Team 28: SemEval 2021 Task 1 - Lexical Complexity Prediction (LCP)

https://github.com/hvarS/CS60075-Team28-Task-1

1 Task and Group Details

Lexical complexity plays a crucial role in reading comprehension. Predicting lexical complexity accurately can enable a system to better guide a user to an appropriate text, or tailor a text to their needs. NLP systems have been developed to simplify texts for second language learners, native speakers with low literacy levels, and people with reading disabilities.

This report presents our best performing systems Team 28 submitted to the SemEval Task1. We describe our best performing system's implementations and discuss our key findings from this research.

Name	Roll Number
Adhikansh Singh	17CS30002
Ankit Bagde	17CS30009
Sanket Meshram	17CS30030
Harshvardhan Srivastava	17EE10058
Fardeen Farhan Pettiwala	17IE10042

Table 1: Team 28 details

2 Individual contributions

Every member of the group contributed **equally** towards the submission. Adhikansh Singh (17CS30002) and Sanket Meshram (17CS30030) extracted the features from the dataset and other corpuses. Fardeen Farhan Pettiwala (17IE10042) worked the baseline models and it's experimentation. Ankit Bagde (17CS30009) and Harshvardhan Srivastava (17EE10058) worked on modeling and experimenting the attention-based approaches. Once code was ready, a few approaches required hyper-parameter tuning, every member collaborated on this to get the final results.

3 Approaches

3.1 Feature Selection

The set of features employed in our experiments are based on the insights from the CWI shared task 2016 Paetzold and Specia (2016), Gooding and Kochmar (2018) which used Google N-gram word frequencies, POS tags, dependency parsing relations, and Wani et al. (2018) which used Word length, syllable counts, vowel counts and WordNet-based features (Fellbaum (2005)).

One aspect in which our approach significantly deviates from the others is the treatment of all Multi Word Expressions(MWE) as compositional. MWEs can be classified as compositional or non-compositional. Compositional MWEs take their meaning from the constituent words in the MWE(e.g., *christmas tree, notice board, golf cart, etc.*), whereas non-compositional MWEs do not (e.g., *hot dog, red herring, reverse ferret*).

In order to treat multi word tokens as compositional, many features such as POS tag, number of hypernyms, hyponymns, etc. were given 2 values corresponding to each feature. In such cases, 2 features are constructed for each field. Handling the training for the compositional MWEs is explained in the latter sections.

Consequently the chosen features are as follows:

1. **Token Length**: The number of characters in the token.

2. **Number of Syllables**: The number of syllables in the target word. The Datamuse API was used to collect this information.

3. **Token Vowels**: The number of vowels in the token.

4. **WordNet Features**: The target words lemma was extracted followed by the extraction of

number of senses (Synonyms), number of hypernyms and hyponyms for the word's lemma from WordNet (Fellbaum (2005)).

5. **POS and Word N-gram**: The NLPCore pipeline was used for n-grams and the parsed sequences were generated corresponding to each sentence. The part of speech tags(POS) of the target word was then obtained from the parse extracted and performing part of speech tagging on all sentences containing target words using the NLPCore pipeline (Manning et al. (2014)). For multi words the POS was generated for each individual word.

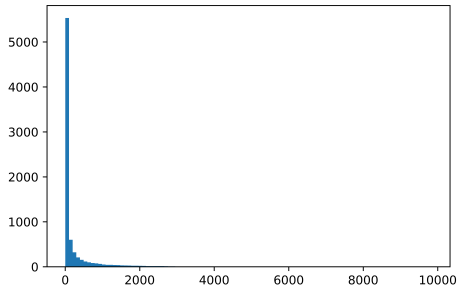
6. **Number of Dependencies from Parse Relations**: The data was parsed using the NLPCore pipeline, and the number of dependency relations for the target word(token) are extracted and used as a feature.

7. **Word Frequency**:The frequency of the target word was then estimated using the Google dataset of syntactic *n* grams (Goldberg and Orwant (2013)).

8. **Occurrence in the SubIMDB corpus**: The data collected from the SubIMDB corpus (Paetzold and Specia (2016)) was used after a preprocessing pipeline of lowercase, removing punctuation, stopwords and lemmatizing all the words. Then the frequency of each word was calculated. Finally the top 1000 words were considered, creating a binary feature which will be **False** if all the words in the token are in top selected words, and **True** Otherwise. The intuition being that if a word is appearing frequently then it is not likely to be more complex. A similar approach is followed in the subsequent features.

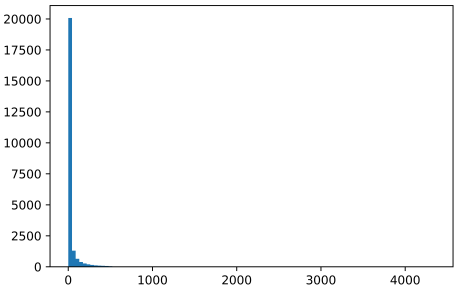
9. **Occurrence in Simple Wikipedia corpus (SimpWiki)**: Similar to above, the SimpleWiki dataset was used (Coster and Kauchak (2011)) following the preprocessing pipeline. Here, we considered top 6368 words.

10. **Occurence in the English Bible**: The data was collected from[1] Christodouloupoulos and Steedman (2015) and the frequency of each word was calculated. Then a the histogram of cumulative frequency was plotted in the direction of decreasing frequency starting from max frequency :



It was observed that the quantity of infrequent words is a lot more then frequent words, as evident from the sudden spike in plot. From manual inspection and plots the top 1158 words were chosen.

11. **Occurrence in the Biomedical corpus**: We collected the data from[2] Bada et al. (2012) and then plotted the histogram of cumulative frequency in the direction of decreasing frequency starting from max frequency:



Top 544 words were chosen from the plot and manual inspection.

12. **Familiarity of token**: This comes from Shardlow et al. (2021) which postulates that the words which are more familiar are less likely to be complex. So a familiarity score was extracted from here[3] which measured familiarity in the form of a score between 100 to 700 based on Wilson (1988) which was added as a feature. In case of multi-word token the average familiarity score of both the tokens was taken. We named the feature **familiarity**

[1] https://github.com/christos-c/bible-corpus

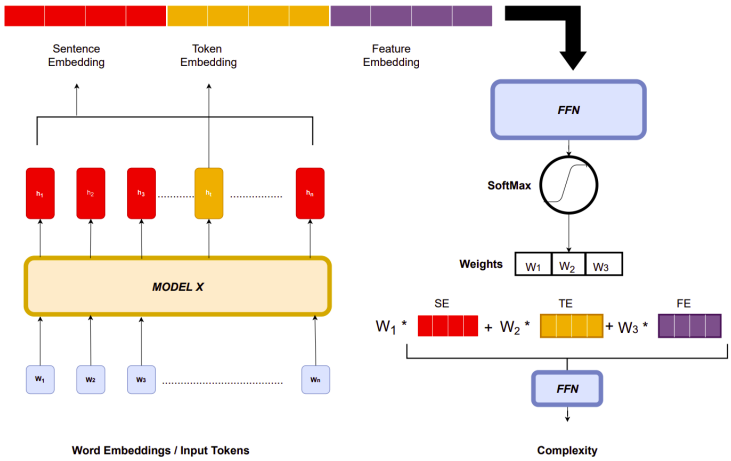


Figure 1: Overview of the Attention-based approach

4 Experiments

In this section, we discuss the preprocessing pipeline, the baseline models, and some other approaches used to include features and contextual information during predicting the complexity.

4.1 Preprocessing and Feature extraction

As stated above, for preprocessing each sentence and it's corresponding token, we follow a simple pipeline. Lowercase ; remove punctuations ; remove stopwords ; lemmatize. Same is applied to tokens too, but care is taken that a token is not omitted.

Apart from preprocessing, we extract features as described above in 3.1. Many features such as POS tag, number of hypernyms, hyponymns, etc. possess 2 values in case of multiword token. In such cases, 2 features are constructed for each field. When combining both data, features for single-word tokens are duplicated.

4.2 Baseline Models

We developed a baseline for predicting the complexity of single and multi-word tokens using the 300-dimensional Glove embedding for each token Pennington et al. (2014). In case of an unknown token, the mean of the sentence embedding was assigned to the token embedding. In case of multi-word tokens, mean of Glove embeddings of both words in a token is taken. We use five different regression models to predict the complexity, namely linear regression, gradient boosting, AdaBoost, MLP Regressor (a hidden layer of size 150) and SVM.

Here, two methods are possible, either take corresponding training data for predicting single and multi-word data or combine both datasets for prediction. As multi-word task data is quite less, we thought of experimenting both approaches. The results are provided in Table 2 and Table 3 which are discussed in the later sections of the report.

4.3 Combining features

Keeping the same setting as of baseline models, and concatenating the features which we extracted as explained in 3.1 and 4.1, we perform the experiments again to check the effect of adding the features. All the features, which are non-binary are standardized by removing the mean, and scaling it to unit variance.

4.4 Attention-based approaches

In our previous experiments, we considered the extracted features and Glove embeddings of tokens of the sentences, ignoring the contribution of the neighbouring part of sentence. The complexity of a token also depends on it's context. In this section, we list the experiments we performed in this direction.

Based on the data provided, we have the following: vectorized representation of each word in a sentence, a sentence embedding, feature embedding (our extracted features), token embedding. To incorporate these sentence-level represenation and feature embedding, we use simple attention based method.

In simple sense, attention-based approaches try

[2] http://bionlp-corpora.sourceforge.net/CRAFT/

[3] https://websites.psychology.uwa.edu.au/school/mrcdatabase/uwa_mrc.htm

to pay attention to the specific input vectors of a input sequence based on the attention weights. It is widely used in many different forms, for example (Lim et al., 2020) presents a very simple network to select the important multiple variables as their relevance and specific contribution to the output might not be known initially, (Yang et al., 2016) provides a simple word attention approach for converting word embeddings to sentence embeddings and so on for document classification.

Figure 1 provides an overview of our approach, say using some model X, we generate a **sentence representation** and **word representation** of each word in a sentence which includes contextual information from both sides, in addition to this we have **feature embedding**. If we know, how tokenization is performed, i.e the start and the end index of the token in a sentence, we can extract the **token embedding**, which is the mean of the word representations for that particular range.

Thus, we have a total of 3 features, sentence embedding ($sentEmb$), token embedding ($tokenEmb$) and feature embedding ($featureEmb$). These three are concatenated and passed to a hidden dense layer followed by a dense layer with softmax activation to output a 3-dimensional weight vector(w).

Approach	Train Data	Model	Pearson	Spearman	MAE	MSE	R2
Baseline	Single	LR	0.682	0.662	0.073	0.009	0.459
		GB	**0.721**	**0.695**	**0.067**	**0.008**	**0.520**
		AdaBoost	0.694	0.671	0.071	0.009	0.472
		MLP	0.467	0.434	0.113	0.022	-0.354
		SVM	0.710	0.683	0.0704	0.008	0.502
	Single + Multi	LR	0.672	0.655	0.076	0.009	0.413
		GB	**0.694**	**0.673**	**0.072**	**0.009**	**0.464**
		AdaBoost	0.663	0.636	0.084	0.011	0.307
		MLP	0.521	0.517	0.110	0.0218	-0.346
		SVM	0.667	0.656	0.075	0.009	0.429
With Features	Single	LR	0.691	0.666	0.071	0.009	0.472
		GB	**0.742**	**0.710**	**0.066**	**0.007**	**0.549**
		AdaBoost	0.703	0.674	0.070	0.008	0.491
		MLP	0.524	0.514	0.108	0.020	-0.220
		SVM	0.716	0.690	0.070	0.008	0.510
	Single + Multi	LR	0.687	0.676	0.072	0.009	0.458
		GB	**0.738**	**0.700**	**0.067**	**0.007**	**0.542**
		Adaboost	0.703	0.687	0.072	0.009	0.458
		MLP	0.467	0.450	0.107	0.020	-0.206
		SVM	0.704	0.682	0.071	0.008	0.488
Attention-based	Single	**Bi-LSTM**	**0.714**	**0.687**	**0.068**	**0.008**	**0.496**
			0.716	**0.694**	**0.069**	**0.008**	**0.504**

Table 2: Comparing various approaches evaluating on **Sub-Task 1 : Single**, with different datasets used for training. Single implies dataset for Sub-task 1 and multiple implies dataset for Sub-task 2. Addition of features implies the features for each dataset extracted by us. **Bold** ones represent the best from each setting. **Underlined** ones compare the best among all three approaches 4.2, 4.3 and 4.4.

Approach	Train Data	Model	Pearson	Spearman	MAE	MSE	R2
Baseline	Multi	LR	0.738	0.710	0.084	0.011	0.536
		GB	0.782	0.753	0.079	0.009	0.605
		AdaBoost	**0.803**	**0.795**	**0.076**	**0.009**	**0.627**
		MLP	0.412	0.405	0.138	0.029	-0.202
		SVM	0.774	0.756	0.079	0.010	0.595
	Single + Multi	LR	0.818	0.805	0.106	0.016	0.327
		GB	0.819	0.809	0.105	0.016	0.335
		AdaBoost	0.794	0.775	0.098	0.014	0.410
		MLP	0.623	0.583	0.111	0.019	0.197
		SVM	**0.814**	**0.807**	**0.072**	**0.008**	**0.639**
With Features	Multi	LR	0.732	0.704	0.085	0.011	0.528
		GB	**0.796**	**0.770**	**0.075**	**0.009**	**0.627**
		AdaBoost	0.783	0.772	0.078	0.010	0.606
		MLP	0.544	0.575	0.124	0.025	-0.046
		SVM	0.792	0.770	0.077	0.009	0.618
	Single + Multi	LR	0.801	0.796	0.0765	0.009	0.611
		GB	0.743	0.730	0.085	0.012	0.518
		Adaboost	0.750	0.748	0.092	0.013	0.473
		MLP	0.633	0.643	0.105	0.017	0.294
		SVM	**0.812**	**0.800**	**0.073**	**0.008**	**0.654**
Attention-based	Single+Multi		**0.832**	**0.829**	**0.085**	**0.011**	**0.513**
		Bi-LSTM	**0.828**	**0.819**	**0.084**	**0.011**	**0.550**
			0.819	**0.824**	**0.076**	**0.008**	**0.633**

Table 3: Comparing various approaches evaluating on **Sub-Task 2 : Multi**, with different datasets used for training. Single implies dataset for Sub-task 1 and multiple implies dataset for Sub-task 2. Addition of features implies the features for each dataset extracted by us. **Bold** ones represent the best from each setting. **Underlined** ones compare the best among all three approaches 4.2, 4.3 and 4.4.

Final embedding ($finalEmb$) is calculated as: finalEmb = w1*sentEmb + w2*tokenEmb + w3*featureEmb

This final embedding is passed to dense layer, predicting the complexity of a single or multiword token. For generating contextual representation, we tried Bi-directional LSTM and BERT, sentence embedding in both cases is mean of word representations and CLS representation respectively.

5 Results and Discussion

Table 2 and 3 summarizes the results of our extensively experimented approaches, (i) *Baselines*, (ii) *Adding Features* and (iii) *Attention-Based*.

Each table has 3 sections, each for our three different approaches. Each approach is experimented on either respective training data or combination of both. For evaluation, we use the script provided here. Our scores matches the ones which are submitted to the CodaLab.

Here, we would like to discuss two important points related to dataset. Firstly, the effect of training data on particular tasks. For Task1-Single, combining both data does'nt help much. This is opposite in case of Task2-Multi, maybe because number of examples for Task2 are very less as compared to that Task1 data. This explains our experimenting

with only good performing setting for Attention-based approaches. Secondly, we did'nt include our other experiments including BERT, as model size was becoming relatively bigger, overfitting early on small amount of examples.

For Task1-Single, Gradient Boosting regressor performs best among all the models. Adding our extracted features brings a good amount of change of **0.048** to Pearson coefficient. Introducing Bi-LSTM, doesn't bring a positive difference as compared to adding features. On the other hand, for Task2-Multi we got comparable results for Baselines and after adding features. The results improves in attention-based approach by **0.018**.

CodaLab leaderboard only considers pearson coefficient for ranking, however considering the other metrics too we have included 2 and 3 different results for Attention-based approach in Task1 and Task2 respectively.

Some other experimentations were also done,(i) Using a CNN layer to extract hidden features from the sentences that were not clearly observable ; and (ii) Using ELMO Embeddings, instead of GloVe to include the context of the input sentences . However we could not achieve competitive results with approach (i) ; and approach (ii) posed computational bottleneck as we were using Google Colab as a source for GPU Accelerator.

References

Michael Bada, Miriam Eckert, Donald Evans, Kristin Garcia, Krista Shipley, Dmitry Sitnikov, William A. Baumgartner, K. Bretonnel Cohen, Karin Verspoor, Judith A. Blake, and Lawrence E. Hunter. 2012. Concept annotation in the craft corpus. *BMC Bioinformatics*, 13(1):161.

Christos Christodouloupoulos and Mark Steedman. 2015. A massively parallel corpus: the bible in 100 languages. *Language Resources and Evaluation*, 49(2):375–395.

William Coster and David Kauchak. 2011. Simple English Wikipedia: A new text simplification task. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 665–669, Portland, Oregon, USA. Association for Computational Linguistics.

Christiane Fellbaum. 2005. Wordnet and wordnets. In Alex Barber, editor, *Encyclopedia of Language and Linguistics*, pages 2–665. Elsevier.

Yoav Goldberg and J. Orwant. 2013. A dataset of syntactic-ngrams over time from a very large corpus of english books. *Second Joint Conference on Lexical and Computational Semantics*, pages 241–247.

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.

Bryan Lim, Sercan O. Arik, Nicolas Loeff, and Tomas Pfister. 2020. Temporal fusion transformers for interpretable multi-horizon time series forecasting.

Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit.

Gustavo Paetzold and Lucia Specia. 2016. Collecting and exploring everyday language for predicting psycholinguistic properties of words. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1669–1679, Osaka, Japan. The COLING 2016 Organizing Committee.

Jeffrey Pennington, Richard Socher, and Christopher 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, Doha, Qatar. Association for Computational Linguistics.

Matthew Shardlow, Richard Evans, and Marcos Zampieri. 2021. Predicting lexical complexity in english texts.

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, New Orleans, Louisiana. Association for Computational Linguistics.

Michael Wilson. 1988. Mrc psycholinguistic database: Machine-usable dictionary, version 2.00. *Behavior Research Methods, Instruments, & Computers*, 20(1):6–10.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489, San Diego, California. Association for Computational Linguistics.

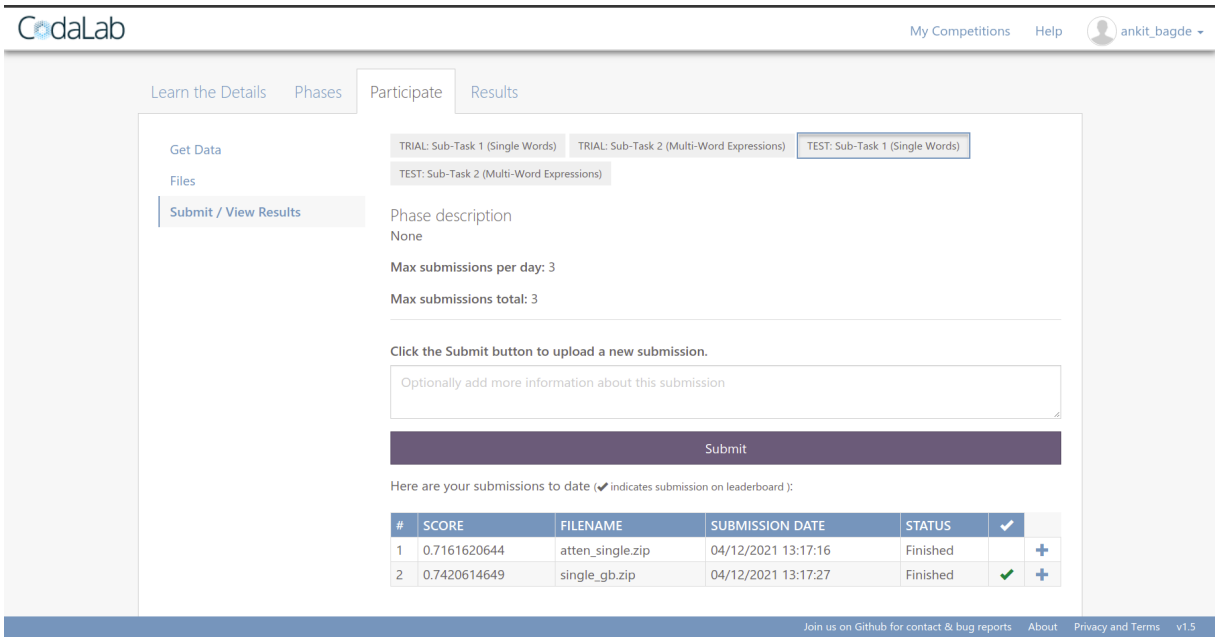


Figure 2: Task1 submission screenshot

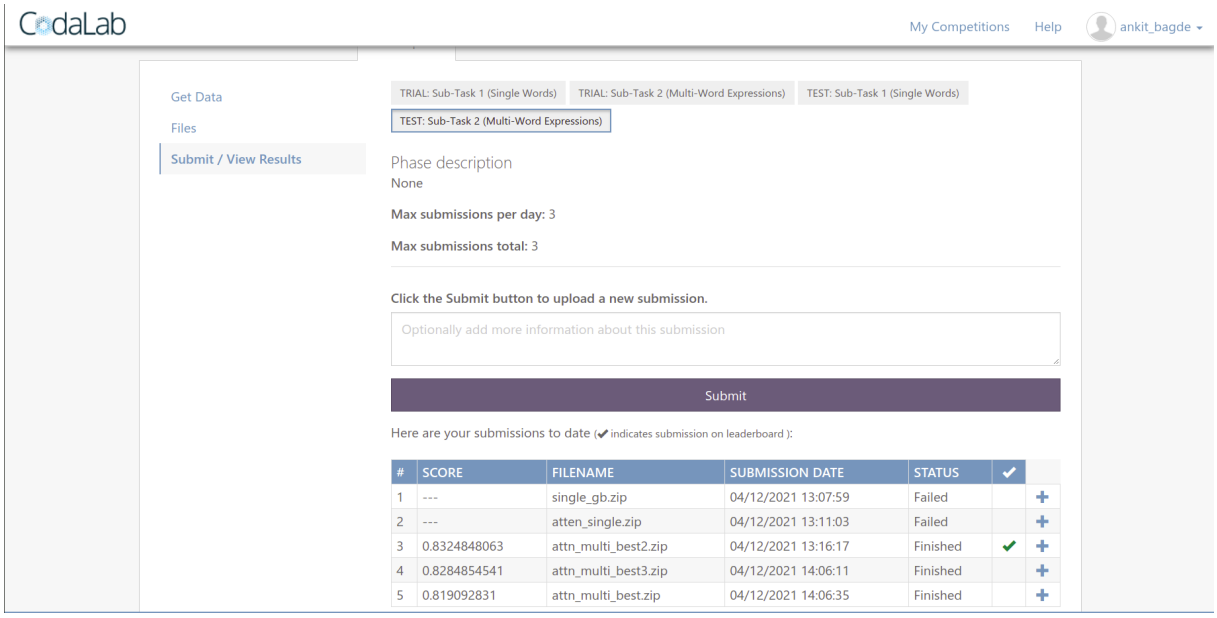


Figure 3: Task2 submission screenshot