Oddballness: universal anomaly detection with language models

Filip Graliński
Adam Mickiewicz University
filipg@amu.edu.pl
Snowflake
filip.gralinski@snowflake.com

Ryszard Staruch
Adam Mickiewicz University
Center for Artificial Intelligence AMU
ryszard.staruch@amu.edu.pl

Krzysztof Jurkiewicz
Adam Mickiewicz University*

Abstract

We present a new method to detect anomalies in texts (in general: in sequences of any data), using language models, in a totally unsupervised manner. The method considers probabilities (likelihoods) generated by a language model, but instead of focusing on low-likelihood tokens, it considers a new metric defined in this paper: oddballness. Oddballness measures how "strange" a given token is according to the language model. We demonstrate in grammatical error detection tasks (a specific case of text anomaly detection) that oddballness is better than just considering low-likelihood events, if a totally unsupervised setup is assumed.

1 Introduction

Not all events with low probability are *weird* or *oddball* when they happen. For example, the probability of a specific deal in the game of bridge is extremely low ($p_b = \frac{1}{5.36 \times 10^{28}}$ for each deal). Thus, every time cards are dealt in bridge, does something incredible happen? Of course not; in fact *an* event of the very low probability p_b *must* happen (with probability 1!).

As another example, imagine two probability distributions:

1. $D_1 = \{p_1 = \frac{1}{100}, p_2 = \frac{99}{100}\}$,
2. $D_2 = \{p_1 = \frac{1}{100}, p_2 = \frac{1}{100}, \dots p_{100} = \frac{1}{100}\}$,

Intuitively, p_1 is much more oddball in D_1 than p_1 in D_2.

How, then, should we measure *oddballness*? We already know that a low probability is not enough. Let us start with basic assumptions or axioms of oddballness. Then we will define oddballness as a specific function and demonstrate its practical usage for anomaly detection when applied to probability distributions generated by language models.

* Work partially done for a thesis submitted for the MSc degree at Adam Mickiewicz University.

2 Axioms of oddballness

Let us assume a discrete probability distribution $D = (\Omega, \Pr)$, where Ω could be finite or countably infinite. From now on, for simplicity, we define D just as a multiset of probabilities:

$$D = \{p_1, p_2, p_3, \dots\} = \{\Pr(\omega_i) : \omega_i \in \Omega\}.$$

We would like to define an oddballness measure[1] for an outcome (elementary event) of given probability p_i within a distribution D:

$$\xi_D(p_i), \xi_D : D \to [0, 1]$$

Note that oddballness, unlike, for instance, entropy, does not work just on a probability distribution. It is defined for a probability distribution *and* an event (or rather its probability) within that distribution.

Let us define some common-sense axioms for oddballness:

(O0) $\xi_D(p_i) \in [0, 1]$ – let us assume our measure is from 0 to 1,

(O1) $\xi_D(0) = 1$ – if an impossible event happens, that's pretty oddball!

(O2) for any distribution $\xi_D(\max\{p_i\}) = 0$ – the most likely outcome is not oddball at all,

(O3) $p_i = p_j \to \xi_D(p_i) = \xi_D(p_j)$ – all we know is a distribution, hence two outcomes of the same probability must have the same oddballness (within the same distribution),

(O4) $p_i < p_j \to \xi_D(p_i) \geq \xi_D(p_j)$ – if some outcome is less likely than another outcome, it cannot be less oddball,

[1] *Measure* understood informally, not as defined in measure theory.

(O5) (continuity) for any distribution $D = \{p_1, p_2, p_3, \ldots\}$, the function $f(x) = \xi_{D_x}(x)$, where $D_x = \{x, p_2 \times \frac{1-x}{1-p_1}, \ldots, p_i \times \frac{1-x}{1-p_1}, \ldots\}$, is continuous – if we change the probabilities by a small amount, the oddballness should not change much.

Note that (O2) implies the following two facts:

(F1) $p_i > 0.5 \rightarrow \xi_D(p_i) = 0$ – something that is more likely than 50% is not oddball at all,

(F2) for any distribution $D = \{p_1 = \frac{1}{N}, \ldots, p_N = \frac{1}{N}\}$, $\xi_D(p_i) = 0$ – as in the bridge example.

3 Oddballness measure

Let us define a measure that fulfills (O0)–(O5). First, let us define an auxiliary function:

$$x^+ = \max(0, x).$$

(In other words, this is the ReLU activation function.)

Now, let us assume a probability distribution $D = \{p_1, p_2, p_3, \ldots\}$. Let us define the following oddballness measure:

$$\xi_D(p_i) = \sum_j (p_j - p_i)^+. \qquad (1)$$

This measure satisfies the axioms (O0)–(O5).

Let us check this measure for the distributions D_1 and D_2 given as examples above:

- $\xi_{D_1}(p_1) = 0.98$,
- $\xi_{D_1}(p_2) = 0$,
- $\xi_{D_2}(p_i) = 0$,

Consider another example: $D_3 = \{p_1 = 0.7, p_2 = 0.25, p_3 = 0.05\}$, then: $\xi_{D_3}(p_1) = 0$, $\xi_{D_3}(p_2) = (0.7 - 0.25)^+ + (0.25 - 0.25)^+ + (0.05 - 0.25)^+ = 0.45$, $\xi_{D_3}(p_3) = 0.85$.

More generally, the following family of functions fulfills the axioms of oddballness as well:

$$\xi_D(p_i) = \frac{\sum_j g((p_j - p_i)^+)}{\sum_j g(p_j)},$$

where g is any monotonic and continuous function for which $g(0) = 0$ and $g(1) = 1$. The more general form of the oddballness measure simplifies just to (1), when the identity function $g(x) = x$ is assumed (although, for instance, x^2 or x^3 can also be used). From now on, we will continue to use the simpler form (1).

Implementing the oddballness measure for LLMs in PyTorch is simple and can be done in a single line of code. In Listing 1 there are two functions – one that returns probability values for each token in the given text, and a second one that returns oddballness values for each token in the given text. We use these functions in the experiments in Section 7.

```
def get_probability_for_decoder(text):
    model_input = tokenizer(text,
        return_tensors='pt')
    tokens_tensor = model_input.
        input_ids[0]

    with torch.no_grad():
        outputs = model(**model_input)
    output = torch.softmax(outputs
        [0][0], dim=1)

    tokens_tensor = tokens_tensor[1:]
    output = output[:-1]

    probabilities = output[torch.arange(
        output.shape[0]), tokens_tensor]

    return probabilities


def get_oddballness_for_decoder(text):
    model_input = tokenizer(text,
        return_tensors='pt')
    tokens_tensor = model_input.
        input_ids[0]

    with torch.no_grad():
        outputs = model(**model_input)
    output = torch.softmax(outputs
        [0][0], dim=1)

    tokens_tensor = tokens_tensor[1:]
    output = output[:-1]

    probabilities = output[torch.arange(
        output.shape[0]), tokens_tensor]
    oddballness = torch.sum(relu(output
        - probabilities[:, None]), dim
        =1)

    return oddballness
```

Listing 1: Python functions that compute probability and oddballness values for a language model in PyTorch.

4 Oddballness as a complement of probability of probability

Interestingly, oddballness can be interpreted as the complement of the probability *of a probability*. By the probability of a probability p_i with respect to the distribution D (denoted by $\pi_D(p_i)$), we mean:

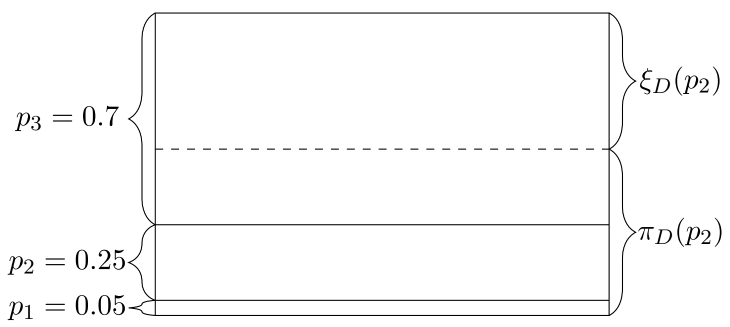


Figure 1: Illustration of oddballness ξ_D and "probability of probability" (π_D) for the event ω_2 of probablity $p_2 = 0.25$ for $D_3 = \{p_1 = 0.7, p_2 = 0.25, p_3 = 0.05\}$.

- the probability that *some* event of probability p_i (not necessarily ω_i) or lower occurs
- …with the additional assumption that each event ω_j with probability $p_j > p_i$ contains a "subevent" of probability p_i, hence for each such event we sum in p_i as well.

It can be shown that

$$\pi_D(p_i) = 1 - \xi_D(p_i).$$

Intuitively, an event is oddball if the probability of any event happening with a similar probability is low. See Figure 1 for an illustration of the relation between oddballness and probability of probability.

5 Connection to entropy

The concept of oddballness introduced in this paper is closely related to entropy in a probability distribution. Entropy is a measure of the uncertainty, the randomness, of a probability distribution. Oddballness can be seen as a way to quantify the "strangeness" of an event, which is really just an expression of outliers in the entropy of a distribution. In particular, the oddballness measure is defined as a normalized version of the surprisal of an event ($\log(1/p)$), which is a measure of the information content of an event.

6 What is the practical use?

The oddballness measure can be used to detect anomalies or errors, for example in a text, assuming that we have a good language model. The language model will give a probability distribution for any word in a text; some words will be given higher probability (likelihood), some lower. We could mark words with low probability as suspicious, but sometimes a low-probability event *must* occur. For instance, the distribution for the word gap in the sentence:

I was born in …, a small village

should be (for a good language model[2]) composed of a large number of names, each with a relatively low probability. Hence, like in the bridge example, we should not be surprised to see a low-probability event. On the other hand, in the sentence:

I was born in New … City

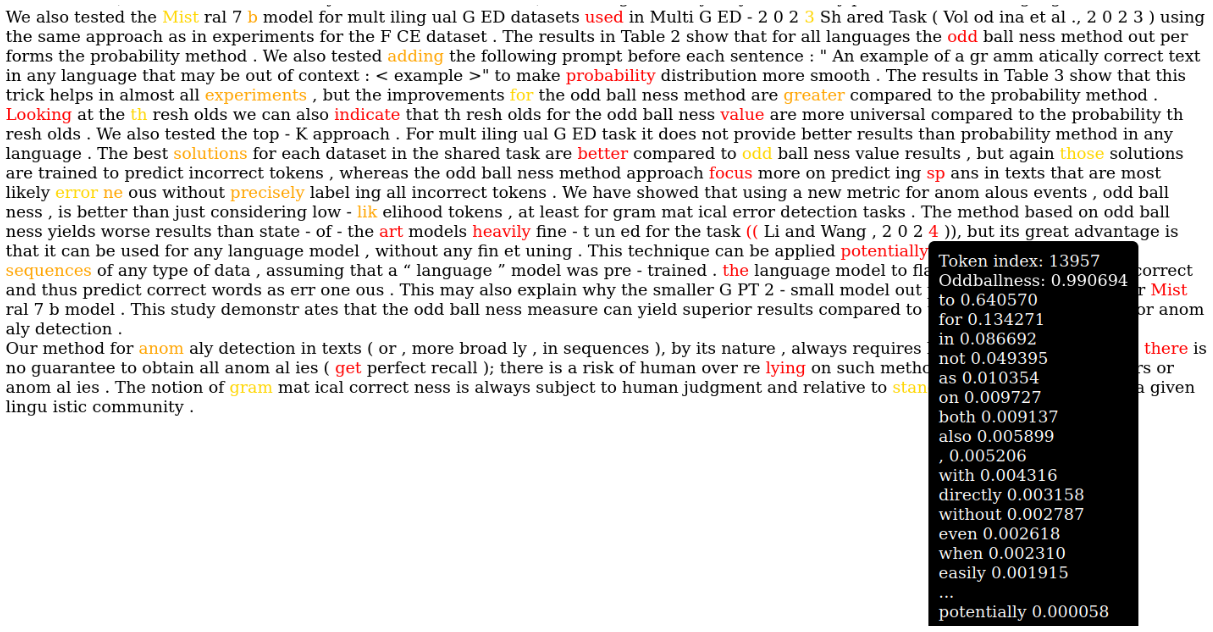
any word other than *York* is pretty unlikely (and *oddball*). Therefore, rather than probability, the oddballness should be used – words with oddballness exceeding some threshold should be marked as suspicious; they are potential mistakes or anomalies to be checked by humans. In this way, we could devise a grammar checking/proofreading system that is not trained or fine-tuned in a supervised manner for the specific task of error detection.

The notion of oddballness might not have been that useful in the world before good language models, when usually only static discrete distributions were assumed. Language models, even within the same text, can generate vastly different types of probability distributions for each position:

- sometimes the model is almost certain and almost all probability will be assigned to one token,
- sometimes the model will predict a group of possible tokens plus a long tail of less likely tokens,
- and sometimes the model is uncertain and the entropy is high.

See Figure 2 for an example of how oddballness can work for detecting 'suspicious' passages in a

[2]For this example, an encoder-only model trained on the masked language task should be assumed, for instance RoBERTa (Liu et al., 2019).



```
We also tested the Mist ral 7 b model for mult iling ual G ED datasets used in Multi G ED - 2 0 2 3 Sh ared Task ( Vol od ina et al ., 2 0 2 3 ) using
the same approach as in experiments for the F CE dataset . The results in Table 2 show that for all languages the odd ball ness method out per
forms the probability method . We also tested adding the following prompt before each sentence : " An example of a gr amm atically correct text
in any language that may be out of context : < example >" to make probability distribution more smooth . The results in Table 3 show that this
trick helps in almost all experiments , but the improvements for the odd ball ness method are greater compared to the probability method .
Looking at the th resh olds we can also indicate that th resh olds for the odd ball ness value are more universal compared to the probability th
resh olds . We also tested the top - K approach . For mult iling ual G ED task it does not provide better results than probability method in any
language . The best solutions for each dataset in the shared task are better compared to odd ball ness value results , but again those solutions
are trained to predict incorrect tokens , whereas the odd ball ness method approach focus more on predict ing sp ans in texts that are most
likely error ne ous without precisely label ing all incorrect tokens . We have showed that using a new metric for anom alous events , odd ball
ness , is better than just considering low - lik elihood tokens , at least for gram mat ical error detection tasks . The method based on odd ball
ness yields worse results than state - of - the art models heavily fine - t un ed for the task (( Li and Wang , 2 0 2 4 )), but its great advantage is
that it can be used for any language model , without any fin et uning . This technique can be applied potentially
sequences of any type of data , assuming that a " language " model was pre - trained . the language model to fla                  correct
and thus predict correct words as err one ous . This may also explain why the smaller G PT 2 - small model out                   r Mist
ral 7 b model . This study demonstr ates that the odd ball ness measure can yield superior results compared to                   or anom
aly detection .
Our method for anom aly detection in texts ( or , more broad ly , in sequences ), by its nature , always requires                there is
no guarantee to obtain all anom al ies ( get perfect recall ); there is a risk of human over re lying on such metho              rs or
anom al ies . The notion of gram mat ical correct ness is always subject to human judgment and relative to stan                  a given
lingu istic community .
```

```
Token index: 13957
Oddballness: 0.990694
to 0.640570
for 0.134271
in 0.086692
not 0.049395
as 0.010354
on 0.009727
both 0.009137
also 0.005899
, 0.005206
with 0.004316
directly 0.003158
without 0.002787
even 0.002618
when 0.002310
easily 0.001915
...
potentially 0.000058
```

Figure 2: Visualization of oddballness on a fragment from one of the earlier versions of this paper. Yellow, orange and red represent oddballness ranges, respectively: (0.88, 0.91), (0.91, 0.94), (0.94, 1.0). For the occurrence of the word *potentially* the probabilities and oddballness are given.

text. Some of the words with high oddballness are indeed grammatical errors or cases of awkwardness; for instance, *can be applied potentially* instead of *can be potentially applied*.[3]

In this paper, we focus on applying oddballness to grammatical error detection (see Section 7). However, some related (but not identical) ideas have been proposed in the field of log anomaly detection, as log sequences can be viewed as a modality similar to natural language. LogBERT by Guo et al. (2021) was trained, in a semi-supervised way, on log sequences. During anomaly detection, some tokens are masked, and the probability distribution is obtained from LogBERT for each of them. If the probability of the actual token is not one of the K highest-likelihood tokens (K is a hyperparameter), the token is considered anomalous (we will refer to this method later as top$K$). LogGPT by Han et al. (2023) is a similar idea, but applied to a decoder-only GPT-like architecture, rather than an encoder-only Transformer; however, the same approach of considering top$K$ prediction is still taken for the detection of anomalies, although the model is also fine-tuned specifically for anomaly detection.

In general, there is a large body of literature on anomaly or outlier detection; see, for instance, Schölkopf et al. (2001), Breunig et al. (2000), Liu et al. (2008). Oddballness is different as it considers only probabilities from a language model (or any other statistical model) rather than any intrinsic feature of the events in question.

7 Experiments with error detection

Table 1 presents the results obtained on the FCE dataset (Yannakoudakis et al., 2011). In each case, using the oddballness value as the threshold gives better results than using the probability value. All thresholds were adjusted to maximize the $F_{0.5}$ score on the development set. The maximum oddballness value from the GPT2-XL and RoBERTa Large (Liu et al., 2019) models produced the best $F_{0.5}$ score on the test set. The result is slightly better than the BiLSTM model by Rei and Yannakoudakis (2016), which was trained specifically to detect errors in texts, while GPT2-XL and RoBERTa Large are models which were trained, in a self-supervised manner, on the masked token prediction task. Although results based on the oddballness value are not competitive with state-of-the-art solutions, it should be noted that the oddballness technique *does not involve any task-specific fine-tuning*, except for adjusting a single hyperparameter: the oddballness threshold.

Furthermore, the texts were written by CEFR-B-

[3]Google Scholar reported 367 results for the query "`can be applied potentially`" and as many as 15,600 for "`can be potentially applied`". Anecdotally, this was not detected by the Writefull grammar checking tool.

Model	Method	Threshold	Dev $F_{0.5}$	Test $F_{0.5}$
Unsupervised methods				
GPT2-small	Probability	0.0002	35.00	37.74
GPT2-small	Oddballness	0.84	37.27	39.19
GPT2-XL	Probabilisty	0.0001	36.00	38.86
GPT2-XL	Oddballness	0.85	38.17	40.52
Yi-6b	Probability	0.0005	34.38	37.35
Yi-6b	Oddballness	0.85	36.77	39.83
Mistral 7B	Probability	0.0003	33.68	36.86
Mistral 7B	Oddballness	0.89	35.04	38.00
RoBERTa Base	Probability	0.005	32.63	33.62
RoBERTa Base	Oddballness	0.91	33.08	34.86
RoBERTa Large	Probability	0.014	32.74	33.39
RoBERTa Large	Oddballness	0.84	34.33	35.78
min(GPT2-XL, RoBERTa Large)	Probability	0.0001	36.88	39.31
max(GPT2-XL, RoBERTa Large)	Oddballness	0.89	**40.32**	**43.15**
Supervised methods				
(Rei and Yannakoudakis, 2016)	Bi-LSTM	-	46.00	41.10
(Bell et al., 2019)	BERT-base	-	-	57.28
(Kaneko and Komachi, 2019)	MHMLA	-		61.65
(Yuan et al., 2021)	ELECTRA	-	-	**72.93**

Table 1: Results for the Grammatical Errror Detection FCE Dataset. Thresholds tuned with the development set.

level students, who may not be fully proficient in the language. This could cause the language model to flag non-fluent words as incorrect and thus predict correct words as erroneous. This may also explain why the smaller GPT2-small model outperforms the much larger Mistral 7B model. This study demonstrates that the oddballness measure can yield superior results compared with the use of probability values for anomaly detection.

We also tested the Mistral 7B model (Jiang et al., 2023) for the multilingual GED datasets used in the MultiGED-2023 Shared Task (Volodina et al., 2023) using the same approach as in the experiments for the FCE dataset.[4] The results in Table 2 show that for all languages the oddballness method outperforms the probability method. We also tried adding the following prompt before each sentence: "An example of a grammatically correct text in any language that may be out of context: <example>" to make the probability distributions smoother. The results in Table 3 show that this trick helps in almost all of the experiments, but the improvements for the oddballness method are greater than for the probability method. The top$K$ approach was also tested. For the multilingual GED task it does not provide better results than the probability method in any language.

Looking at the thresholds, we can also observe that the thresholds for the oddballness value are more universal than the probability thresholds.

The best solutions for each dataset in the shared task are better than the oddballness value results, but again those solutions are trained to predict incorrect tokens, whereas the oddballness method approach focuses more on predicting spans in texts that are most likely erroneous without precisely labeling all incorrect tokens.

8 Conclusions

We have shown that using a new metric for anomalous events, oddballness, is better than just considering low-likelihood tokens, at least for grammatical error detection tasks. The method based on oddballness yields worse results than state-of-the-art models that have been heavily fine-tuned for the task (Li and Wang, 2024), but its great advantage is that it can be used for any language model, without any fine-tuning. This technique can be potentially

[4]The source code for the experiments is available here: `https://github.com/richardxoldman/oddballness`.

Language	**Method**	**Threshold**	**Dev $F_{0.5}$**	**Test $F_{0.5}$**
Czech	TopK	30	41.17	38.56
Czech	Probability	0.002	44.32	41.87
Czech	Oddballness	0.84	49.16	46.58
German	TopK	86	30.53	28.67
German	Probability	0.001	32.53	31.32
German	Oddballness	0.89	37.62	36.67
English – FCE	TopK	310	29.18	30.67
English – FCE	Probability	0.00009	32.47	34.39
English – FCE	Oddballness	0.90	35.21	35.91
English – REALEC	TopK	390	27.66	28.17
English – REALEC	Probability	0.00006	31.03	31.12
English – REALEC	Oddballness	0.94	32.93	32.69
Italian	TopK	18	22.76	24.53
Italian	Probability	0.003	23.71	25.90
Italian	Oddballness	0.80	27.33	29.77
Swedish	TopK	34	35.16	33.99
Swedish	Probability	0.003	37.41	36.08
Swedish	Oddballness	0.78	40.18	38.85

Table 2: Results for the Mistral 7B model on the MultiGED-2023 Shared Task dataset.

Language	**Method**	**Threshold**	**Dev $F_{0.5}$**	**Test $F_{0.5}$**
Czech	TopK	18	41.21	39.53
Czech	Probability	0.003	44.24	42.75
Czech	Oddballness	0.84	49.77	47.76
German	TopK	80	31.78	30.41
German	Probability	0.0009	34.48	33.04
German	Oddballness	0.89	39.40	39.27
English – FCE	TopK	140	29.82	31.82
English – FCE	Probability	0.0003	32.85	35.02
English – FCE	Oddballness	0.90	35.96	36.35
English – REALEC	TopK	520	27.74	27.46
English – REALEC	Probability	0.0001	30.42	29.95
English – REALEC	Oddballness	0.92	32.84	32.43
Italian	TopK	30	23.13	25.19
Italian	Probability	0.0008	25.39	26.48
Italian	Oddballness	0.92	31.43	32.37
Swedish	TopK	50	36.98	34.99
Swedish	Probability	0.002	39.20	38.24
Swedish	Oddballness	0.84	43.55	41.84

Table 3: Results for the Mistral 7B model on the MultiGED-2023 Shared Task dataset with an additional prompt.

applied to anomaly detection in sequences of any type of data, assuming that a "language" model was pre-trained on such data.

9 Limitations

Our method for anomaly detection in texts (or, more broadly, in sequences), by its nature, always requires human supervision. Also, there is no guarantee of obtaining all anomalies (achieving perfect recall); there is a risk of humans overrelying on such methods while looking for errors or anomalies.

Our study does not cover human evaluation, which could provide more accurate results. The texts used in the MultiGED-2023 Shared Task were annotated on the sentence level. It would be desirable to test the oddballness method on texts that are split on the document level, so as to take longer contexts into account (this affects the probability distributions created by language models).

The notion of grammatical correctness is always subject to human judgment and is relative to standards established within a given linguistic community.

References

Samuel J. Bell, Helen Yannakoudakis, and Marek Rei. 2019. Context is key: Grammatical error detection with contextual word representations. *CoRR*, abs/1906.06593.

Markus M Breunig, Hans-Peter Kriegel, Raymond T Ng, and Jörg Sander. 2000. LOF: identifying density-based local outliers. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, pages 93–104.

Haixuan Guo, Shuhan Yuan, and Xintao Wu. 2021. LogBERT: Log anomaly detection via BERT. In *2021 international joint conference on neural networks (IJCNN)*, pages 1–8. IEEE.

Xiao Han, Shuhan Yuan, and Mohamed Trabelsi. 2023. LogGPT: Log anomaly detection via GPT. In *2023 IEEE International Conference on Big Data (BigData)*, pages 1117–1122. IEEE.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b. *Preprint*, arXiv:2310.06825.

Masahiro Kaneko and Mamoru Komachi. 2019. Multi-head multi-layer attention to deep language representations for grammatical error detection. *CoRR*, abs/1904.07334.

Wei Li and Houfeng Wang. 2024. Detection-correction structure via general language model for grammatical error correction. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1748–1763, Bangkok, Thailand. Association for Computational Linguistics.

Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. 2008. Isolation forest. In *2008 Eighth IEEE International Conference on Data Mining*, pages 413–422. IEEE.

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 pretraining approach. *Preprint*, arXiv:1907.11692.

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.

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.

Elena Volodina, Christopher Bryant, Andrew Caines, Orphée De Clercq, Jennifer-Carmen Frey, Elizaveta Ershova, Alexandr Rosen, and Olga Vinogradova. 2023. MultiGED-2023 shared task at NLP4CALL: Multilingual grammatical error detection. In *Proceedings of the 12th Workshop on NLP for Computer Assisted Language Learning*, pages 1–16, Tórshavn, Faroe Islands. LiU Electronic Press.

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 Technologies*, pages 180–189, Portland, Oregon, USA. 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.