# C3SL at SemEval-2021 Task 1: Predicting Lexical Complexity of Words in Specific Contexts with Sentence Embeddings

**Raul Gomes Pimentel de Almeida**
C3SL Labs
Universidade Federal do Paraná
Curitiba - Brazil
rgpa18@inf.ufpr.br

**Hegler Tissot**
Cognitive Computation Group
University of Pennsylvania
Philadelphia - USA
hegler@seas.upenn.edu

**Marcos Didonet Del Fabro**
C3SL Labs
Universidade Federal do Paraná
Curitiba - Brazil
marcos.ddf@inf.ufpr.br

## Abstract

We present our approach to predicting lexical complexity of words in specific contexts, as entered LCP Shared Task 1 at SemEval 2021. The approach consists of separating sentences into smaller chunks, embedding them with Sent2Vec, and reducing the embeddings into a simpler vector used as input to a neural network, the latter for predicting the complexity of words and expressions. Results show that the pre-trained sentence embeddings are not able to capture lexical complexity from the language when applied in cross-domain applications.

## 1 Introduction

Lexical complexity plays a crucial role in reading comprehension. Predicting lexical complexity of words within sentences in specific textual context can enable systems to better perform certain NLP tasks, such as simplifying texts, and favoring less fortunate readers in a giving target language. The SemEval 2021 proposes a Lexical Complexity Prediction (LCP) shared task (Task 1) (Shardlow et al., 2021) based on a new annotated English dataset with a Likert scale (Shardlow et al., 2020).

Word embeddings (Mikolov et al., 2013; Devlin et al., 2019) are very important resources that support several NLP tasks by providing a semantic latent representation that heuristically captures relationships in language that are very difficult to observe otherwise. Furthermore, such semantic representation can be used to embed language structures other than just words.

Sent2Vec (Pagliardini et al., 2018) is an unsupervised model designed to compose sentence embeddings using word vectors along with n-gram embeddings, simultaneously training composition and the embedding vectors themselves. Sent2Vec has been used in several NLP tasks, such as analysing semantic properties of sentences (Zhu et al., 2018), classification of sentences in the biomedical domain (Agibetov et al., 2018), automatic detection of incoherent speech (Iter et al., 2018), and measuring sentence similarity (Quan et al., 2019),

In this work, we aim to test the ability of Sent2Vec to detect the complexity of English words and expressions in specific contexts. We evaluate in what extent the semantic information captured when learning the embeding representation is able to incorporate word complexity.

Our approach uses pretrained Sent2Vec models and aims to validate in what extent such models are able to predict complexity of words. Results show strong evidence that pretrained sentence embeddings do not capture complexity features from language, specially when applied in cross-domain applications.

## 2 Method

The proposed shared task consists of determining the complexity of token words in the context of given input sentences (Lexical Complexity Prediction-LCP). Training input sentences annotated with a Likert scale corresponding to the complexity score of target words are given. Sub-task 1 focus on predicting the complexity score of single words. Meanwhile, Sub-task 2 targets multi-word expressions.

Our overall strategy consists on generating chunks of the sentence, embed those chunks with Sent2Vec and then reduce those embeddings into a simpler vector, which would then finally be used as the neural network's input.

Our approach uses pretrained Sent2Vec models to obtain embedding representation of multiple parts of the input sentence split by the target token words. For a given input sentence $S$, we obtain embeddings for: a) $S$: the full sentence; b) $S_0$: the amount of text from the beginning of the sentence



*Proceedings of the 15th International Workshop on Semantic Evaluation (SemEval-2021)*, pages 683–687
Bangkok, Thailand (online), August 5–6, 2021. 

up to the target token(s) (except the latter); c) $T$: the target word token(s), and d) $S_1$: the amount of text from the target token(s) up to the end of the sentence (except the former). When target tokens are in the beginning or in the end of a sentence, $S_0$ or $S_1$ are represented by a zero-value vector.

We reduce the vector representation of each sentence constituent into distance-based arrays that are fed into a neural network estimator. The reduction itself consisted of simplifying the text (using Spacy) and then feeding different chunks into Sent2Vec. Finally, an array of the distances between the chunks' embeddings to the token word's embeddings was used as input to a neural network coming from the Scikit-Learn package.

We use the context given by the two splits of the input sentence ($S_0$ or $S_1$) to measure the complexity of the token word(s) ($T$). Thus, our pipeline consists of four major steps: *text preprocessing*, *chunking*, *context embedding* and *estimator training*. The chosen estimator was a Multi-Layer Perceptron (MLP).

Figure 1 illustrates this pipeline for an example sentence and token word.

### 2.1 Text Preprocessing

We use a simple preprocessing step based on off-the-shelf components from Spacy Python package (Honnibal et al., 2020)[1] that apply the following filters in the raw input sentence:

- Turning it into only lowercase characters;
- Removing all punctuation;
- Removing all words recognized by Spacy as stop words (unless the stop word was part of the target token $T$).

The reformatted sentences are then split into parts $S$, $S_0$, $S_1$ and $T$ to be fed into the next step of the pipeline.

### 2.2 Context Embedding

We embed the four elements of a sentence resulted from the previous preprocessing step using a pretrained Sent2Vec model (`torontobooks_unigrams` model has been chosen).[2] As a setback, the token word was often embedded as a null vector (consisting only of zeroes), as expected that such target words from specific domains would be off the pretrained vocabulary.

Instead of concatenating the four resulting embeddings into a single vector to be used as input by our estimator, we perform a reduction that aims to represent how close (distance) to its context a token word is. Thus, our estimator input comprises: a) three norm distances (NumPy's `linalg.norm` method (Harris et al., 2020)) between each sentence embedding ($S$, $S_0$, and $S_1$) and the token word's embedding ($T$), and b) a boolean value in the set $\{0, 1\}$ indicating whether the token word's embedding was a null vector.

### 2.3 Estimator Training

Finally, we use the obtained embedding reductions from the previous step to train a MLP from the Scikit-Learn Python package (`neural_network.MLPRegressor`) (Pedregosa et al., 2011). The MLP was instantiated with the `max_iter` and `random_state` arguments set to 500 and 1, respectively. The estimator was fitted to the input with its `fit` method, with the `X` and `y` parameters being vectors of the reduced embeddings (`X`) and corresponding given token word complexities (`y`).

Our estimator is designed to predict the complexity value for a given token word in context. However, token words can be a misuse of given information, since the complexity of a word varies with context. Our goal with reducing embeddings from fractions of the original sentence into a small vector that served as input to the estimator was similar to the Input Hypothesis, an approach to language learning that grows in popularity.

The Input Hypothesis, also referred to as "i+1 method" and surveyed in Wang (2017), discusses a process for acquiring languages by being exposed to content that is slightly above the learner's current level. This is argued to work because the learner is then able to understand the whole sentence (i) and consequently annex a new word or construction (+1) to their vocabulary because of intelligible context.

Reducing embeddings from chunks of the original sentence into a vector of distances from the token word can be expressed as an attempt to symbolize the size (complexity) of this "+1" (the token word) - that is, the neural network input is an observation of how obtainable from its context a given

[1] https://spacy.io

[2] https://github.com/epfml/sent2vec#downloading-sent2vec-pre-trained-models

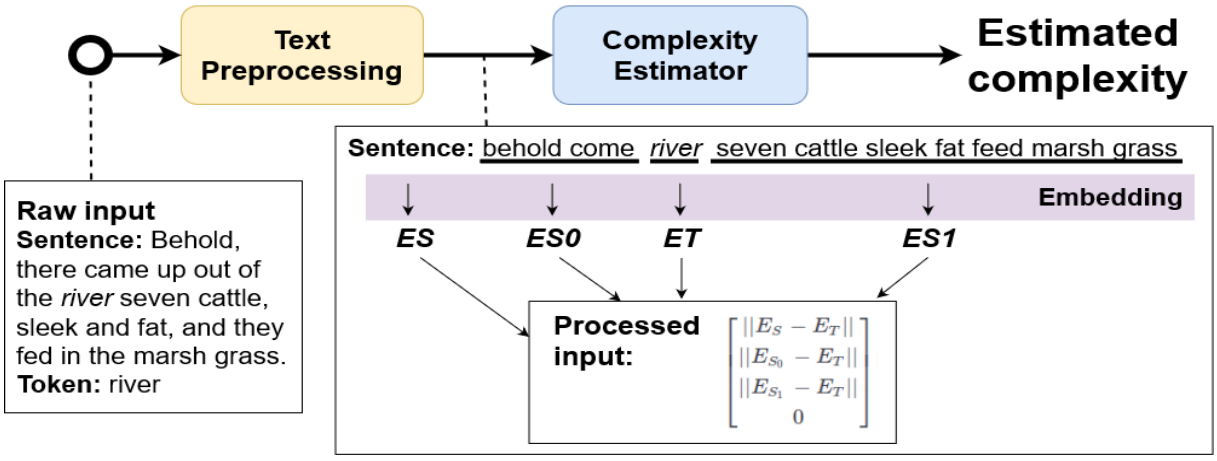


Figure 1: Illustrated pipeline for our approach

token word is.

We then validate the trained model using `model_selection.train_test_split` method (Scikit-Learn), setting `random_state` parameter set to 1. This method divides the dataset into training and testing subsets. With the MLP fitted with the training subset, the efficiency with which it would handle foreign inputs is measured with a call to its `score` function (using the testing sets as arguments). This function returns the coefficient of determination $R^2$ of a prediction.

### 2.4 Development Environment

Our solution is made available in a Google Colab file[3]. In order to execute the code, it's necessary to duplicate the file and change the environment variables related to file access (the Sent2Vec models and input files). More information on this process can be found on the file itself.

Since the Google Colab platform permits by default the use of up to a little over 12.5GB RAM, the options of Sent2Vec pretrained models to use are limited: the model we used for this task has only 2GB.

## 3 Results

The task is divided between two subtasks: single- and multi-word tokens. Both have the same format: each line of the input has an ID, a sentence, a corpus to which it belongs, and a token. For training data, each line also includes a complexity value. The difference between the first and second subtasks is that the first limits a token to one single word, while the second doesn't.

Tables 1 and 2 present the final results of the competition for the two subtasks, respectively. Participant's performance is ranked in each subtask separately. For the submissions without a team name - that is, the submissions made by users not linked to teams -, the user name is available inside brackets. The tables also show each participant's scores for individual corpora. Our approach is identified as C3SL.

We believe the flaws in our approach can be explained in two-fold: a) pretrained embeddings are sensitive to the domain or context used during trained, and the way semantic information is captured in a latent representation is not reflected the in the same way in cross-domain applications; and b) even if a latent representation of text would able to capture semantic complexity of certain expression in context, the same is not reflected by the norm or cosine distances between multiple chunks and the target expressions.

## 4 Related Work

Some of the previous work show that Sent2Vec is not the best alternative to consistently achieve high accuracy in subsequent NLP and NLU tasks based on embedding representation of sentences, except when training Sent2Vec with domain-specific corpora.

Miftahutdinov et al. (2019) attempt to use the Twitter unigram pretrained model from Sent2Vec in order to improve their approach when performing extraction of adverse drug reactions from Tweets. However, results show that utilizing Sent2Vec as tweet representations did not improve classification quality.

[3] https://colab.research.google.com/drive/1MCNfDzM-BW9Zopxs9qFFT8sDLb6FrAM_?usp=sharing

Table 1: Results for Subtask 1.

| # | Team Name | Pearson | Spearman | MAE | MSE | R2 |
|---|---|---|---|---|---|---|
| 1 | JUST BLUE | 0.7886 (1) | 0.7369 (2) | 0.0609 (61) | 0.0062 (60) | 0.6172 (2) |
| 2 | DeepBlueAI | 0.7882 (2) | 0.7425 (1) | 0.0610 (60) | 0.0061 (61) | 0.6210 (1) |
| 3 | Alejandro Mosquera | 0.7790 (3) | 0.7355 (5) | 0.0619 (57) | 0.0064 (59) | 0.6062 (3) |
| 57 | C3SL | 0.4598 (57) | 0.3983 (58) | 0.0866 (6) | 0.0130 (6) | 0.1989 (56) |
| 61 | RACAI | -0.0272 (61) | -0.0268 (61) | 0.2777 (1) | 0.1270 (1) | -6.8449 (61) |

Table 2: Results for Subtask 2.

| # | Team Name | Pearson | Spearman | MAE | MSE | R2 |
|---|---|---|---|---|---|---|
| 1 | DeepBlueAI | 0.8612 (1) | 0.8526 (3) | 0.0616 (38) | 0.0063 (38) | 0.7389 (1) |
| 2 | [rg_pa] | 0.8575 (2) | 0.8529 (2) | 0.0672 (34) | 0.0072 (34) | 0.7035 (5) |
| 3 | [xiang_wen_tian] | 0.8571 (3) | 0.8548 (1) | 0.0675 (33) | 0.0072 (32) | 0.7012 (7) |
| 35 | C3SL | 0.3941 (35) | 0.3675 (35) | 0.1145 (4) | 0.0206 (4) | 0.1470 (35) |
| 38 | [glitterosu] | 0.1860 (38) | 0.1316 (38) | 0.1332 (1) | 0.0255 (1) | -0.0564 (38) |

Cho et al. (2019) use SentVec embeddings to propose a language scheme that generates candidate utterances using paraphrasing and methods from semi-supervised learning.

An empirical study of sentence embeddings (Krasnowska-Kieraś and Wróblewska, 2019) aims to analyse in what extent linguistic information is retained in vector representations of sentences by comparing ten embeddings approaches. Results show that Sent2Vec was only able to outperform other approaches in only 1 out of 11 tasks (word classification task).

Lo et al. (2018) use Sent2Vec trained on the WMT18 news translation task parallel training data (Koehn et al., 2018) to calculate distance of sentence vectors aiming to improve their semantic textual similarity approach when filtering a noisy web crawled parallel corpus.

Iter et al. (2018) aim to automatically extract linguistic features for detecting symptoms of schizophrenia. They compare a number of sentence embeddings and show that, although Sent2Vec outperforms the mean vector sentence embedding (used as a baseline model within the experiments), all other models perfom better when measuring coherence using concept overlap and ambiguous pronoun usage.

Zhu et al. (2018) evaluate semantic properties of sentence embeddings models in five tasks: negation detection, negation variants, clause relatedness, argument sensitivity, and fixed point reorder. Results show that Sent2Vec is only able to outperform other embedding models in the clause relatedness task, which explores whether the similarity between a sentence and its embedded clause is higher than between a sentence and its negation. The resulting accuracy is in the 30-35% range.

## 5 Conclusions

We presented an approach for the LCP Shared Task 1 at SemEval 2021 for predicting the lexical complexities of words in context. We used pretrained Sent2Vec models using a vector formed from embedded chunks of sentences as input for a neural network to perform as the final complexity estimator. Implemented and executed on the Google Colab platform, our approach used little resources.

The results show that pretrained Sent2Vec models alone cannot capture a semantic representation that reflects a word's or expression's complexity within cross-domain contexts. As a way of extending our experiments, we plan to perform further analysis in subsequent classification tasks in the biomedical domain using reinforcement strategies to enrich the semantic information captured by embedding representation of sentences.

## References

Asan Agibetov, Kathrin Blagec, Hong Xu, and Matthias Samwald. 2018. Fast and scalable neural embedding models for biomedical sentence classification. *BMC Bioinformatics*, 19(1).

Eunah Cho, He Xie, and William M. Campbell. 2019. Paraphrase generation for semi-supervised learning

in NLU. In *Proceedings of the Workshop on Methods for Optimizing and Evaluating Neural Language Generation*, pages 45–54, Minneapolis, Minnesota. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Charles R. Harris, K. Jarrod Millman, St'efan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fern'andez del R'ıo, Mark Wiebe, Pearu Peterson, Pierre G'erard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. 2020. Array programming with NumPy. *Nature*, 585(7825):357–362.

Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. 2020. spaCy: Industrial-strength Natural Language Processing in Python.

Dan Iter, Jong Yoon, and Dan Jurafsky. 2018. Automatic detection of incoherent speech for diagnosing schizophrenia. In *Proceedings of the Fifth Workshop on Computational Linguistics and Clinical Psychology: From Keyboard to Clinic*, pages 136–146, New Orleans, LA. Association for Computational Linguistics.

Philipp Koehn, Huda Khayrallah, Kenneth Heafield, and Mikel L. Forcada. 2018. Findings of the WMT 2018 shared task on parallel corpus filtering. In *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, pages 726–739, Belgium, Brussels. Association for Computational Linguistics.

Katarzyna Krasnowska-Kieraś and Alina Wróblewska. 2019. Empirical linguistic study of sentence embeddings. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5729–5739, Florence, Italy. Association for Computational Linguistics.

Chi-kiu Lo, Michel Simard, Darlene Stewart, Samuel Larkin, Cyril Goutte, and Patrick Littell. 2018. Accurate semantic textual similarity for cleaning noisy parallel corpora using semantic machine translation evaluation metric: The NRC supervised submissions to the parallel corpus filtering task. In *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, pages 908–916, Belgium, Brussels. Association for Computational Linguistics.

Zulfat Miftahutdinov, Ilseyar Alimova, and Elena Tutubalina. 2019. KFU NLP team at SMM4H 2019 tasks: Want to extract adverse drugs reactions from tweets? BERT to the rescue. In *Proceedings of the Fourth Social Media Mining for Health Applications (#SMM4H) Workshop & Shared Task*, pages 52–57, Florence, Italy. Association for Computational Linguistics.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc.

Matteo Pagliardini, Prakhar Gupta, and Martin Jaggi. 2018. Unsupervised learning of sentence embeddings using compositional n-gram features. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 528–540, New Orleans, Louisiana. Association for Computational Linguistics.

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.

Z. Quan, Z. Wang, Y. Le, B. Yao, K. Li, and J. Yin. 2019. An efficient framework for sentence similarity modeling. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 27(4):853–865.

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)*.

Yan Wang. 2017. The Influence of "i+1" on Chinese Foreign Language Teaching Methods. *Sino-US English Teaching*, 14(8).

Xunjie Zhu, Tingfeng Li, and Gerard de Melo. 2018. Exploring semantic properties of sentence embeddings. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 632–637, Melbourne, Australia. Association for Computational Linguistics.