Detecting Anomalies on Texts using GPT-2

Anonymous ACL submission

Abstract

Large neural network language models trained on huge corpora of text have achieved state-of-the-art results on several natural language tasks. Using the pre-trained language model GPT-2, we propose algorithms for grammar error detection (GED). Our approach frames the GED problem as an anomaly detection problem and requires no additional training data. We leverage the next-word probability, word-embeddings from GPT-2 to detect anomalous sentences, and evaluate the result on the English learners' corpora, Lang-8, CoNLL-2014, FCE, and BEA-2019[1]. Our methods achieve a competitive area under the receiver operating characteristic (AUROC) on the English learners' corpora when detecting ungrammatical sentences. An experimental comparison of normalization methods shows that rule-driven methods are the most effective.

1 Introduction

Anomaly detection (AD) on texts is a task to identify datapoints that have distinct properties from the normality in a dataset. In the domain of grammar, this type of task can be seen as a type of Grammar Error Detection (GED) and can further be developed into Grammar Error Correction (GEC). AD on texts, GED, and GEC can be beneficial in the process of language learning and document editing, since language learners and editors often need to pick out unnatural sentences to modify. In addition to grammar, some tasks focus on detecting the contextual anomalies (Ruff et al., 2019) and words that evoke specific events in texts, namely, Event Detection (Veyseh et al., 2021). AD on texts also can be repurposed for other types of data. For instance, previous studies have applied grammar-based detection to time-series data (Senin et al., 2014, Gao, 2020). It also has been widely applied to real-world problems, such as fraud detection (Dorronsoro et al., 1997).

In this paper, we present methods of AD primarily based on the next-word probability and word-embeddings of the popular pre-trained language model, GPT-2 (Radford et al., 2019). The next-word probability can include a mix of information, specifically, grammar, meaning, and contexts. These pieces of information are all included in the next-word probability and word-embeddings. We leverage this to detect anomalous sentences. We applied four algorithms to detect anomalous sentences based on the given next-word probability and word-embeddings: Baseline, embedding-conditioned (Cosine) method, Frequency-conditioned methods, and Positional probability-conditioned method. We also reduce sentence-level algorithms to token-level ones and compare our results with those in the GED literature. Our methods do not require further training and labeling and can save a large amount of time.

To summarize, we present an anomaly detection study based on the next-word probability and word-embeddings in the pre-trained language model GPT-2. Our main contributions are (i) an examination of the performance of the popular language model, GPT-2, (ii) detecting anomalies that leverage the pre-trained next-word probability and word-embeddings in GPT-2 to save time, (iii) new methods which have the potential to develop into AD on other types of data, (iv) competitive scores compared with existing AD on grammar.

[1] Our code is available at https://github.com/limkhaiin1012/ad

2 Related Work

Early Grammar error detection works train a classifier to detect specific grammatical types of errors made by non-native English speakers (e.g. Han et al., 2006, Tajiri, 2012). Rei and Yannakoudakis (2016), and subsequent improvements based on this work (Rei, 2017, Rei et al., 2017) train GED neural networks which capture many different types of grammatical errors. These models perform a binary and a multi-class classification task based on token-level embeddings. More recent works (e.g. Bell et al., 2019, Yuan et al. 2021) started to leverage embeddings of pre-trained language models as the input to a similar bi-LSTM model that Rei (2017) proposed. These studies are focused on token-level accuracy and require another stage of training. Yasunaga et al. (2021)'s work employs the next-word probability from GPT-2, but mainly focuses on training models for GEC, a task that provides suggestions for anomalous tokens.

Previous studies on AD of texts, such as Schölkopf et al. (2001) and Manevitz and Yousef (2001) use a one-class classification of OC-SVM. Manevitz and Yousef (2007) employ a compression-decompression autoencoder on document classification. Kannan et al. (2017)'s method includes a non-negative matrix factorization and the document-terms matrices. Mahapatra et al. (2012) created a context-detection algorithm based on external corpora. This is for detecting anomalies in particular datasets and uses an LDA-based text clustering algorithm. Ruff et al. (2019) introduce a Context Vector Data Description (CVDD) method which leverages the embeddings of pre-trained models—Glove and Fasttext and BERT. The CVDD finds several compact descriptions of contexts in the training data which are optimized as context vectors. An anomaly in a CVDD model is defined as a deviation from the contextual embedding of a datapoint from these context vectors. Most of these AD studies focus on contextual anomalies and require another training stage.

Our study is similar to AD, in the sense that a model is trained to represent a normal probability distribution and an anomaly is defined as a sample with low probability. Also, it detects sentence-level plausibility, which conforms to a typical anomaly detection task if a sentence is seen as a datapoint. On the other hand, since we perform this task on the naturalness of sentences, it could be seen as a type of GED. We also include the result of token-level detection and compare it to those in previous GED literature.

3 Description of Algorithms

In this section, we introduce several sentence-level algorithms for detecting anomalies. Our algorithms include steps of hyperparameter-tuning (described in 3.5) and detection. We further reduce each of these sentence-level algorithms to token-level ones. The results are included in the next section.

3.1 Baseline Method

We define sentence alpha (α) as the formal conceptualization of the naturalness of sentences. We use the term 'naturalness' as the opposite of anomaly because next-word probability and word-embeddings can not only include grammatical information, but also include semantic and pragmatic correctness. The next-word probability in GPT-2 is predicted based on the past tokens in a sentence as shown in (1).

$$W_{<t} := (w_1, \ldots, w_{t-1}) \quad (1)$$

GPT-2 can give the next token using the conditional probability of past tokens of a current token in a sentence. This is defined in (2), where V is the vocabulary of GPT-2.

$$P(w_t|W_{<t}),\ w_t \in V \quad (2)$$

To get the sentence alpha (α) for each input token in an evaluation sentence, we extract the unnormalized next-word probabilities of all the tokens (logits) in the vocabulary and we normalize these logits using a softmax layer. The normalized probability of the next token w_t in the evaluation sentence ($p(w_t|W_{<t})$) is selected. After all the next-word probabilities in the evaluation sentence are selected, we process these next-word probabilities in the following way: if a next-word probability is lower than a threshold (T_w), it is replaced with the negative length of the evaluation sentence, as shown in (3). This manipulation is to ensure that the alpha (α) of an anomalous sentence can always be negative. If not lower than the threshold, the original next-word probability is appended to a sentence list.

$$p(w_t|W_{<t}) = \begin{cases} -len(sentence), if\ \ p(w_t|W_{<t}) < T_w \\ p(w_t|W_{<t}), otherwise \end{cases} \quad (3)$$

We keep processing until all the next-word probabilities are processed. A sentence list should now consist of the same amounts of next-word probabilities as the sentence length. Then we take the arithmetic mean of the sentence list to get the sentence alpha (α). When the sentence alpha is positive, the sentence is seen as natural; when the sentence alpha is negative, the sentence is seen as an anomaly.

In the token-level algorithm, a token is labeled as an anomaly once the next-word probability is lower than a threshold (T_w). If not lower, it is labeled as normal.

3.2 Frequency-conditioned method

The Frequency-conditioned method is also a rule-based method. We use the same method to get the next-word probabilities of a sentence, but these next-word probabilities are further conditioned on the occurrence count in the Natural Language Tool Kit (NLTK) Brown database. The Normalized Frequency (NF) is defined as in (4), where V_B is the total number of tokens in the Brown corpus and the function c is the count of the token:

$$NF = c(w_t)/c(V_B) \quad (4^2)$$

In this method, the next-word probability conditioned on NF is defined as (5). This manipulation can ensure a token gets a high or low probability not because it is frequent or rare.

$$P_{ft}(\text{next token}|NF) = P(w_t|W_{<t})/\ NF \quad (5)$$

After converting the next-word probability into a frequency-conditioned probability, we process the frequency-conditioned probability using a rule. If P_{ft} is lower than a threshold (T_w), we replace P_{ft} with a negative threshold (T_w) multiplied by the length of the sentence (see (6)). This is to ensure the alpha (α_f) of an anomalous sentence can always be negative and rejected at the end of the algorithm if the sentence is not natural.

$$P_{ft} = -T_w * len(sentence) \quad (6)$$

After all the next-word probabilities in a sentence are processed, we take the arithmetic mean of each P_{ft} in the sentence list to get the normalized frequency-adjusted sentence alpha (α_f).

$$\alpha_f = \text{mean}\ (P_{f1}, P_{f2}, \ldots P_{ft}) \quad (7)$$

The same rule is used for α_f: if α_f is lower than 0, the sentence is rejected and seen as an anomaly; otherwise, the sentence is seen as natural.

In addition to the frequency-conditioned method, we use Youden's Index to select an optimal threshold value. Youden's Index is a statistical method to obtain an optimal threshold of AUROC. An optimal threshold is defined as the largest value of the difference between the True positive rate and the False positive rate in this method. After all the next-word probabilities are converted into frequency-conditioned probabilities, no replacement rule of P_{ft} is further implemented, and the original P_{ft} in a sentence is always appended to the sentence list. We take the arithmetic mean of the sentence list to get α_f and use Youden's Index to calculate an optimal threshold value. This optimized threshold is used to get the prediction of an anomaly: if α_f is lower than the threshold, the sentence is rejected and seen as an anomaly; otherwise, the sentence is seen as natural. No hyperparameters are tuned in this method.

In the token-level algorithm, a token is marked as an anomaly once P_{ft} is lower than a threshold (T_w, rule-method). We also leverage Youden's Index to get an optimal threshold. A token is marked as an anomaly when P_{ft} is lower than the optimized threshold; otherwise, P_{ft} is seen as natural.

3.3 Word Embeddings Cosine Similarity method

The cosine method follows the same steps before getting the next-word probability ($P(w_t|W_{<t})$) in previous methods. If the next-word probability of an input token is lower than the threshold, we find the k most probable next tokens from the next-word probabilities of an input token. After extracting word vectors of these k tokens from the embedding layer of GPT-2, we compare the cosine similarity loss of word vectors of the k most probable tokens, to the word vector of the next

[2] This is also known as term frequency (tf) in tf-idf.

token (w_t) in the evaluation sentence. If the cosine similarity loss[3] of any of the k most probable tokens is lower (very similar) than a cosine-threshold (T_c), the original next-word probability of an input token is appended to the sentence list. Otherwise, a negative integer, that equals the negative of the length of the sentence is appended to the sentence list. After all the next-word probabilities in a sentence are processed, we take the arithmetic mean of the sentence list. The same procedure to get the sentence alpha with a cosine method (α_c) is used: when the sentence alpha is positive, the sentence is seen as natural; when the sentence is negative, the sentence is seen as an anomaly.

In the token-level algorithm, a token (w_t) is marked as an anomaly if its next-word probability $P(w_t|W_{<t})$ is lower than a threshold (T_w), and its cosine similarity losses of word vectors with any k most probable token are not lower than T_c.

3.4 Positional probability conditioned method

The Positional probability-conditioned method is similar to the Frequency-conditioned method. The same method is used to get the next-word probabilities in a sentence, but these next-word probabilities are further conditioned on the positional probabilities that GPT-2 provides. We first get the positional probability of a token P_t from GPT-2, which is the probability of a token being in different positions in a sentence. The positional probability P_t is defined in (8):

$$P_t = p\,(P_t|P_1 \ldots P_t) \tag{8}$$

We take the log to avoid an overflow of numbers, and define the next-word probability conditioned on a positional probability P_{Lt} as:

$$P_{Lt} = \log(P(w_t|W_{<t})/P_t) \tag{9}$$

We then process P_{Lt} in the following way. If P_{Lt} is lower than a threshold (T_w), we replace P_{Lt} with T_w multiplied by the length of the sentence. This is to ensure that the alpha (α_p) of an anomalous sentence can always be larger than the threshold and be rejected at the end of the algorithm.

$$P_{Lt} = \text{len(sentence)} * T_w \tag{10}$$

After all of the next-word probabilities are processed in a sentence, the sentence alpha conditioned on the positional probability (α_p) is defined as the arithmetic mean of all the P_{Lt} in a sentence list.

$$\alpha_p = \text{mean}\,(P_{L1}, P_{L2}, \ldots P_{Lt}) \tag{11}$$

The same procedure is used to get α_p: if α_p is lower than the threshold (T_w), the sentence is rejected and seen as an anomaly; otherwise, the sentence is seen as natural.

In addition to the above rule-based method, we also use Youden's Index to get an optimal threshold and apply the threshold to get the prediction of anomaly. Similar to the frequency-based method, after a next-word probability is converted to positional probability-conditioned probability, no replacement of P_{Lt} is further implemented, and the original P_{Lt} in a sentence is appended to a sentence list. We take the arithmetic mean of the sentence list to get sentence alpha (α_{Lt}) and use Youden's Index to calculate an optimal threshold. We use this threshold to get the prediction of an anomaly: when the sentence alpha is positive, the sentence is seen as natural; when the sentence is negative, the sentence is seen as an anomaly. No hyperparameters are tuned.

In the token-level algorithm, a token is marked as an anomaly when P_{Lt} is lower than a threshold (T_w, rule-method). We also leverage Youden's Index to get an optimal threshold. In this method, a token is marked as an anomaly when P_{Lt} is higher than the optimized threshold.

3.5 Hyperparameter tuning

We select hyperparameters for these methods and use Grid search (LaValle et al., 2004) to get optimal hyperparameters. In section 3.1, $T_w \in \{0.002, 0.0001, 0.0005, 0.00001, 0.00005, 0.000001, 0.000005\}$ for sentence-level algorithms and $T_w \in \{0.1, 0.05, 0.01, 0.005, 0.002, 0.0001, 0.0005\}$ for token-level algorithms. In 3.2, $T_w \in \{5, 10, 20, 40, 100, 200\}$. In 3.3 $T_w \in \{0.002, 0.0001, 0.0005, 0.00001, 0.00005, 0.000001, 0.000005\}$ for sentence-level algorithms and $T_w \in \{0.1, 0.05, 0.01, 0.005, 0.002, 0.0001, 0.0005\}$ for token-level

[3] Cosine similarity is often used to measure the angle distance between two word vectors. The cos(θ) is defined as

$$\cos(\theta) = A \cdot B / \| A \| \| B \|$$

, where A and B are two vectors. The numerator is the inner product of two vectors and the denominator is the product of the lengths of two vectors.

AD	Lang-8				CoNLL-14				BEA-dev			
	P	**R**	**F1**	**AUROC**	**P**	**R**	**F1**	**AUROC**	**P**	**R**	**$F_{0.5}$**	**AUROC**
Baseline	61.69	79.89	**69.62**	61.68	49.30	**99.28**	**65.89**	51.60	68.24	**96.22**	72.45	56.96
Frequency (auroc)	54.25	51.09	52.63	51.00	45.92	47.07	46.49	50.77	64.18	43.62	58.65	49.44
Frequency (rule)	50.23	48.93	49.57	51.45	49.89	34.52	40.80	52.67	66.50	71.09	67.37	52.65
Cosine	**64.58**	64.04	64.47	**61.75**	45.91	97.47	62.42	51.57	**74.59**	71.09	**73.86**	**63.30**
Position (auroc)	53.23	53.23	50.42	53.12	45.57	50.11	47.73	49.56	66.77	54.95	64.02	52.35
Position(rule)	56.87	**87.16**	68.83	61.45	**52.27**	82.57	64.01	**55.80**	70.88	75.13	71.69	59.22
Yasunaga et al. (2021)									71.4	71.3	71.4	

Table 1: sentence-level performance on Lang-8, CoNLL-2014 and BEA-dev.

GED	CoNLL-2014				BEA-dev				FCE-test			
	P	**R**	**$F_{0.5}$**	**AUROC**	**P**	**R**	**$F_{0.5}$**	**AUROC**	**P**	**R**	**$F_{0.5}$**	**AUROC**
Baseline	08.60	44.10	10.25	**54.26**	11.44	70.45	13.75	55.65	18.14	56.02	20.98	**57.41**
Frequency (auroc)	08.25	51.63	09.91	54.00	12.07	56.78	14.33	**55.96**	17.26	57.34	20.07	56.27
Frequency (rule)	09.94	52.15	09.70	54.10	10.32	84.20	12.51	52.40	15.30	82.69	18.28	54.05
Cosine	08.36	33.42	09.83	52.80	11.65	66.07	13.94	55.85	18.10	55.64	20.91	57.30
Position(auroc)	07.36	52.79	08.89	51.18	10.02	53.50	11.96	50.68	14.44	49.20	16.81	50.83
Position (rule)	07.14	**91.49**	08.75	50.56	09.77	**95.00**	11.91	49.92	14.26	**91.65**	17.15	50.90
Bell et al. (2019)	38.04	33.12	36.94		53.31	35.65	48.50		64.96	38.89	57.28	
Yuan et al. (2021)	**55.15**	39.78	**51.19**		**72.81**	46.85	**65.54**		**82.05**	50.49	**72.93**	

Table 2: token-level performance on CoNLL-2014, BEA-dev and FCE-test.

algorithms. $T_c \in \{-0.25, -0.26, -0.27, -0.28, -0.29, -0.30, -0.31, -0.32, -0.33, -0.34, -0.35\}$ for sentence-level algorithms and $T_c \in \{-0.4, -0.45, -0.5, -0.55, -0.6, -0.65, -0.7, -0.75, -0.8, -0.85\}$ for token-level algorithms, $k \in \{2, 3, 4, 5, 6, 7\}$. In 3.4, $T_w \in \{40, 50, 60, 70, 80, 90, 100\}$. We use AUROC scores to tune hyperparameters in sentence-level algorithms and F1 scores to tune hyperparameters in token-level algorithms.

4 Experiment

4.1 Pre-trained model, Datasets

We employ the pre-trained GPT-2 LMHeadModel to test the proposed algorithms. The next-word probabilities and the word embedding are extracted from this model.

We follow previous GED studies and use the four English learners' corpus of Lang-8 (Mizumoto et al., 2011), the First Certificate in English (FCE) corpus (Yannakoudakis et al., 2011), CoNLL-14 (Ng et al., 2014), and the Cambridge English Write & Improve + LOCNESS corpus released in the Building Educational Applications (BEA-19) shared task. The Lang-8 data are labeled with the number of grammatical errors in sentences. A sentence is labeled as anomalous during preprocessing when there is at least one error in the sentence. The CoNLL-14 training data, FCE, and BEA-19 are corpora with suggestions on the errors of tokens. In the sentence-level detection task, once a suggestion on an error is given in a sentence in CoNLL-14, we label the entire sentence as anomalous. We shuffle Lang-8 and CoNLL-14 and extract 10,000 evaluation sentences for the sake of processing time. The other 100 sentences are extracted for hyperparameters tuning. As for BEA-19, we extract the first 100 sentences for hyperparameter tuning and the rest of the sentences are used for evaluation.

To compare our result with Yasunaga et al. (2021), we perform model selection on 70% of the BEA-19 dev sentences (2717 sentences), and the result is included in the Appendix. The 30% held-out evaluation sentences (1186 sentences) are used for comparison.

As for CoNLL-14, FCE, and BEA-19 in the token-level detection task, we extract the first 100 sentences from each corpus for hyperparameter tuning and the rest of the sentences in each corpus are used for evaluation. A token is labeled as an anomaly once a suggestion is given.

4.2 Results

The results of the experiments are shown in Table 1 and Table 2. Overall, the results show competitive AUROC scores. This demonstrates the effectiveness of our algorithms. Specifically, in sentence-level detection (Table 1[4]), most methods show high F scores and AUROC scores. The Baseline method works the best as evidenced by the two highest F scores (69.62% and 65.89%) out of three corpora. This suggests that our method can correctly filter out anomalous sentences. It also suggests the next-word probability from GPT-2 model is effective in finding anomalous sentences from different types of language data we tested. Only a slight adjustment to next-word probabilities is required to get the optimal result.

Moreover, Frequency (rule) and Positional probability-based methods (rule) often show higher AUROC scores than methods including Youden's Index. The F1 scores based on Positional probability also largely improve with the rule-based method rather than methods including Youden's Index method. This suggests that the rule-based detection and hand-tuning hyperparameters can detect anomalous sentences more than methods including Youden's Index.

We also note that the Cosine method can give satisfactory results, as shown by its competitive precision, recall rate, and F score in some corpora. Since the Cosine method compares the cosine similarity between the k most probable tokens and the next token using word vectors in the word-embeddings, it suggests word-embeddings can be another effective resource to leverage. This again shows that the next-word probability and word-embeddings from GPT-2 provide reliable information for AD.

We compare our result on BEA-dev with Yasunaga et al. (2021). Our $F_{0.5}$ scores from the baseline, Cosine, and Position methods show competitive results on 70% of the sentences during our model selection (Table 3), so we specifically focus on these methods when evaluating on the 30% held-out sentences (Table 1). We found that these three methods consistently show better performances and all outperform Yasunaga et al. (2021)'s 71.0-71.4 $F_{0.5}$ score,

As for token-level detection (Table 2), we include precision, recall, and $F_{0.5}$ to compare our results with previous works. We found that these algorithms show decent AUROC scores and reflect a similar trend to what we have observed in the previous sentence-level detection: the Baseline method yields most of the best AUROC scores. Other methods, e.g. frequency-based method (auroc) also show a high AUROC score. Note that our methods do not require training data and labeling of training data, which are required in the literature. Also, unlike models in previous literature, we do not train a model specifically for GED tasks and GPT-2 is not specifically trained for GED as well. Therefore, even though the $F_{0.5}$ scores are disappointing, we suggest these algorithms can still be useful.

Overall, we have shown that next-word probabilities and word-embeddings from GPT-2 can successfully be harnessed to detect anomalous sentences in the corpora, but to detect anomalous tokens, further improvements should be considered.

5 Conclusion

We have performed several anomaly detection tasks based on the next-word probabilities and the word-embeddings in GPT-2. Our methods do not require further training and labeling and have yielded competitive results. This largely reduces the time for training and data-preprocessing. Importantly, the finding in this study reveals that the next-word probabilities in GPT-2 are reliable for checking sentential anomalies, and rule-driven methods can correctly detect anomalous sentences. Token-level experiments also show decent AUROC scores but require further improvement on precision and recall rates. We applied these algorithms on AD and GED but these algorithms

[4] Yasunaga et al. (2021)'s LM-Critic is the closest work but is different from ours in that we include 1186 sentences solely from BEA-dev (not from GMEG-wiki and GMEG-yahoo) but Yasunaga et al. (2021) samples 600 sentences from the three datasets. Despite the difference, we still use $F_{0.5}$ scores here just to compare our results with Yasunaga et al. (2021).

can be further adapted into a GEC task or other AD tasks on different datatypes.

References

Samuel Bell, Helen Yannakoudakis, and Marek Rei. 2019. Context is key: Grammatical error detection with contextual word representations. In *Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 103–115, Florence, Italy. Association for Computational Linguistics.

Jose R. Dorronsoro, Francisco Ginel, Carmen Sanchez, and Carlos S. Cruz. 1997. Neural fraud detection in credit card operations. *IEEE transactions on neural networks*, 84:827–34.

Yifeng Gao, Jessica Lin and Constantin Brif. 2020. Ensemble Grammar Induction For Detecting Anomalies in Time Series. In *Proceedings of the 22nd International Conference on Extending Database Technology (EDBT)*.

Na-Rae Han, Martin Chodorow, and Claudia Leacock. 2006. Detecting errors in English article usage by nonnative speakers. *Natural Language Engineering*, vol. 12, no. 02, 115–129,

Ramakrishnan Kannan, Hyenkyun Woo, Charu C Aggarwal, and Haesun Park. 2017. Outlier detection for text data. In *Proceedings of the 2017 SIAM International Conference on Data Mining*, pages 489–497.

LaValle, S. M., Branicky, M. S., & Lindemann, S. R. 2004. On the relationship between classical grid search and probabilistic roadmaps. *The International Journal of Robotics Research*, 23(7–8), 673–692.

Amogh Mahapatra, Nisheeth Srivastava, and Jaideep Srivastava. 2012. Contextual anomaly detection in text data. *Algorithms*, 5(4):469–489.

Larry M. Manevitz and Malik Yousef. 2001. One class svms for document classification. *Journal of Machine Learning Research* 2(Dec):139–154.

Larry M. Manevitz and Malik Yousef. 2007. One-class document classification via neural networks. *Neurocomputing*, 70(7-9):1466–1481.

Tomoya Mizumoto, Mamoru Komachi, Masaaki Nagata, and Yuji Matsumoto. 2011. Mining revision log of language learning SNS for automated Japanese error correction of second language learners. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, pages 147–155, Chiang Mai, Thailand. Asian Federation of Natural Language Processing.

Hwee Tou Ng, Siew Mei Wu, Ted Briscoe, Christian Hadiwinoto, Raymond Hendy Susanto, and Christopher Bryant. 2014. The CoNLL-2014 Shared Task on Grammatical Error Correction. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI blog*.

Marek Rei and Helen Yannakoudakis. 2016. Compositional sequence labeling models for error detection in learner writing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics* (Volume 1: Long Papers), pages 1181–1191, Berlin, Germany. Association for Computational Linguistics.

Marek Rei. 2017. Semi-supervised multitask learning for sequence labeling. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics* (Volume 1: Long Papers), pages 2121–2130, Vancouver, Canada. Association for Computational Linguistics.

Marek Rei, Mariano Felice, Zheng Yuan, and Ted Briscoe. 2017. Artificial error generation with machine translation and syntactic patterns. In *Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 287–292, Copenhagen, Denmark. Association for Computational Linguistics.

Lukas Ruff, Yury Zemlyanskiy, Robert Vandermeulen, Thomas Schnake, and Marius Kloft. 2019. Selfattentive, multi-context one-class classification for unsupervised anomaly detection on text. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4061– 4071, Florence, Italy. Association for Computational Linguistics.

Bernhard Schölkopf, John C. Platt, John Shawe-Taylor, Alex J. Smola, and Robert C.Williamson. 2001. Estimating the Support of a High-Dimensional Distribution. Neural computation, 13(7), 1443–1471.

Pavel Senin, Jessica Lin, Xing Wang, Tim Oates, Sunil Gandhi, Arnold P. Boedihardjo, Crystal Chen, Susan Frankenstein, and Manfred Lerner. 2014. GrammarViz 2.0: a tool for grammar-based pattern discovery in time series. In *Machine Learning and Knowledge Discovery in Databases*. Springer, 468–472.

Toshikazu Tajiri, Mamoru Komachi, and Yuji Matsumoto. 2012. Tense and aspect error correction for ESL learners using global context. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*, 198–202.

Amir Pouran Ben Veyseh, Viet Lai, Franck Dernoncourt, and Thien Huu Nguyen. 2021. Unleash GPT-2 Power for Event Detection. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing* (Volume 1: Long Papers), pages 6271–6282, Online. Association for Computational Linguistics.

Helen Yannakoudakis, Ted Briscoe, and Ben Medlock. 2011. A new dataset and method for automatically grading ESOL texts. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologie*s, pages 180–189, Portland, Oregon, USA. Association for Computational Linguistics.

Michihiro Yasunaga, Jure Leskovec, Percy Liang. 2021. LM-Critic: Language Models for Unsupervised Grammatical Error Correction. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processin*g, pages 7752–7763. Association for Computational Linguistics.

Zheng Yuan, Shiva Taslimipoor, Christopher Davis, and Christopher Bryant. 2021. Multi-Class Grammatical Error Detection for Correction: A Tale of Two Systems. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 8722–8736, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Appendix

Table 3 shows the result of 70% of the BEA-dev evaluation sentences. We found that some rule-driven methods (Baseline, Cosine, Position) show higher AUROC and $F_{0.5}$ scores, so we specifically compare our results from the three methods with Yasunaga et al. (2021).

	BEA-dev			
	P	**R**	**$F_{0.5}$**	**AUROC**
Baseline	70.22	**96.59**	**74.28**	55.09
Frequency (auroc)	68.84	47.9	63.3	51.08
Frequency (rule)	71.11	72.51	71.38	55.19
Cosine	**75.01**	70.06	73.97	**60.42**
Position (auroc)	69.66	52.85	65.5	52.16
Position(rule)	73.34	75.49	73.76	58.81

Table 3: model selection on BEA-dev