JCT at SemEval-2021 Task 1: Context-aware Representation for Lexical Complexity Prediction

Chaya Liebeskind, Otniel Elkayam and Shmuel Liebeskind
Department of Computer Science, Jerusalem College of Technology
21 Havaad Haleumi St., P.O.B. 16031
9116001 Jerusalem, Israel
(liebchaya, otniel.elkayam, israellieb)@gmail.com

Abstract

In this paper, we present our contribution in SemEval-2021 Task 1: Lexical Complexity Prediction, where we integrate linguistic, statistical, and semantic properties of the target word and its context as features within a Machine Learning (ML) framework for predicting lexical complexity. In particular, we use BERT contextualized word embeddings to represent the semantic meaning of the target word and its context. We participated in the sub-task of predicting the complexity score of single words.

1 Introduction

Over the last decade, automated methods for detecting complex words have been developed. At the beginning, most of these methods assumed that lexical complexity is binary, words are either ”difficult” or ”not difficult”. Thus, the first Complex Word Identification (CWI) shared task referred to binary identification of complex words (Zampieri et al., 2017). The main limitation of this assumption is that a word close to the decision boundary is considered to be as complex as one farther apart. Therefore, three years ago, the CWI included an additional probabilistic classification task where the participants were asked to give a probability of the given target word in particular context being complex (Štajner et al., 2018).

Recently, CompLex, a new English corpus for lexical complexity prediction was introduced (Shardlow et al., 2020). The corpus is annotated using a 5-point Likert scale (1-5) (corresponding to very easy, easy, neutral, difficult, and very difficult), and covers 3 genres: Bible translation, European Pariliament proceedings, and biomedical articles. SemEval-2021 (Task 1) shared task on Lexical Complexity Prediction (LCP) (Shardlow et al., 2021a,b) provided the participants with Complex and defined two sub-tasks: predicting the complexity score of single words, and predicting the complexity score of multi-word expressions.

We present our system for the first sub-task of predicting the complexity score of single words. Our system incorporates linguistic, statistical, and semantic properties of the target word and its context as features within a Machine Learning (ML) framework for predicting lexical complexity.

This paper is organized as follows: First, in Section 2, we describe features from previous works that we have adopted. Then, in Section 3, we describe our feature sets, the feature selection process, and the results on the trial data. Finally, Our system results on the test data are detailed in Section 4, followed by conclusions in Section 5.

2 Related work

In this section, we shortly describe linguistic, statistical, and semantic features which were encoded as features in previous complexity prediction tasks and were integrated in our system.

Linguistics features, such as Part-Of-Speech (POS) tag, dependency parsing relations, and syllable counts, as well as statistical features, such as word length and word frequency, have been widely used for predicting lexical complexity (Mukherjee et al., 2016; Ronzano et al., 2016; Alfter and Pilán, 2018; Gooding and Kochmar, 2018; Hartmann and Dos Santos, 2018; Kajiwara and Komachi, 2018; Wani et al., 2018). Some of these works found WordNet (Miller, 1998) as a valuable source of lexical features. The main extracted feature is the number of synsets, but also information on hypernyms, hyponyms, holonym, and meronym is useful (Gooding and Kochmar, 2018; Hartmann and Dos Santos, 2018; Wani et al., 2018).

Semantic features were commonly encoded using word embedding representation of the meaning of words (Kuru, 2016; AbuRa'ed and Saggion, 2018). These word embeddings were generated using Word2Vec context-independent models (Mikolov et al., 2013). Word2Vec models combine different senses of the word into one single vector. However, recently, there is a growing interest in contextualized word representations, such as BERT (Devlin et al., 2018). BERT model generates context-dependent embeddings that allow a word to have several vector representations depending on the context in which it is used. In contrast to previous works that only use context-independent embeddings, our system uses the BERT-based context-dependent embeddings.

3 System Description

We adopt a supervised Machine Learning (ML) approach for lexical complexity prediction. The first step in a classifier training is to determine which text characteristics are relevant and how those features are coded.

3.1 Feature Sets

We next detail how the semantic properties of the sentence, as well as the linguistic and statistical properties found useful in prior work, are encoded as features. Then, in Section 3.2, we describe our feature analysis procedure and the supervised ML model. The features in our model are divided into 3 sets: linguistic, statistical and semantic.

3.1.1 Linguistic features

Our dataset contains three corpora: Bible, Europarl, and Biomedical, to add variation. Since each corpus has its own unique linguistic features, we first encode the text source by three binary features.

Most of our linguistic features are based on information extracted from a POS tagger. Our linguistic properties include two families of properties: morphological and syntactical.

First, we encode the target word POS. The POS is extracted by the Spacy's statistical POS tagger[1]. Each possible POS tag is represented as a binary feature. We use the following 12 tags from the Universal POS tags[2]: ADJ, ADP, ADV, CONJ, DET, NOUN, NUM, PRT, PRON, VERB and X (other). As an additional feature, the number of syllables in the target word is encoded[3]. Then, we calculate the number of punctuation marks and stopwords in the sentence (two features).

Next, we represent syntactic forms by POS patterns. The POS pattern refers to seven words, the target word and three words before and after it. Each of the words is encoded by 12 binary features, resulting with 84 features.

We also measure the polysemy degree of the target word using the number of senses in WordNet. We obtain two lexical features: number of synsets for the target word and number of synsets for the target word given its POS.

3.1.2 Statistical features

We define some statistical features based on frequency. First, we calculate target word length and sentence length. Then, we extract the target word frequency using Google N-gram[4] word frequencies. We encode the logarithm of this frequency as a feature to speed the ML algorithm's convergence (three features).

3.1.3 Semantic features

We represent the meaning of the surrounding context of the target word by vectors in the same semantic space. We use the BERT semantic space. BERT is a bidirectional transformer pre-trained on a large corpus containing the Toronto Book Corpus and Wikipedia using a combination of masked language modeling objective and next sentence prediction. BERT contextualizing vectors are used to represent the semantic meaning of the sentence by averaging the BERT vectors of seven words, the target word and three words before and after it. Thus, our semantic representation add 768 features (the size of BERT output layer).

To extract additional features, we use two machine learning algorithm: K-Means and k-Nearest Neighbors (KNN) algorithm. K-Means is an unsupervised learning algorithm used for clustering. It takes the unlabeled dataset and tries to group them into k number of clusters. We encode the K-Mean results by four binary features, a feature per cluster (k=4). The results of the KNN algorithm are encoded similarly. However, KNN is a supervised learning algorithm used for classification. It takes the labeled dataset and uses it to learn how to label other sentences. KNN classifies an unseen sentence using it k nearest neighbors voting. We use four complexity classes: 0-0.25, 0.26-0.5, 0.51-0.75, 0.76-1.

[1] https://spacy.io/

[2] https://universaldependencies.org/u/pos/index.html

[3] https://eayd.in/?p=232

[4] https://books.google.com/ngrams

3.2 Feature Selection

For each of the above feature sets, we tried to filter out non-relevant features using several approaches.

First, we discharged features that decrease the system performance on the training set, namely, the POS pattern features, the WordNet features, and the K-Means and KNN features. We were left with 794 features. These features were selected using the Linear Regression algorithm, which was also selected as a baseline algorithm by the task organizers. To further improve the performance of our systems, we used additional ML algorithms, such as SVM and XGBoost (see more details in Section 3.3).

Next, since correlated features do not carry unique information and may interfere the learning, we tried to discharge highly correlated features. We implemented this approach using the following iterative process. The input is the desired final number of features. First, we define an initial correlation threshold (0.9). Then, we calculate the features' pairwise correlation and features with correlation above the threshold are removed. Next, if we still have more features than desired, we will lower the correlation threshold (by 10%) and repeat the process. This approach improved the performance of the SVM and Linear Regression models (selecting 97 features), but did not increase the performance of the XGBOOST method.

We note that we also tried to filter out feature using the principal component analysis (PCA) feature selection method (Song et al., 2010). PCA aims to pick a subset of features that retains as much information present in the full data as possible. PCA was performed both on the full feature list and on specific features, such as BERT features, but it was not successful.

Some of the classification models had low performance using such amount of features (794 features). Therefore, we further filleted features by calculating their correlation with the complexity score and discarding features with low correlation (less than 0.072). We resulted with the following list of 101 features:

- Biomedical corpus indicator
- Europal corpus indicator
- NOUN POS tag
- PRON POS tag
- number of syllables in the target word
- target word length
- target word frequency (Google N-gram)
- 94 features from BERT vector

It is interesting to note that even though, there are 12 POS tags, only 2 are informative for the complexity prediction task. Considering the source text indicators, the third Bible indicator is not useful. Out of the BERT 768 features, only 94 remained (12.2% of the vector).

The BERT representation of the sentence is generated by pre-trained language representation model. These models can be trained on different datasets of various domains. Since one of our corpora is from the Biomedical domain, we examined the system performance using the domain specific BioBERT (Lee et al., 2020). Figure 1 shows a comparison between the error rate of our system using the classic BERT and BioBERT (BERT on the left and BioBERT on the right). The columns show the error rate for different text sources. The red line is the average error rate. Columns from left to right: Bible, Biomedical, and Europarl. Surprisingly, the error rate of the BioBERT on the Biomedical domain is higher than that of the classic BERT. However, the average error for both is the same (~ 0.69).

3.3 Application of five Machine Learning methods

We combined the features in a supervised classification framework using five ML methods: Linear Regression, Supported Vector Machine (SVM), XGBoost (XGB), KNN, and Stacking (Stack). We trained the ML methods on the train set and evaluated their performances on the trial set.

We ran these ML methods by the scikit-learn open-source machine-learning package in python[5] (Pedregosa et al., 2011) using the default parameters. Table 1 shows the performances of the different ML methods on the feature set of 101 features, as described above. The MAE is omitted from the table because it is similar for all the ML algorithms (0.01). The performance differences between the algorithms were not so substantial. Therefore, we next report the performances of all these methods on the test set.

[5]https://scikit-learn.org/stable/

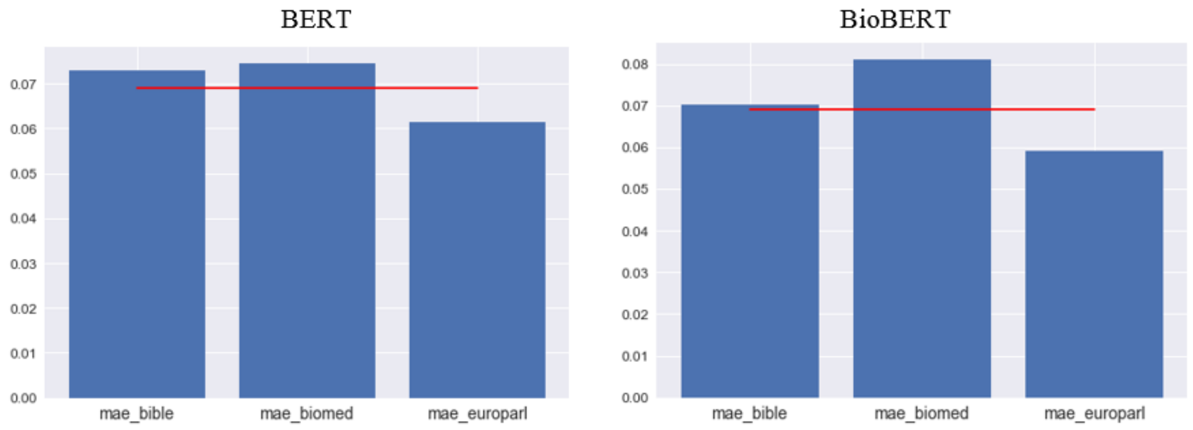

Figure 1: A comparison between the error rate of our system using the classic BERT and BioBERT

Alg.	Pearson	Spearman	MSE	R2
LR	0.672	0.656	0.077	0.45
SVM	0.693	0.67	0.075	0.476
XGB	0.671	0.655	0.076	0.449
KNN	0.682	0.64	0.077	0.464
Stack	0.689	0.66	0.077	0.436

Table 1: The performances of the different ML algorithms on the trial set

4 Results

To increase the size of our train set for the test phase of the task, we used both the train and trial sets to train the final model. Table 2 presents our results on the test set. The predictions of the XGBoost were submitted to the shared task competition. The results of the different algorithms are close to each other and consistent with the results on the trial set. The results of the KNN method are a bit lower. Even though, stacking allows to use the strength of each individual classifier by using their output as input of a final classifier, it did not obtain better result. This may imply that the different classifiers exploit the same information and do not reveal supplementary information.

Alg.	Pear.	Spea.	MAE	MSE	R2
LR	0.629	0.622	0.079	0.01	0.384
SVM	0.669	0.645	0.074	0.009	0.439
XGB	0.666	0.646	0.074	0.009	0.44
KNN	0.618	0.598	0.074	0.01	0.358
Stack	0.658	0.633	0.079	0.009	0.433

Table 2: The performances of the different ML algorithms on the test set

To analyze our results, we converted the complexity scores to labels following Shardlow et al. (2020) descriptors. In Figure 2, we present the classification confusion matrix of the XGBoost algorithm. Each column of the matrix represents the instances in a predicted class while each row represents the instances in an actual class. Most of the classification errors (18.54%) were due to incorrect classification of very easy words as easy. There were also errors in the opposite direction (4.36%). Most of the rest of the classifications were between neutral and easy in both directions $(7.42\% + 6.43\% = 13.85\%)$. We note that the 5th class, very difficult, does not appear in the confusion matrix since there are not any very difficult words in the test set and the system did not classified any of the words as very difficult.

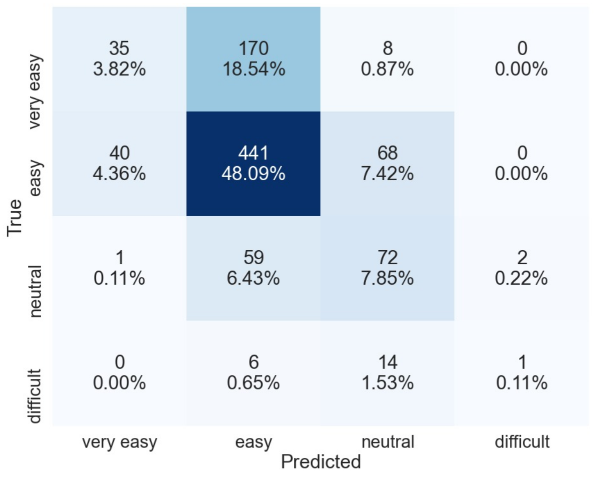

Figure 2: A confusion matrix for the XGBoost complexity predictions

5 Conclusions and Future Work

We have implemented a system that incorporates linguistic, statistical, and semantic features to predict lexical complexity of target word in context. BERT semantic space was used to represent the word and its context. We investigated several feature selection approaches and used various supervised algorithms.

Even though our system was not highly ranked, we believe that some of the presented ideas can be useful for future research on lexical complexity prediction. In particular, we think that BERT is a powerful model that should be explored. Perhaps, fine-tuning BERT for the complexity prediction task would increase the system performance.

References

Ahmed AbuRa'ed and Horacio Saggion. 2018. LaS-TUS/TALN at the Complex Word Identification 2018 Shared Task. In *Proceedings of the Thirteenth Workshop on Innovative Use of NLP for Building Educational Applications; 2018 Jun 5; New Orleans, LA. Stroudsburg (PA): ACL; 2018. p. 159–65.* ACL (Association for Computational Linguistics).

David Alfter and Ildikó Pilán. 2018. SB@ GU at the Complex Word Identification 2018 Shared Task. In *Proceedings of the Thirteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 315–321.

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*.

Sian Gooding and Ekaterina Kochmar. 2018. CAMB at the Complex Word Identification 2018 Shared Task: 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.

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

Tomoyuki Kajiwara and Mamoru Komachi. 2018. Complex word identification based on frequency in a learner corpus. In *Proceedings of the thirteenth workshop on innovative use of NLP for building educational applications*, pages 195–199.

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.

Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. 2020. BioBERT: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 36(4):1234–1240.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. *arXiv preprint arXiv:1310.4546*.

George A Miller. 1998. *WordNet: An electronic lexical database*. MIT press.

Niloy Mukherjee, Braja Gopal Patra, Dipankar Das, and Sivaji Bandyopadhyay. 2016. Ju_nlp 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.

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. *Journal of machine learning research*, 12(Oct):2825–2830.

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, 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. 2021a. SemEval-2021 Task 1: Lexical Complexity Prediction. In *Proceedings of the 14th International Workshop on Semantic Evaluation (SemEval-2021)*.

Matthew Shardlow, Richard Evans, and Marcos Zampieri. 2021b. Predicting lexical complexity in english texts. *arXiv preprint arXiv:2102.08773*.

Fengxi Song, Zhongwei Guo, and Dayong Mei. 2010. Feature selection using principal component analysis. In *2010 international conference on system science, engineering design and manufacturing informatization*, volume 1, pages 27–30. IEEE.

Sanja Štajner, Chris Biemann, Shervin Malmasi, Gustavo Paetzold, Lucia Specia, Anaïs Tack, Seid Muhie Yimam, and Marcos Zampieri. 2018. A report on the Complex Word Identification Shared Task 2018.

In *Proceedings of the 13th Workshop on Innovative Use of NLP for Building Educational Applications*.

Nikhil Wani, Sandeep Mathias, Jayashree Aanand Gajjam, and Pushpak Bhattacharyya. 2018. The whole is greater than the sum of its parts: Towards the effectiveness of voting ensemble classifiers for complex word identification. In *Proceedings of the thirteenth workshop on innovative use of NLP for building educational applications*, pages 200–205.

Marcos Zampieri, Shervin Malmasi, Gustavo Paetzold, and Lucia Specia. 2017. Complex Word Identification: Challenges in data annotation and system performance. In *Proceedings of the 4th Workshop on Natural Language Processing Techniques for Educational Applications (NLPTEA 2017)*.