Complex Word Identification: a Comparative Study Between ChatGPT and a Dedicated Model for this Task

Abdelhak Kelious[1] , **Mathieu Constant**[1], **Christophe Coeur**[2]
University of Lorraine and CNRS/ATILF[1], consultant[2]
{abdelhak.kelious, mathieu.constant}@univ-lorraine.fr, christophe.coeur@gmail.com

Abstract

This paper focuses on the task of lexical complexity prediction. We explore deep learning methods to assess the complexity of a word based on its context. Specifically, we investigate how to use pre-trained language models to encode both the sentence and the target word, and then fine-tune them by combining them with additional frequency-based features. Our approach outperforms the best systems in SemEval-2021 (Shardlow et al., 2021). Finally, we carry out a comparative study with ChatGPT to assess its potential for predicting lexical complexity, and whether prompt engineering can be an alternative to this task.

Keywords: Natural language processing, Lexical complexity prediction, Language models, ChatGPT

1. Introduction

Understanding language is a complex process that requires a multitude of linguistic skills such as knowledge of grammar, acquisition of vocabulary, and the ability to produce correct and fluent sentences. The complexity of words can play an important role in this process as it can influence the speed and ease with which learners acquire new language skills. Sometimes, the presence of a word in an unusual or infrequent context can make it difficult for a reader to comprehend, causing them to either give up, misinterpret, or continue without understanding. Even an attentive reader may need to look up the meaning of a word, which can detract from their concentration and overall understanding of the text. Natural language processing can be used to identify difficult words in a text and thus help readers better understand the content.

Lexical complexity prediction is a research field that focuses on predicting the complexity level of a lexical item in a given textual context. Research in this field has explored various approaches (North et al., 2023), including the use of lexical features such as word frequency or sentence length (Zampieri et al., 2016), as well as the use of pre-trained language models such as BERT (Devlin et al., 2019), etc. These works have important practical applications, particularly in the field of language teaching and reading comprehension (Alfter, 2021). However, there is still much to be done to improve the accuracy of lexical complexity prediction and research in this field is therefore continuing to explore new approaches.

Inspired by the work published in SemEval-2021 Task 1 (Shardlow et al., 2021), we propose a method for predicting lexical complexity by combining pre-trained language models, such as DeBerta (He et al., 2023), with features based on text frequency. The rarity of words can have an influence on their level of complexity (Chen and Meurers, 2016), and the frequency of a word can be correlated with its complexity. Therefore, our approach of combining these two features could provide accurate results for predicting lexical complexity.

The development of generative Large Language Models (LLMs), such as ChatGPT, has recently gained significant attention from the scientific community. However, their evaluation on real data remains relatively unexplored, primarily due to the challenges posed by the output produced by these models. In our paper, we will conduct a comparative study on lexical complexity and propose a comparative method for evaluating the performance of ChatGPT.

The main contributions of this paper are the following:

- A new model trained on Complex 2.0 data, combining pre-trained language models with frequency features.
- An evaluation of ChatGPT's capability for our task. Finding: it is not proficient at predicting a complexity score, but has some ability to rank contexts by their complexity, especially when the contexts are clearly difficult or clearly easy.

2. Related work

2.1. Complex word identification: methods and datasets

There are several reasons to evaluate the identification of complex words in a sentence (context), and this has been the subject of research for several years. For instance, this has been explored in the context of lexical simplification, which involves

LREC-COLING 2024, pages 3645–3653

automatically replacing complex words with simpler alternatives (Shardlow, 2013).

The task of complex word identification (CWI), recently extended to lexical complexity prediction (North et al., 2023), has been studied as a sequence annotation task, taking into account the context (sentence) in which the word appears (Gooding and Kochmar, 2019). During the 15th International Workshop on Semantic Evaluation in 2021 (Shardlow et al., 2021), this task has been further explored, prompting the reuse of the same data and evaluation metrics for comparative analysis and performance enhancement. This shared task highlighted the impact of using pre-trained language models for CWI, as well as using various fine-tuning techniques. For instance, one of the best system, JUST BLUE, (Bani Yaseen et al., 2021) combines BERT (Devlin et al., 2019) and RoBERTa (Liu et al., 2019) models. The models were separately fine-tuned to predict complexity scores, the final score being their average. Another notable example is the DeepBlueAI system (Pan et al., 2021) that integrated pre-trained language models fine-tuned with techniques like pseudo-labeling, data augmentation, stacked training models, and a multi-sample dropout layer. The data was encoded for the transformer models using the corpus type and token as the query string, with the given context as additional input. For a more comprehensive survey on lexical complexity prediction, readers may refer to North et al. (2023).

There exist various datasets for complex word identification. Some of the commonly used datasets include the Complex 2.0 dataset (Shardlow et al., 2022), CWI-2018 (Yimam et al., 2018) and CWI-2016 (Paetzold and Specia, 2016). Complex 2.0 is an improvement over Complex 1.0, as it includes more instances and additional annotations for each instance.

In the Complex 2.0 dataset, they observed various reasons why a word can be complex (Shardlow et al., 2022). These include the word being outdated or archaic, originating from another language or referring to a concept that is unusual within the reader's culture. Additionally, a word may be considered complex if it is uncommon and not frequently encountered by many individuals or if it pertains to a highly specialized subject matter. Furthermore, even if a word is commonly used, it can still be regarded as complex if it is being used with an unusual or uncommon meaning in a particular context.

2.2. ChatGPT

Given that we are currently in the era of ChatGPT, it is challenging to approach our study without including a comparison to assess the role of this task in relation to ChatGPT. Recent studies have demonstrated the promising potential of ChatGPT for various text annotation and classification tasks. Among the tasks that have piqued the interest of the scientific community is data augmentation, and one of the data augmentation techniques used with ChatGPT is paraphrasing. By rephrasing the input text in various ways, the model can generate a more diverse set of examples for training. This helps the model grasp the underlying meaning of the text rather than simply memorizing specific sentences or patterns. For example, the AugGPT framework (Dai et al., 2023) reformulates each sentence in the training samples into multiple conceptually similar but semantically different samples. The augmented samples can then be used for model training, and research results regarding the AugGPT framework clearly demonstrate that it significantly improves performance (Dai et al., 2023). Huang et al. (2023) explore the potential and limitations of ChatGPT in answering two questions: Can it effectively detect implicit hate tweets, and does it generate high-quality natural language explanations? The study shows that ChatGPT is capable of recognizing hate tweets with an 80 percent accuracy rate. Regarding the remaining 20 percent of disagreements, further examination revealed that these tweets are potentially debatable. A second evaluation with humans was conducted, and the results indicate that these individuals tend to strongly lean towards ChatGPT's classification results. As for the second question, the study demonstrates that ChatGPT generates explanations comparable to those of humans in terms of informativeness and clarity. In this paper, we investigate to what extent ChatGPT can predict word complexity in a comparative way with a model specifically trained for this task.

3. A model dedicated to complex word identification

3.1. Data

During the training and testing phase, we used the "CompLex 2.0" dataset, an improvement on "CompLex 1.0" (Shardlow et al., 2020). The corpus contains human evaluations of lexical complexity for a set of English texts using a 5-point Likert scale. The texts in the corpus were collected from sources such as Wikipedia, educational books, and newspaper articles, covering a wide variety of topics. The texts were annotated by human evaluators who assessed the lexical complexity of a target word in its context (sentence) using the Likert scale. Each instance was annotated multiple times, and the average of these annotations was taken as the score for each data instance. This score represents a continuous value between 0 and 1, once normalized. The training and the test data respectively

contain 7662 and 917 instances.

3.2. Model

In our study, we developed a neural network model to predict word complexity based on the target words and their contexts. We used transformer-based word embeddings, which are dense representations of words trained on large text datasets. We also used frequency features of the target word and the sentence to enhance the prediction quality. To capture non-linearity in the data, we added hidden layers to the model (cf. Figure 1).

3.3. Features

Our experimentation takes into account two types of inputs: one is based on word embeddings for semantic representation, while the other is based on frequency features.

3.3.1. Pre-trained language model

As indicated in the related work section, pre-trained language models have shown the best performances for Lexical Complexity Prediction.

In our case, we use the DeBERTa model which is a new architecture (Decoding-enhanced BERT with disentangled attention) that introduces relative position information and enhances BERT and RoBERTa models by using two innovative techniques (He et al., 2021). The first is the use of a disentangled attention mechanism, which allows the model to attend to specific parts of the input sequence without being influenced by irrelevant information. The second is an improved mask decoder, enabling the model to better predict masked tokens during the pre-training phase. Compared to RoBERTa-Large, DeBERTa was trained on half of the training data and consistently achieves better results on a wide range of natural language processing tasks. The inclusion of relative position information in DeBERTa's architecture enhances its ability to capture long-range dependencies and improve performance on tasks involving sequential data.

3.3.2. Frequency features

We have also chosen to calculate input features based on the Zipf frequency of words, as we believe that the complexity of a word depends not only on its semantics or morphological structure, but also on its rarity in a given corpus and the words that surround it (Chen and Meurers, 2016). Zipf's law, which describes the statistical distribution of word frequencies in a text, is often used in natural language processing to characterize the lexical richness of a text and identify keywords. To determine the frequency of word usage, we will use the wordfreq (Speer, 2022) library, which provides estimates of word usage frequency from over 40 different languages, collected from various diverse sources.

By using this measure to extract input features for our model, we hope to improve its ability to discriminate between different types of words and generalize to new and diverse data.

In our case the frequency features correspond to :

F1 : Zipf score of the target word frequency given by the following equation :

$$f(k, s, N) = \frac{1/k^s}{\sum_{n=1}^{N} 1/n^s}$$

where k is the rank of the word in the frequency table, s is the shape parameter, the value of the exponent characterizing the distribution, N is the total number of words in the corpus.

F2 : This feature calculates the average Zipf score of all the words in the sentence.

F3 : This feature measures the difference between the Zipf score of the target word and the average Zipf score of all the words in the sentence. This difference can indicate whether the target word is more or less frequent than the words in the sentence, and therefore may show some contrast between the target word and its context.

F4 : This feature counts the number of words in the sentence that have a Zipf score higher than the target word. A higher count suggests that the target word may be less common than the other words in the sentence.

F5 : This feature is a binary value that indicates whether the target word has a Zipf score less than or equal to 3, which suggests that it is very rare. If the word is very rare, it may be more difficult for readers to understand its meaning. We chose the value 3 for this feature qualitatively without a thorough statistical analysis.

To normalize frequency features between 0 and 1, we use the MinMaxScaler function. The mathematical formula can be expressed as:

$$X_{scaled} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

where X is the original feature value, X_{min} and X_{max} are the minimum and maximum values of the feature, respectively, and X_{scaled} is the scaled value.

Since pre-trained language models can process multiple input sentences, we add special tokens [CLS] and [SEP] to separate the context and the

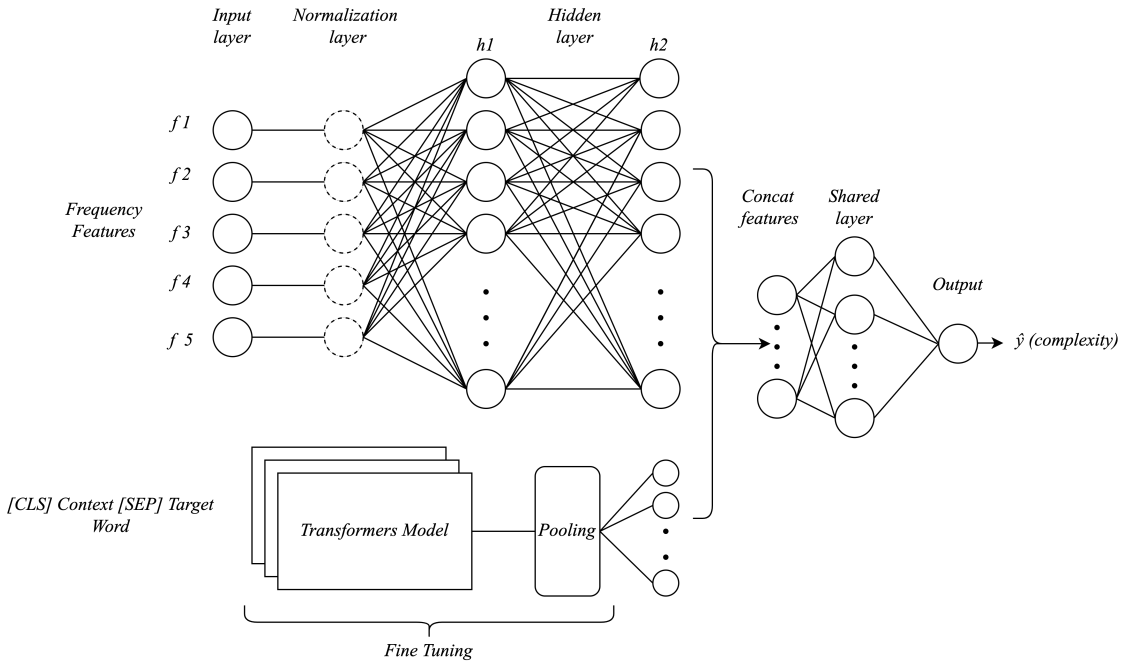

Figure 1: The overall architecture for predicting complexity scores.

target word as shown in Figure 1. We also perform pooling using the [CLS] approach. This approach involves taking the hidden state of the special token [CLS] as the representation of the entire input sequence. The [CLS] token is inserted at the beginning of the input sequence, and its hidden state is then concatenated with the hidden representation of the frequency. This concatenated representation is then fed to the final classification layer.

3.4. Evaluation

We have run our experiments for four word embedding models: bert-base, DeBERTa-base-v3, as well as their multilingual versions. To evaluate the results, we use correlation metrics such as Pearson, Spearman and R2 score. We kept the same hyperparameters for each training, including learning rate = 5e-5, batch size = 4,h1=228,h2=128, shared layer=200 and maximum sequence length = 300. We also include our baseline as well as the shared task baseline, the disparity between the two baselines lies in the fact that the one suggested by the task is a Frequency Reference generated using the log-frequency of Google Web1T (Evert, 2010) and linear regression, while ours is based solely on the frequency features listed in section 3.3.2 (F1 to F5), excluding lexical embeddings.

Table 1 shows the results of our approach using several different language models. We also include the best score obtained during the SemEval-2021 shared task (Shardlow et al., 2021). Deberta-v3-large model outperforms the other language models, as well as the best system of the shared task, the JUST BLUE system (Bani Yaseen et al., 2021).

Models	Pearson	Spearman	R2
Deberta-v3-large	**0.81**	**0.74**	**0.65**
Deberta-v3-base	0.79	0.74	0.62
The highest score in SemEval 2021 (Bani Yaseen et al., 2021)	*0.78*	-	*0.61*
mDeberta-v3-base	0.75	0.70	0.57
bert-base-cased	0.74	0.70	0.55
bert-base-multilingual	0.67	0.64	0.45
Frequency Baseline provided by (Shardlow et al., 2021)	0.52	-	0.27

Table 1: Results of different models

In order to understand the impact of pre-trained language models and frequency features in each of the models, we conducted an ablation study to measure the importance of each component.

Figure 2 is a graphical representation that shows the performance of the models for predicting lexical complexity. Each column represents a different model, with different colors to distinguish the dif-

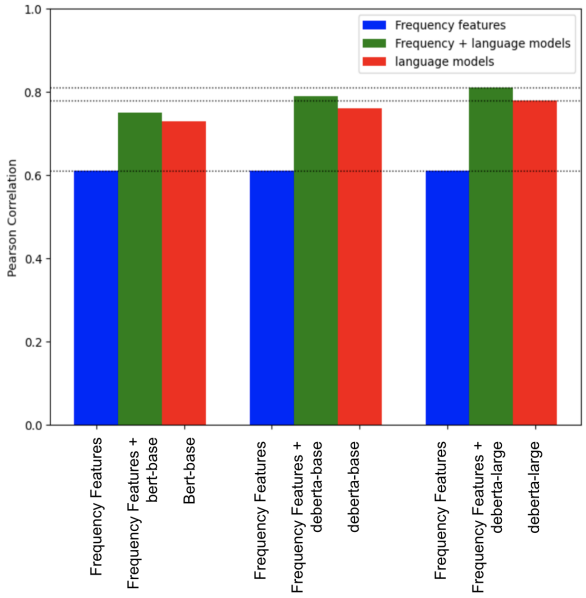

Figure 2: Comparison of Pearson Correlation values for the different models.

ferent types of models. The blue bar represents a model that only uses frequency features (We observe equivalent performances for this frequency model since it remains unchanged), the red bar represents a model based only on language models, while the green one uses both frequency features and language models. Pearson correlation scores are used to evaluate the performance of models by comparing their predictions to the actual data. The results indicate that adding frequency features to language models slightly improves their accuracy in predicting lexical complexity. In other words, models that use both frequency features and language models are better at predicting lexical complexity than models that use only one or the other. Moreover, our frequency-based model outperforms the baseline in Table 1 with a Pearson correlation score of 0.61.

4. Comparative evaluation with ChatGPT

One goal of this paper is to assess the potential of ChatGPT in measuring lexical complexity and to compare it to our current approach. This will enable us to determine whether ChatGPT can be used as a viable alternative for this specific task. This comparative evaluation is crucial for making informed decisions regarding the use of ChatGPT in our specific context. If ChatGPT proves to be more effective, it could open up new possibilities for enhancing our lexical complexity evaluation approach. Conversely, if our current approach proves to be more efficient, it could help us identify areas where ChatGPT still requires improvements or adjustments.

For this experiment, we used ChatGPT Turbo 3.5 (October 4, 2023) via the API provided by OpenAI. It is important to underline, however, a significant limitation of our experiment: ChatGPT is regularly updated, which poses reproducibility issues (Chen et al., 2023). Moreover, it is difficult to deeply interpret the results obtained since ChatGPT operates as a "black box". Therefore, we will limit ourselves to surface level observations of the results.

4.1. Comparison methodology

Our initial idea was to provide an input (i.e. a target and a context) to ChatGPT and ask it to assess the complexity of the word based on its context (the sentence in which it appears), on a scale from 0 to 1. However, the results obtained were very poor. The Pearson correlation score between human evaluations and ChatGPT is actually 0.034.

As ChatGPT was not trained specifically for the CWI task on our specific dataset, such a direct comparison is somewhat unfair. Moreover, ChatGPT does not have access to complete information on how humans have evaluated this data, making it difficult for ChatGPT to "interpret" the scale. Even with a sophisticated prompt, the task remains very complex.

We therefore need to make the evaluation more fair. Instead of evaluating the capacity of ChatGPT to predict a complexity score for a given instance (target word + context), we evaluate its capacity to compare two instances with respect to their complexity, and thus to rank a set of instances according to it. In other words, we eliminate the need to predict scores between 0 and 1, allowing ChatGPT to focus solely on evaluating the relative order of complexity among instances.

To do so, we used the bubble sort algorithm to sort a list of instances, where the comparison between two instances is performed by ChatGPT. For each pair of instances to be compared, we use the following ChatGPT prompt:

> """
>
> I give you two sentences, evaluate the complexity of the target word in quotes based on its context, and return only the sentence or the target word that is simpler to understand. The output format should be as follows:
>
> {'simplest sentence': sentence}
>
> The two sentences are:
>
> sentence 1
>
> sentence 2
>
> """

The prompt was produced through successive trial and error. The first challenge was to create a prompt that consistently generates the desired output. The temperature setting was fixed at 0 to avoid favoring creativity, which could lead to undesirable effects (Peng et al., 2023).

The bubble sort algorithm is very costly in terms of complexity. However, despite our efforts to find alternatives, pairwise comparison remains preferable. We have observed that the use of other algorithms leads to hallucinations for the ChatGPT model, increasing its non-deterministic nature, and the scores obtained are lower than those with bubble sort. It is possible that these issues will be resolved in new versions, such as GPT-4.

To make the output fully comparable with our gold data and our CWI approach described in section 3, the list of instances are sorted according to their gold and predicted scores.

For the evaluation, we use Kendall's Tau (also known as Kendall rank correlation coefficient), which is a statistical measure employed to quantify the similarity between two rankings. It assesses the correspondence or agreement between the orderings of the same set of items in two different lists. Kendall's Tau is frequently utilized when working with ordinal data, where the ranking or order of items is pertinent. We will employ this metric when comparing with ChatGPT. The Kendall's Tau formula is given by:

$$\tau = \frac{concord - discord}{\frac{1}{2}n(n-1)}$$

Let $X = (x_1, x_2, x_3, \ldots, x_n)$ and $Y = (y_1, y_2, y_3, \ldots, y_n)$ be two lists of n elements. We count the number of pairs of elements in the two lists that are in agreement ($concord$) and the number of pairs in disagreement ($discord$). A pair (i,j) is in agreement if $x_i < x_j$ and $y_i < y_j$ (or $x_i > x_j$ and $y_i > y_j$), and in disagreement otherwise. The result τ is between -1 and 1.

4.2. Evaluation

We divide our evaluation into two experiments. In the first experiment, for each target word, we sort the list of instances where the word occurs. In the second experiment, we sort lists of instances randomly picked varying the sample size.

4.2.1. Ranking contexts by target word

In the first experiment, for each target word, we sort the list of instances where the word occurs.

Among the 917 entries in our dataset, we only retain those in which the word appears in more than one sentence (at least two occurrences). This allows us to keep 685 entries. Figure 3 illustrates the number of sentences per target keyword. The data does not exhibit a uniform distribution, for instance, the number of target keywords appearing in 5 sentences is equal to 310. We have excluded entries with only one sentence. This exclusion is important because, when comparing two lists in terms of order, including entries with only one sentence would result in a perfect 100 percent match, biasing the comparison. To prevent this bias, we have chosen to exclude such cases to ensure a more equitable comparison between different words and their contexts. It is important to note that as the number of instances per target word increases, the precision of the comparison should decrease.

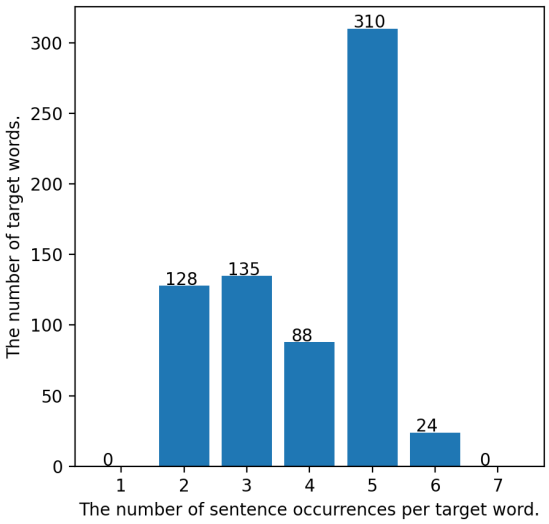

Figure 3: Distribution of the number of sentences per target word.

The evaluation per target word for our model and for ChatGPT is based on the Kendall's Tau comparing the ranking generated from our model or ChatGPT with the ones based on human annotations. The final score for a system is the average of the Kendall's Tau scores for all target words of the dataset.

As shown in Table 2, even without being specifically trained on the target dataset, ChatGPT is capable of outperforming a model that was trained on this dataset.

Models	Kendall's Tau Score
ChatGPT	**0.61**
Our approach	0.52

Table 2: Results of the scores based on the ranks.

Figure 4 shows that scores vary depending on the number of instances for each target word in the dataset. For example, words occurring in two

sentences have a higher score than those occurring in three. The curve decreases as the number of sentences per word increases. Additionally, it is noticeable that the performance curve of ChatGPT decreases at a slightly higher rate than that of our approach. Both curves appear to converge as the number of sentences increases. When the number of sentences equals 6, our model outperforms ChatGPT, however, it is important to note that this cannot be conclusive due to the low number of target words occuring in six instances (24).

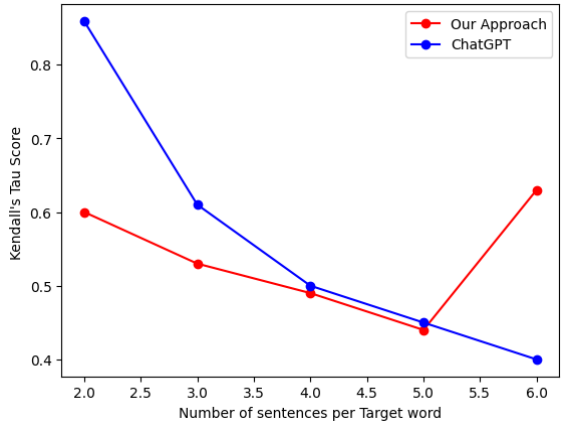

Figure 4: Scores based on the number of sentences per target word

4.2.2. Ranking sampled instances

In this part, we evaluate the ranking of instances randomly sampled in the dataset. This means that the considered lists can include contexts with various target words. The idea is to evaluate the ability of the different systems to deal with word complexity in general independently of a given target word. In our experiment, we vary the sample size to evaluate its impact on the ranking performances.

We will run the experiments for 5 sample sizes (n=4, 5, 8, 10, 20). For each sample size, we will conduct 10 separate random draws to obtain more robust measurements and enhance our assessment. Subsequently, we will calculate the mean and standard deviation of these ten draws to obtain a more precise estimation.

Table 3 shows that our model outperforms ChatGPT regardless of the sample size n. For example, with n=20, our model achieves an average Kendall score of 0.415, whereas ChatGPT scores -0.079, indicating a significant difference. Additionally, as the sample size increases, the standard deviation primarily decreases due to the reduction in variability resulting from a larger number of instances, making the mean a more precise estimate of the central tendency.

This shows that our model exhibits robustness

n	Mean		Standard Deviation	
	ChatGPT	Our Model	ChatGPT	Our Model
4	0.145	**0.491**	0.566	0.391
5	0.011	**0.152**	0.391	0.288
8	0.062	**0.376**	0.275	0.281
10	-0.078	**0.153**	0.236	0.324
20	-0.079	**0.415**	0.065	0.095

Table 3: Means and Standard Deviations for Various Sample Sizes (100% random)

when it comes to deal with different words based on their context and ranking them from the simplest to the most complex. In other words, our model consistently shows better performance compared to ChatGPT.

Considering the demonstrated capabilities of ChatGPT in various research studies for other tasks, the current question is why this task appears to be challenging for ChatGPT ? One of the hypotheses we want to verify is that ChatGPT may have difficulty distinguishing degrees of complexity when they are close.

To test this hypothesis, we will continue to randomly sample instances, but 50% of the instances will have an easy complexity level (degree <0.25), and the remaining 50% will have a difficult complexity level (degree > 0.5). This will create a clear distinction between instances based on their complexity.

n	Mean		Standard Deviation	
	ChatGPT	Our Model	ChatGPT	Our Model
4	**0.655**	0.425	0.261	0.321
5	**0.6**	0.567	0.249	0.3
8	**0.483**	0.412	0.154	0.142
10	**0.504**	0.429	0.116	0.099
20	**0.495**	0.432	0.077	0.091

Table 4: Means and Standard Deviations for Various Sample Sizes *(Random but 50 % contexts easy, 50 % very difficult)*

Table 4 shows that the scores are better with ChatGPT in this scenario. ChatGPT performs more satisfactorily when the gap in complexity levels between contexts is higher, indicating that it can differentiate between contexts having clear distinct level of complexity. The standard deviation values also become smaller when comparing with the results in Table 3, indicating that the sample is more representative, and the values are closer to the mean compared to when the data is randomly sampled 100%.

Figure 5 indicates that our approach performs better than ChatGPT when instances are 100% randomly selected. However, when we introduce a distinction in the complexity levels of the contexts (50% easy, 50 % very difficult), it makes the task easier for ChatGPT, resulting in better performance, we can also notice that this gap between the data contributes to improving the scores generated by our model as well.

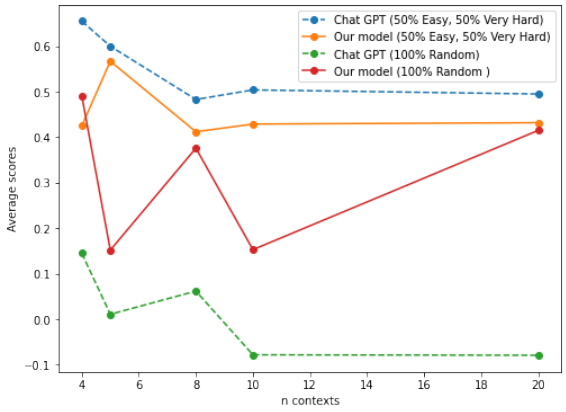

Figure 5: Average scores as a function of sample size.

5. Conclusion

In this paper, we proposed a method to improve the quality of lexical complexity prediction. This method is based on the use of pre-trained language models and frequency features. We also showed that ChatGPT has some capacity in distinguishing lexical complexity among different contexts for the same word, especially when the number of contexts is limited. In such case, it outperforms our model. However, this task becomes more challenging as the number of sentences increases, particularly when introducing different words in diverse contexts. Indeed, ChatGPT faces challenges when complexity between sentences is very similar, as demonstrated in this study when specifically evaluating extremely difficult sentences compared to simple ones, excluding medium complexity levels. This notable difference aids in the prediction for ChatGPT. In situations where contexts can be very similar in terms of complexity, it is preferable to use a model that has been trained specifically for this task. Furthermore, such a model can produce a more precise and coherent degree of complexity.

6. Limitations

Our current study focuses on assessing the lexical complexity for English only. It is possible that our model may react differently in contexts for other languages. In future work, we plan to explore multiple models for various languages. We also intend to explore the use of our system in a real-world scenario to evaluate its robustness. Another limitation of our study is that ChatGPT being regularly updated, some reproducibilty issues arise. Moreover, as ChatGPT operates as a "black box" in-depth qualitative analysis is limited.

7. Bibliographical References

David Alfter. 2021. *Exploring natural language processing for single-word and multi-word lexical complexity from a second language learner perspective*.

Tuqa Bani Yaseen, Qusai Ismail, Sarah Al-Omari, Eslam Al-Sobh, and Malak Abdullah. 2021. JUST-BLUE at SemEval-2021 task 1: Predicting lexical complexity using BERT and RoBERTa pre-trained language models. In *Proceedings of the 15th International Workshop on Semantic Evaluation (SemEval-2021)*, pages 661–666, Online. Association for Computational Linguistics.

Lingjiao Chen, Matei Zaharia, and James Zou. 2023. How is chatgpt's behavior changing over time? *arXiv preprint arXiv:2307.09009*.

Xiaobin Chen and Detmar Meurers. 2016. Characterizing text difficulty with word frequencies. pages 84–94.

Haixing Dai, Zhengliang Liu, Wenxiong Liao, Xiaoke Huang, Yihan Cao, Zihao Wu, Lin Zhao, Shaochen Xu, Wei Liu, Ninghao Liu, Sheng Li, Dajiang Zhu, Hongmin Cai, Lichao Sun, Quanzheng Li, Dinggang Shen, Tianming Liu, and Xiang Li. 2023. Auggpt: Leveraging chatgpt for text data augmentation.

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

Stefan Evert. 2010. Google web 1t 5-grams made easy (but not for the computer). In *Proceedings of the NAACL HLT 2010 Sixth Web as Corpus Workshop*, pages 32–40.

Sian Gooding and Ekaterina Kochmar. 2019. Complex word identification as a sequence labelling task. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1148–1153, Florence, Italy. Association for Computational Linguistics.

Pengcheng He, Jianfeng Gao, and Weizhu Chen. 2023. Debertav3: Improving deberta using electra-style pre-training with gradient-disentangled embedding sharing.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2021. Deberta: Decoding-enhanced bert with disentangled attention.

Fan Huang, Haewoon Kwak, and Jisun An. 2023. Is ChatGPT better than human annotators? potential and limitations of ChatGPT in explaining implicit hate speech. In *Companion Proceedings of the ACM Web Conference 2023*. ACM.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pre-training approach.

Kai North, Marcos Zampieri, and Matthew Shardlow. 2023. Lexical complexity prediction: An overview. *ACM Comput. Surv.*, 55(9).

Gustavo Paetzold and Lucia Specia. 2016. 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.

Chunguang Pan, Bingyan Song, Shengguang Wang, and Zhipeng Luo. 2021. DeepBlueAI at SemEval-2021 task 1: Lexical complexity prediction with a deep ensemble approach. In *Proceedings of the 15th International Workshop on Semantic Evaluation (SemEval-2021)*, pages 578–584, Online. Association for Computational Linguistics.

Keqin Peng, Liang Ding, Qihuang Zhong, Li Shen, Xuebo Liu, Min Zhang, Yuanxin Ouyang, and Dacheng Tao. 2023. Towards making the most of chatgpt for machine translation. *arXiv preprint arXiv:2303.13780*.

Matthew Shardlow. 2013. A comparison of techniques to automatically identify complex words. In *51st Annual Meeting of the Association for Computational Linguistics Proceedings of the Student Research Workshop*, pages 103–109, Sofia, Bulgaria. 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 Henrique Paetzold, and Marcos Zampieri. 2021. SemEval-2021 task 1: Lexical complexity prediction. In *Proceedings of the 15th International Workshop on Semantic Evaluation (SemEval-2021)*, pages 1–16, Online. Association for Computational Linguistics.

Matthew Shardlow, Richard Evans, and Marcos Zampieri. 2022. Predicting lexical complexity in english texts: the complex 2.0 dataset. *Language Resources and Evaluation*, 56(4):1153–1194.

Robyn Speer. 2022. rspeer/wordfreq: v3.0.

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.

Marcos Zampieri, Liling Tan, and Josef van Genabith. 2016. MacSaar at SemEval-2016 task 11: Zipfian and character features for Complex-Word identification. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 1001–1005, San Diego, California. Association for Computational Linguistics.