SemEval 2021 Task 1 - Lexical Complexity Prediction (LCP)

Abhishek Mukherjee (20CS60R03), Kirti Agarwal (20CS6014), Vipray Jain (20CS6037)
Rishabh Thakur (20CS6038), Mehul Vaidya (20CS6042)
Codalab ID: iitkgp_cs60075_team10
GitHub: https://github.com/RST1996/CS60075-Team-10-Task-1
Youtube: https://www.youtube.com/watch?v=dxfQe6vQbYI

Abstract

This paper presents the systems we submitted to the Complex Word Identification Shared Task 2021. We describe our systems' implementations and discuss our key findings from this research. Our systems achieve an Person score of 0.7604 on the single word task and 0.8285 on the multi word expression.

1 Introduction

Poor reading ability is often caused by the presence of complex words in paragraph. Complex Word Identification (CWI) is concerned with automated identification of words that might present challenge for the target readers and should thus be simplified.

In SemEval 2021 Lexical Complexity Prediction (LCP)(4) task is about predicting lexical complexity of highlighted single and multi-word in sentence. In this we have output real value between 0 to 1 as complexity score for single and multi-word.

We participated in LCP task and successfully built a system which predict complexity score between 0 to 1 for given word. Section 2 of this describes approach we followed for this task. Finding appropriate features is important part LCP as well as many other NLP task. Section 3 describes features that we used for our task. Section 4 provide details about out model. Section 5 and 6 discuss results and conclusion.

2 Contribution

The contribution of members is mentioned in fig.1.

3 Approach

We studied various approach followed by participant in SemEval 2016 and SemEval 2018 for LCP task. Then we decided upon various features that we thought will be helpful for our task. We used various static features. We also used semantics information provided by Glove Embedding. We also included various context information provided by BERT.
We got feature vector of size 2153 for single and 1385 for multi word prediction. We then trained six different types of models on this feature vector. We then stack these models together to get final prediction.

4 Features Used

We used two types of features for our task. Type 1 features are static features without any context specific information. Type 2 features captures the semantic and contextual information.

Static Features

1. Length of Target Word
2. Number of Vowels / Syllables
3. Synonym / Antonym / Hypernyms /Homonym (not in use, taken for experiment)
4. Part of Speech of Target Word
5. High/Low character frequency score
6. No of splits of target word by WordPiece tokenizer.
7. One-hot encoding of category of document from which sentence was taken
8. Position of token in sentence.

Context Features

1. Average of Glove Embedding of word in sentence.
2. Glove Embedding of token or target word

Roll No.	Name	Contribution
20CS60R03	Abhishek Mukherjee	BERT and Multi-layer Perceptron
20CS60R14	Kirti Agarwal	Lexical Features and Mean Stacking Model
20CS60R37	Vipray Jain	Data Cleaning and Glove Embedding
20CS60R38	Rishabh Thakur	POS Tags Feature and Ensembling(Bagging and Boosting Models)
20CS60R42	Mehul Vaidya	Character-to-Word Complexity feature and Stacking

Figure 1: Contribution of Team Members

3. Pooled output of each sentence from BERT uncased base model
4. Context vector of target word from sequence output of BERT uncased base model

Total size of our feature vector is 2153 for single and 1385 for multi word prediction.

5 Learning Model

Using above feature vectors we trained six different types of Model

1. Using Simple Neural Network(3) : We have used four dense layer of sigmoid activation and used drop out of 20% in each layer to avoid over-fitting. In total we had 286321 trainable parameters in neural network. We have used custom callback to save model which was giving highest Pearson score on validation data while training model
2. Adaboost using Random Forest Regressor: We are using Adaboost from Sklearn library(1) using N estimator of 80.
3. Adaboost using Neural Network as Base Estimator : We are using adaptive boosting method with neural network as it's base estimator.
4. Bagging regressor using random forest based estimator: We are using bagging regressor from Sklearn library using N estimator of 100
5. Bagging regressor using neural network as base estimator : We are using bagging method with neural network as it's base estimator.
6. Gradient boosting(2) regressor using random forest base estimator: We are using Gradient boosting regressor from Sklearn library(2) using N estimator of 100

We then stacked various combination of above models. Then we selected best combination of models based on Pearson and R2 score. For stacking we used mainly two techniques:

1. The predication values of above models are stacked and we then train a regressor mainly linear, Adaboost, gradient and bagging using stacked prediction as `X` values and actual prediction as `Y` values for those regressors. We then take regressor which performs best on the stacked data.
2. Second process we used is to take central tendencies like mean, median on the prediction of regressor models.

6 Procedure

For Data Cleaning and Prepossessing we performed following operations:

1. We removed punctuation from given text
2. Made sentence lowercase
3. Removed spaced from start and ending of sentence.
4. Removed stop words from sentence
5. We tokenized sentence using BERT tokenizer (WordPiece tokenizer)
6. We padded tokenized sentence to size of 512 so that it can fed to BERT uncased base model
7. We created masked and segment vector for tokens we prepared in previous steps.
8. Then we stacked all the features into a single feature vector and trained this feature vector over various models.

Lexical features Extracted: We computed the following lexical features for each sentence in data.

1. Part of Speech(POS) tag of target word: for each target word we calculated it's part of speech tag like whether it is determinant, noun, adjective, pronoun, adverb, verb etc.
2. We computed number of synonyms, antonyms, homonyms, hypernyms of target word in a sentence.
3. We also calculated low frequency and high frequency score of sentence using standard values(5).
4. We calculated number of vowels present in each sentence.
5. We also added length of target word as feature.

Word Vector and Context features Extraction:

1. we used glove embedding of 300 vector size of target word as feature.
2. We used average of glove vector embedding of 300 dimension of each word of sentence as feature.
3. We used pooled output of each sentence from BERT uncased base model of 768 dimension.
4. We used word embedding of target word from sequence output of from BERT uncased model of 768 dimension as feature

Total Feature vector creation (common for both sub-task):

1. Pooled output of sentence from BERT(6) model (768 dimension vector).
2. Glove Embedding of the target word. Average of all word in case of multi word target (300 dimension).
3. Average glove embedding of all words in sentence (300 dimension)
4. Other static features as mentioned above.
5. In addition to the above features, in case of the single word lexical complexity prediction we have also used the sequence output of the target word from BERT(6) model. In cases where the target word got split into multiple tokens we have taken the average of the tokens.
6. in addition to above features we also appended the static lexical features , we extracted for target word and the sentence as mentioned above.

We horizontally stacked all the features mentioned above to form final feature vector for training, validation and testing.
Models Architecture implemented and tested:

1. Machine Learning regressors models like linear regressor, Support vector regressor: We tried using models like linear regressor, Support vector regressor and decision tree regressor but results we got were not satisfactory hence it was apparent that we should use some word vector based models.
2. Neural Net based Models using only word embedding from glove: We saw significant increase in Pearson and R2 score when we used neural Net based models on word embedding from glove on the target word and also used the average of word embedding of all words in a sentence. Therefore it is quite clear that semantic information of target words captured in 300 dimension glove vectors has significant impact on prediction capability of Neural Net based model.
3. Neural Net based models using pooled output from BERT(6) and Glove embedding: On previous models we also added the pooled output of BERT(6) uncased model of 768 dimensions so that we can capture contextual information of a sentence. As expected we also saw bit increase in Pearson and R2 score on the validation data.
4. Models using the word embedding of target word from BERT along with the BERT pooled output and glove embedding : In addition to above feature vectors , we also included word vector of target word from sequence output of BERT uncased base model
5. bagging and boosting models stacked with the models described at above lines : We finally used above mentioned model with bagging and boosting regressor stacked together in ensemble format. Using base model like random forest and multi layer neural network.

7 Experiments

1. Initially we started using the basic models like linear regression and Random forest models to get the regression score with the obtained features set, but the result obtained was very bad. R2_Score was less than 0. Thus we moved

towards the multi-layer layer neural network. As already mentioned in the procedure.

2. Used Weighted Glove vector having more weight to the words which were near to the target word, but it didn't worked as for most of the sentences there were long distance relation between the words.

3. Also gave a try by using some lexical features of the sentence like hyponyms, hepernymers, synonyms, antonyms but found that these features are not that promising and not affecting the working of the model much.

4. Also in Multi-layer layer neural network first when we have not used the dropout, the scores on the test data was not good as compared to the train set, thus we used a dropout of 20% to prevent the over fitting.

5. We also tried to add a feature which describes the number of splits of this word when given to word tokenizer. But the result was not impressive. As very few words were splitted and thus no significant information were there for the model to extract from this feature.

8 Results

Single Word Target

Model Type	Model	Pearson Score	R2 Score
Single	Keras Multilayer Neural Network (KNN)	0.7367	0.5402
Single	AdaBoost on default estimator	0.7266	0.5213
Single	AdaBoost on Keras NN	0.7352	0.5182
Single	Bagging Regressor with Keras NN	0.7423	0.5505
Single	GradientBoosting Regressor	0.7237	0.5191

Model Type	Model	Pearson Score	R2 Score
Stacked	Adaptive Boost with KNN Adaptive Boost with Random Forest KNN	0.7568	0.5720
Stacked	Adaptive Boost with KNN Bagging Regressor with KNN GradientBoosting Regressor	0.7569	0.5724
Stacked	Adaptive Boost with KNN Bagging Regressor with KNN GradientBoosting Regressor Adaptive Boost with Random Forest	0.7574	0.5730

Model Type	Model	Pearson Score	R2 Score
Stacked	Adaptive Boost with KNN Bagging Regressor with KNN GradientBoosting Regressor KNN	0.7594	0.5761
Stacked	Adaptive Boost with KNN Bagging Regressor with KNN GradientBoosting Regressor Adaptive Boost with Random Forest KNN	0.7596	0.5765

Model Type	Model	Pearson Score	R2 Score
Stacked	**KNN** **Adaptive Boost with KNN** **Bagging Regressor with KNN** **GradientBoosting Regressor** **Adaptive Boost with Random Forest** **Bagging Regressor with Random Forest**	**0.7604**	**0.5776**
Stacked	Mean of: Adaptive Boost Bagging Regressor GradientBoosting Regressor KNN	0.7530	0.5666

Multi Word Target

Model Type	Model	Pearson Score	R2 Score
Single	AdaBoost on Keras NN	0.7593	0.5733
Single	Bagging Regressor with Keras NN	0.7769	0.5982
Single	Gradient Boosting Regressor	0.7796	0.5906
Single	AdaBoost on default estimator	0.7822	0.6093
Single	Keras Multilayer Neural Network (KNN)	0.7700	0.5929
Single	BaggingRegressor with base estimator	0.7992	0.6293

Model Type	Model	Pearson Score	R2 Score
Stacked	Adaptive Boost with KNN Adaptive Boost with Random Forest KNN	0.8230	0.6682
Stacked	Adaptive Boost with KNN Bagging Regressor with KNN Gradient Boosting Regressor	0.8061	0.6436
Stacked	Adaptive Boost with KNN Bagging Regressor with KNN GradientBoosting Regressor Adaptive Boost with Random Forest	0.8098	0.6498

Model Type	Model	Pearson Score	R2 Score
Stacked	Adaptive Boost with KNN Bagging Regressor with KNN GradientBoosting Regressor KNN	0.8155	0.6544
Stacked	Adaptive Boost with KNN Bagging Regressor with KNN GradientBoosting Regressor Adaptive Boost with Random Forest KNN	0.8173	0.6576

Model Type	Model	Pearson Score	R2 Score
Stacked	Adaptive Boost with KNN Bagging Regressor with KNN GradientBoosting Regressor Adaptive Boost with Random Forest KNN Bagging Regressor with Random Forest	0.8135	0.6522
Stacked	**Mean of:** **Adaptive Boost** **Bagging Regressor** **GradientBoosting Regressor** **KNN**	**0.8285**	**0.6627**

9 Conclusion

From our experimentation of various models , we find that stacked model which takes all the context vectors like BERT pooled output, BERT word vector of the target word from the sequence output,glove vectors of both the target word and average glove embedding of all the words in a sen-

tence and also the lexical features we got from each sentence gives the best Pearson and r2 score . We found that for single word we selected the model based on stacking of all 6 models linearly. For multi-word complexity analysis we used mean model of 4 single models as described earlier. Results we obtained are not that high and some improvements are still needed.

References

[1] An adaboost regressor), 2007 - 2020.

[2] Gradient boosting for regression, 2007 - 2020.

[3] Keras neural network, 2007 - 2020.

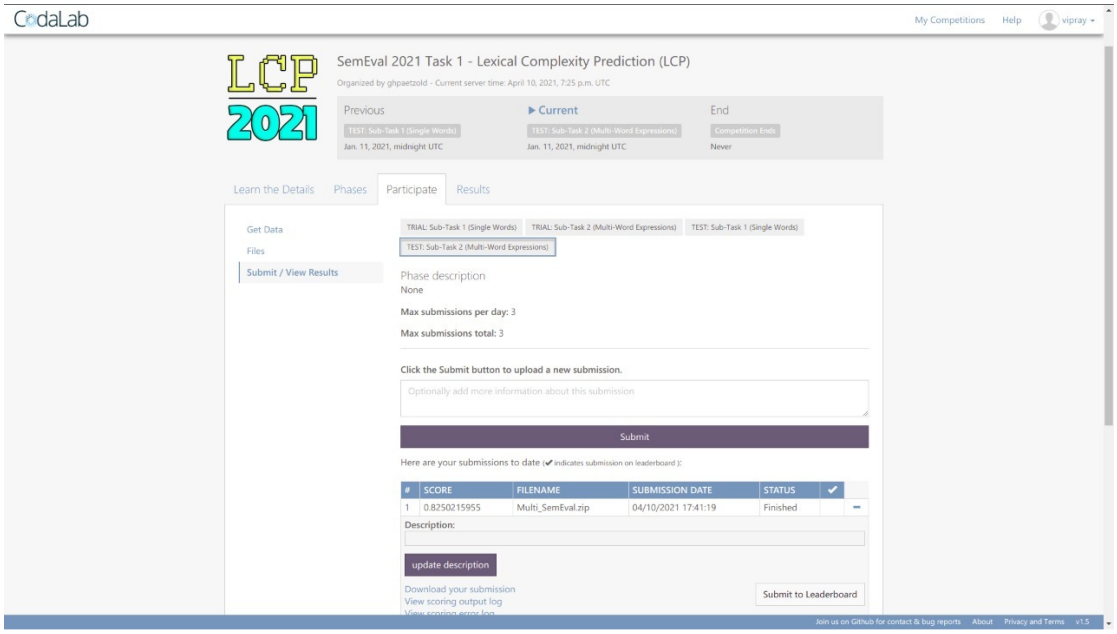
[4] Lcp shared task 2021, Mar 2021.

[5] FAJARDO, M. A. H. Complex word identification. *Project is to identify whether a word is considered complicated or not* (2018).

[6] JACOB DEVLIN, MING-WEI CHANG, K. L. K. T. Bert: Pre-training of deep bidirectional transformers for language understanding.

10 Submission Screenshot

10.1 Test: Subtask-1 (Single Words)



10.2 Test: Subtask-2 (Multi-Word Expressions)