown in Figure 4. It was observed that the model was underperforming on the tuples from Biomed corpus. However the scores did not improve using BERT variants like BioBERT (Lee et al., 2019), BioMedBERT (Chakraborty et al., 2020) and a few other transformer based models pretrained on biomedical texts. A variant of ELECTRA on biomedical texts could have improve on this, however due to its unavailability it could not be tried out.

In majority of the prior work on LCP, there is abundance use of word frequency as a feature. However, in this system the scores got worse when frequency features were used along with others in ensemble. And the feature in itself could not produce competitive results. Previously, Gong et al. (2020) and Mu et al. (2018) have shown that frequency information causes significant distortion in the embedding space. We also hypothesize that the frequency information in GloVe embeddings help us in this regard.

7 Conclusion

In this paper we presented a system for lexical complexity prediction in the form of a regression task. The proposed system's primary novelty is in treating it as a classification task and trying to model the annotation scheme. An ensemble of these classification models and vanilla fine-tuning of ELECTRA model proved to be very useful. Also the weak supervision based approach gave the scores a significant boost for the Sub-Task 1.

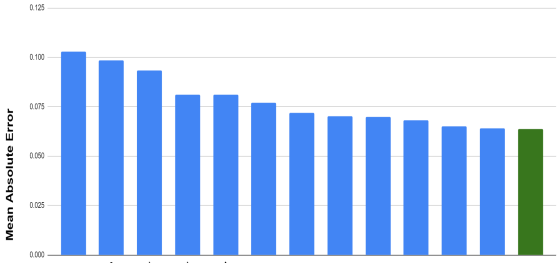

Figure 4: Comparison of MAE values of the models we tried (Subtask I). From left the models are (1) Linear Regression with hand crafted features, (2) Character level RNN, (3) Character level CNN, (4) and (5) sentence and character level GRU and LSTMs (6), (7), (8), (9) and (10) Linear Regression with GloVe, ELECTRA and BERT embeddings, (11) the current regression pipeline, (12) classification pipeline, (13) Final ensemble

References

Joachim Bingel, Natalie Schluter, and Héctor Martínez Alonso. 2016. CoastalCPH at SemEval-2016 task 11: The importance of designing your neural networks right. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 1028–1033, San Diego, California. Association for Computational Linguistics.

Julian Brooke, Alexandra Uitdenbogerd, and Timothy Baldwin. 2016. Melbourne at SemEval 2016 task 11: Classifying type-level word complexity using random forests with corpus and word list features. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 975–981, San Diego, California. Association for Computational Linguistics.

Souradip Chakraborty, Ekaba Bisong, Shweta Bhatt, Thomas Wagner, Riley Elliott, and Francesco Mosconi. 2020. BioMedBERT: A pre-trained biomedical language model for QA and IR. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 669–679, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Prafulla Choubey and Shubham Pateria. 2016. Garuda & Bhasha at SemEval-2016 task 11: Complex word identification using aggregated learning models. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 1006–1010, San Diego, California. Association for Computational Linguistics.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. Electra: Pre-training text encoders as discriminators rather than generators.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding.

Chengyue Gong, Di He, Xu Tan, Tao Qin, Liwei Wang, and Tie-Yan Liu. 2020. Frage: Frequency-agnostic word representation.

Sian Gooding and Ekaterina Kochmar. 2018. CAMB at CWI shared task 2018: Complex word identification with ensemble-based voting. In *Proceedings of the Thirteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 184–194, New Orleans, Louisiana. Association for Computational Linguistics.

Luyao Huang, Chi Sun, Xipeng Qiu, and Xuanjing Huang. 2019. GlossBERT: BERT for word sense disambiguation with gloss knowledge. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3509–3514, Hong Kong, China. Association for Computational Linguistics.

David Kauchak. 2016. Pomona at SemEval-2016 task 11: Predicting word complexity based on corpus frequency. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 1047–1051, San Diego, California. Association for Computational Linguistics.

Onur Kuru. 2016. AI-KU at SemEval-2016 task 11: Word embeddings and substring features for complex word identification. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 1042–1046, San Diego, California. Association for Computational Linguistics.

Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. 2019. Biobert: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*.

Shervin Malmasi, Mark Dras, and Marcos Zampieri. 2016. LTG at SemEval-2016 task 11: Complex word identification with classifier ensembles. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 996–1000, San Diego, California. Association for Computational Linguistics.

Jiaqi Mu, Suma Bhat, and Pramod Viswanath. 2018. All-but-the-top: Simple and effective postprocessing for word representations.

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, San Diego, California. Association for Computational Linguistics.

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, San Diego, California. Association for Computational Linguistics.

Gustavo H. Paetzold and Lucia Specia. 2017. A survey on lexical simplification. *J. Artif. Int. Res.*, 60(1):549–593.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. 2011. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12(85):2825–2830.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Francesco Ronzano, Ahmed Abura'ed, Luis Espinosa-Anke, and Horacio Saggion. 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, San Diego, California. Association for Computational Linguistics.

Matthew Shardlow, Michael Cooper, and Marcos Zampieri. 2020. CompLex — a new corpus for lexical complexity prediction from Likert Scale data. In *Proceedings of the 1st Workshop on Tools and Resources to Empower People with REAding DIfficulties (READI)*, pages 57–62, Marseille, France. European Language Resources Association.

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)*.

Punardeep Sikka and Vijay Mago. 2020. A survey on text simplification.

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

Wikipedia. 2021. Knuth–morris–pratt algorithm — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Knuth%E2%80%93Morris%E2%80%93Pratt_algorithm&oldid=1005503556. [Online; accessed 17-February-2021].

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen,

Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Huggingface's transformers: State-of-the-art natural language processing.

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, San Diego, California. Association for Computational Linguistics.

Seid Muhie Yimam, Chris Biemann, Shervin Malmasi, Gustavo Paetzold, Lucia Specia, Sanja Štajner, Anaïs Tack, and Marcos Zampieri. 2018. A report on the complex word identification shared task 2018. In *Proceedings of the Thirteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 66–78, New Orleans, Louisiana. Association for Computational Linguistics.