GermDetect
Verb Placement Error Detection Datasets for Learners of Germanic Languages

Noah-Manuel Michael
Kiel University, Germany
Leibniz Institute for Science and Mathematics Education, Kiel, Germany
Linköping University, Sweden
michael@ipn.uni-kiel.de

Andrea Horbach
Kiel University, Germany
Leibniz Institute for Science and Mathematics Education, Kiel, Germany
horbach@ipn.uni-kiel.de

Abstract

Correct verb placement is difficult to acquire for second-language (L2) learners of Germanic languages. However, word order errors and, consequently, verb placement errors, are heavily underrepresented in benchmark datasets of NLP tasks such as grammatical error detection (GED)/correction (GEC) and linguistic acceptability assessment (LA). If they are present, they are most often naively introduced, or classification occurs at the sentence level, preventing the precise identification of individual errors and the provision of appropriate feedback to learners. To remedy this, we present **GermDetect**: Universal Dependencies-based (UD), linguistically informed verb placement error **detect**ion datasets for learners of **Germ**anic languages, designed as a token classification task. As our datasets are UD-based, we are able to provide them in most major Germanic languages: Afrikaans, German, Dutch, Faroese, Icelandic, Danish, Norwegian (Bokmål and Nynorsk), and Swedish. We train multilingual BERT (mBERT) models on GermDetect and show that linguistically informed, UD-based error induction results in more effective models for verb placement error detection than models trained on naively introduced errors. Finally, we conduct ablation studies on multilingual training and find that lower-resource languages benefit from the inclusion of structurally related languages in training.

1 Introduction

Correct verb placement is difficult to acquire for L2 learners of Germanic languages. This is due to the placement depending on different factors such as finiteness and the clause type in which the verbs occur. Example (1) illustrates a Dutch sentence consisting of a single main clause.

(1)

Hij	***heeft***	*een*	*hond*	***gekocht***
he	has	a	dog	bought
S	V2	O	O	V

‘He has bought a dog.’

Two characteristics can be observed here: The finite verb *heeft* is placed in the second position (V2) and the non-finite, participle verb *gekocht* is placed clause-finally. Example (2), consisting of a main and a subordinate clause, illustrates how the verb placement changes: Here, the finite *weet* occupies the V2 position in the main clause, but both the finite *heeft* and non-finite *gekocht* are placed clause-finally in the subordinate clause, following subject-object-verb (SOV) word order.

(2)

Ik	***weet***	*dat*	*hij*	*een*	*hond*	***heeft***	***gekocht***
I	know	that	he	a	dog	has	bought
S	V2	_	S	O	O	V	V

‘I know that he has bought a dog.’

These and other context-dependent changes in verb placement in Germanic languages are not trivial for L2 learners. Therefore, being able to provide accurate feedback to learners about their verb placement is crucial.

However, word order errors are heavily underrepresented in both GED/GEC shared task datasets and LA datasets. If they are present, they are often introduced naively, which means they do not represent the kinds of errors L2 learners are likely to make. This leaves the capability of GED systems to detect naturalistic word order errors underexplored. Verb placement errors, as a subset of word order errors, are even more poorly represented, which, in the context of Germanic languages, is particularly critical. Additionally, LA datasets are often formulated as sentence-level binary classification or pair-wise ranking tasks. This does not allow for locating errors within a sentence and therefore offers

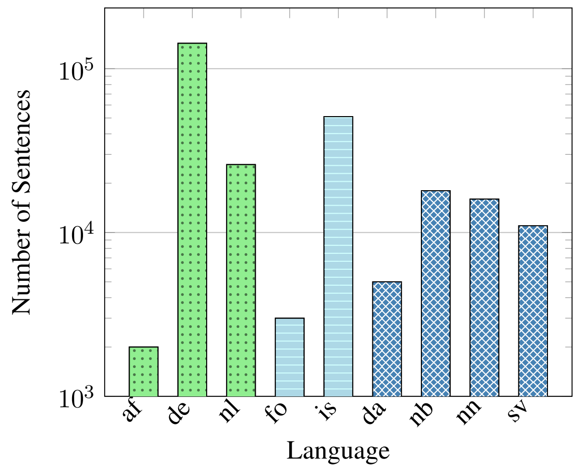


Figure 1: Number of sentences per language in GermDetect (log scale); language codes follow ISO 639-1.

little usability for providing feedback to learners.

To address this gap, we present GermDetect: UD-based, linguistically informed verb placement error detection datasets for learners of Germanic languages that can serve as a new benchmark to test GED systems' capabilities in detecting naturalistic verb placement errors. The token-level classification design allows individual errors in verb placement to be located, and the datasets are available in most major Germanic languages:[1] Afrikaans, German, Dutch, Faroese, Icelandic, Danish, Norwegian (Bokmål and Nynorsk), and Swedish. Figure 1 presents a brief overview of the magnitude of our dataset, with a more detailed breakdown in Appendix A. We make the datasets and the code available at `https://github.com/noahmanu/gerlangmod`.

In the following sections, we provide a brief introduction to related work, followed by a description of our dataset creation algorithm. We then present the results of different mBERT configurations on our new benchmark.

2 Related Work

In this section, we introduce verb placement rules in Germanic languages, briefly talk about the frequency of verb placement errors in learner corpora, present relevant GED/GEC/LA datasets and their shortcomings with regard to the evaluation of word order errors, and survey popular GED/GEC tools' capabilities in detecting verb placement errors.

[1]Excluding English as English is the only modern Germanic language that does not follow V2.

2.1 Verb Placement Rules in Germanic Languages

Correct verb placement is challenging to acquire for L2 learners of Germanic languages (Orgassa, 2009; Schimke and Dimroth, 2018; Westergaard and Lohndal, 2019; Angantýsson, 2021). This is due to the placement depending on different factors such as finiteness and the clause type in which the verbs occur.

All Germanic languages covered in GermDetect follow V2 in main clauses, which means that the finite verb always occupies the second position. Example (1) illustrates how in West Germanic languages, SOV is the default syntax pattern for all other verbs, i.e., non-finite verbs are placed clause-finally in main clauses. In subordinate clauses, where V2 does not hold, all verbs follow SOV and are therefore found at the end of the phrase. Example (3) demonstrates by means of a Swedish sentence that, in North Germanic languages, all verbs follow SVO in main clauses.

(3)

Han	***har***	***köpt***	*en*	*hund.*
he	has	bought	a	dog
S	V2	V	O	O

This masks the V2 constraint, as the surface word order in main clauses often resembles that of canonical SVO languages. Example (4) briefly presents how, when an element other than the subject takes the clause-initial position, V2 holds while the rest of the main clause follows SVO.

(4)

Kanske	***har***	*han*	***köpt***	*en*	*hund.*
maybe	has	he	bought	a	dog
_	V2	S	V	O	O

'Maybe he has bought a dog.'

Additionally, all GermDetect languages form polar questions by inversion, placing the finite verb in the clause-initial position.[2] Examples (5) and (6) illustrate this structure for both German and Faroese. Non-finite verbs follow the respective default syntax patterns.

(5)

Hat	*er*	*einen*	*Hund*	***gekauft****?*
has	he	a	dog	bought
V1	S	O	O	V

'Has he bought a dog?'

[2]In analogy to V2, we call this position V1.

Languages	Main–Finite	Main–Non-Finite	Subordinate	PolarQ–Finite	PolarQ–Non-Finite
af, de, nl	V2	SOV	SOV	V1	SOV
fo, is, da, nb, nn, sv	V2	SVO	SVO	V1	SVO

Table 1: Overview of the most unmarked syntax patterns in the languages covered by GermDetect; distinction between finite and non-finite verbs in main clauses and polar questions (PolarQ), and subordinate clauses.

Dataset	Task	% WOE	Languages
Dale and Kilgarriff (2011)	GEC	< 7.5	en
Ng et al. (2013)	GEC	0.0	en
Ng et al. (2014)	GEC	2.4	en
Napoles et al. (2017)	GEC	N/A	en
Bryant et al. (2019)	GEC	1.6	en
Warstadt et al. (2019)	LA	N/A	en
Warstadt et al. (2020)	LA	19.4	en
Nielsen (2023)	LA	50.0	da, fo, is, nb, nn, sv
Volodina et al. (2023)	GED	N/A	cs, de, en, it, sv
Masciolini et al. (2025)	GEC	N/A	cs, de, el, en, et, is, it, lv, ru, sl, sv, uk

Table 2: Percentage of word order errors (WOE) in different benchmark datasets; languages covered by GermDetect underlined.

(6)

Hevur	*hann*	***keypt***	*ein*	*hund?*
has	he	bought	a	dog
V1	S	V	O	O

Table 1 summarizes the most unmarked syntax pattern for both language groups, i.e., West Germanic and North Germanic. With verb placement being this complex in Germanic languages, learners are prone to make errors when trying to acquire the correct syntax patterns. However, learners typically do not make random errors in verb placement. Instead, the errors they make are often influenced by their previous language background. Therefore, certain error types are less likely to occur than others. Example (7) shows an unlikely example of an error where the learner splits the noun phrase *een hond* by misplacing a verb between the elements of this constituent. This is unlikely because nominal constituents are generally joint units, so this would most likely be considered an error in any of the languages the learner knows.

(7)

**Ik*	***weet***	*dat*	*hij*	*een*	***heeft***	*hond.*
I	know	that	he	a	has	dog

Example (8), in contrast, illustrates a more likely error that could be produced by a first-language English speaker.

(8)

**Ik*	***weet***	*dat*	*hij*	***heeft***	*een*	*hond.*
I	know	that	he	has	a	dog

It is evident that verb placement presents itself as a very complex system of rules for L2 learners of Germanic languages to acquire. Next, we briefly show how this is reflected in real learner corpora.

2.2 Verb Placement Errors in Germanic Learner Corpora

In the German part of the MERLIN corpus (Boyd et al., 2014),[3] which comprises texts from all CEFR levels but is especially rich in texts from levels A2–C1,[4] 34% of errors are annotated as “movement errors”, i.e., errors corrected by placing a token in a different position within the same sentence.[5] Out of these, 12.1% involve the misplacement of a verb.[6] The vast majority of verb placement errors in MERLIN are concentrated among the levels A2–B2 (94.9%).

In the FalkoEssayL2 v2.4 corpus (Lüdeling et al., 2008),[7] which only comprises German L2 texts at the levels B2–C2, movement errors make up 29.6% of total errors. Out of these, 12.5% involve the misplacement of a verb. This suggests that even at advanced levels, learners continue to produce verb placement errors at rates comparable to those at

[3]Accessible at: `https://commul.eurac.edu/annis/merlin/`, last accessed: 2025/06/03.

[4]CEFR: Common European Framework of Reference for Languages.

[5]We only count grammatical errors towards the total, i.e., spelling errors are excluded.

[6]Appendix B contains the queries with which we determined the number of total grammatical error occurences in MERLIN and Falko, the number of movement errors, and the number of movement errors involving the misplacement of a verb.

[7]Accessible at: `https://korpling.german.hu-berlin.de/falko-suche/`, last accessed: 2025/06/03.

Tool	Languages	Verb Placement	Large-Scale Eval
GermDetect	af, de, nl, fo, is, da, nb, nn, sv	af, de, nl, fo, is, da, nb, nn, sv	✓
Grammarly	en	/	✗
LanguageTool	en, de, nl, da, sv, +21 others	de	✓
ProWritingAid	en	/	✓
Quillbot	en, de, nl, +3 others	de, nl	✗

Table 3: Overview of grammar checking tools with language support, verb placement error detection capabilities, and large-scale evaluation capabilities.

earlier stages, highlighting the pedagogical value of providing targeted feedback on such errors.

In the Icelandic Child Language Error Corpus and the Icelandic L2 Error Corpus (Ingason et al., 2021; Glisic and Ingason, 2022), there are two designated error categories for verb placement errors related to V2 violations.[8] This indicates that verb placement is not trivial not only in L2 learning but also in first-language acquisition.

However, verb placement errors as a subset of word order errors are heavily underrepresented in all relevant benchmark datasets as we will point out in the following section.

2.3 Word Order Errors in GED/GEC/LA Benchmark Datasets

In recent years, several shared tasks have been organized in the field of GED/GEC. Additionally, LA tasks have been developed to test language models' linguistic capabilities. Table 2 presents the most prominent benchmark datasets from all three domains covering Germanic languages and the percentage of word order errors they contain.

All of the datasets have in common that word order errors typically only make up a very small fraction of the errors present within them, or no information about the distribution of word order errors is provided at all. Germanic languages other than English have only recently seen their representation increase. However, with the exception of the ScaLA dataset (Nielsen, 2023), no information is available about the presence and distribution of word order errors in the datasets containing subsets of the languages covered by GermDetect. This leaves the capabilities of language models in detecting erroneous word order underexplored.

This situation is especially critical in the context of Germanic languages, whose successful acquisition depends on mastering their complex verb placement rules. Verb placement errors, as a subset of word order errors, are consequently even more poorly represented in the datasets, thus limiting the development of GED systems capable of reliably detecting and providing feedback on such errors. Moreover, in the only Germanic dataset where information about the presence of word order errors is available – the ScaLA dataset – such errors were introduced using a naive corruption strategy that swaps adjacent tokens.[9] As a result, the dataset neither specifically targets verb placement errors nor reflects the kind of word order errors that naturally occur in learner language. Thus, it provides little insight into whether language models can detect naturalistic word order errors that could be produced by L2 learners.

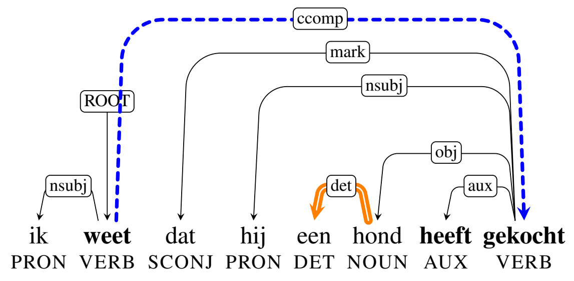
2.4 Survey GED/GEC Tools

While numerous writing assistants claim to provide feedback on grammatical errors, few are suitable for detecting verb placement errors in Germanic languages other than English. Among those capable of handling word order errors to some extent, many lack API access, which complicates large-scale evaluation.

Table 3 presents an overview of the most popular tools, the languages they support, and whether word order errors, particularly those involving verb placement, are among the phenomena they claim to be able to identify and provide feedback on. As the table shows, two of the most popular tools – Grammarly and ProWritingAid – do not support any of the relevant languages covered by GermDetect. While Quillbot claims to handle syntax errors in German and Dutch, and LanguageTool does so

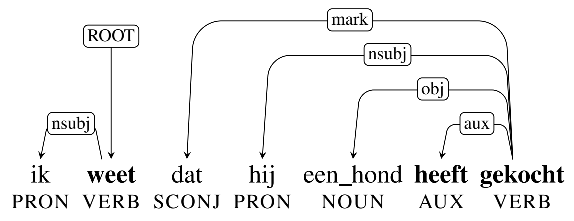
[8]Both corpora are accessible at: `https://github.com/icelandic-lt/iceErrorCorpusSpecialized/`, last accessed: 2025/06/03. Verb placement errors are annotated with the error code *v3*.

[9]To ensure the corrupted sentences are indeed ungrammatical, Nielsen (2023) enforces a variety of part-of-speech restrictions prohibiting certain token swaps.



ccomp
mark
nsubj
ROOT
obj
nsubj
det
aux
ik weet dat hij een hond heeft gekocht
PRON VERB SCONJ PRON DET NOUN AUX VERB

Figure 2: Dependency tree for the Dutch sequence "ik weet dat hij een hond heeft gekocht"; blue arc indicates where our algorithm splits verb phrases; orange arc indicates where our algorithm aggregates noun phrases into impermeable units.



mark
ROOT
nsubj
obj
nsubj
aux
ik weet dat hij een_hond heeft gekocht
PRON VERB SCONJ PRON NOUN AUX VERB

Figure 3: Dependency tree for the Dutch sequence "ik weet dat hij een hond heeft gekocht"; verb-headed phrases are aggregated as verbal heads plus their dependencies; noun-headed phrases are impermeable units.

for German, our manual evaluation on 100 GermDetect test sentences for each of the two languages found that their feedback on verb placement errors was largely unreliable, inaccurate, or incomplete.[10] This highlights that effective tools for providing feedback on verb placement in Germanic languages are unavailable as of today. To remedy this, in the following section, we present our linguistically informed corruption algorithm that can introduce verb placement errors into any UD-annotated Germanic language subject to V2 verb placement.

[10]A locally hosted server of LanguageTool's Python wrapper (Version 6.5; available at https://pypi.org/project/language-tool-python/, last accessed: 2025/06/03) flagged none of the 132 verb placement errors present in the German test sentences as word order errors; via their respective GUIs, LanguageTool was able to identify 41 and Quillbot was able to identify 38 out of 132 verb placement errors. For Dutch, Quillbot was able to identify 37 out of 130 verb placement errors. We counted errors as correctly identified even when they were not explicitly flagged as syntax errors, as long as the tool highlighted the relevant part of the sentence – whether as a missing or superfluous word or verb, or with a vague message such as "something is wrong" in combination with the appropriate error correction suggestion, among other generic error types.

3 Dataset Creation

Here, we briefly present the UD datasets as a basis for GermDetect, we present our preprocessing steps, and we describe our data corruption algorithm in detail.

3.1 Universal Dependencies

As the basis for our datasets, we use the UD Treebanks (Version 2.15; Zeman et al., 2024), including all available datasets for each of the GermDetect languages. Appendix A summarizes the datasets and their sizes after removing sentences without verbs.

Figure 2 illustrates how sentences are annotated for dependency relations in UD. Our algorithm currently operates both on the part-of-speech (POS)-tag level and the dependency arc level, but tokens are typically annotated with more linguistic information. We remove common punctuation tokens and lowercase all characters to ensure that models trained on GermDetect data cannot fall back on any orthographic information when determining the (in)correctness of verb placement and have to rely on their syntactic understanding only. This is especially relevant for sentences that begin with verbs, as the initial capitalization can reveal their original position. If such verbs are moved to another part of the sentence, the capitalization may serve as an unintended signal of misplacement.

3.2 Corruption Algorithm

Given a dependency parse tree T for a sentence $S = \{w_1, w_2, \ldots, w_n\}$, we define an aggregation procedure to extract syntactic substructures centered around **full verbs** and **nouns**. The algorithm proceeds as follows:

Step 1 - Aggregation of Verb-Headed Phrases. Identify all tokens $v \in S$ such that POS(v) = VERB. These serve as the roots of *verb-headed phrases*.[11]

For each full verb v, construct a *verb-headed phrase* $V_v \subseteq S$ defined as:

- V_v includes the full verb v itself,
- Plus all tokens in S that are **recursively governed by** v – i.e., all descendants of v in the subtree rooted at v,

[11]Due to inconsistent annotations, some UD treebanks use the VERB POS-tag for adjectives derived from verbs. We implemented a number of additional checks to prevent these tokens from heading their own verb-headed phrases.

- With the constraint that no token is included if the dependency path from it to v passes through another full verb (cf. verb phrase splitting in Figure 2).

This ensures that each verb-headed phrase captures the local syntactic scope of a full verb, while nested verbs and their subtrees are aggregated independently.

Step 2 - Aggregation of Noun-Headed Phrases within Verb-Headed Phrases. Identify all tokens $n \in V_v$ such that $\text{POS}(n) = \text{NOUN}$. These serve as the roots of *noun-headed phrases* within verb-headed phrases.

For each noun n, construct a *noun-headed phrase* $N_n \subseteq V_v$ defined as:

- N_n includes the noun n itself,
- Plus all tokens in V_v that are **recursively governed by** n – i.e., all descendants of n in the subtree rooted at n,
- With the constraint that no token is included if the dependency path from it to n passes through another noun.

Each noun phrase N_n is treated as an **impermeable unit**: During any subsequent processing, i.e., the data corruption, the tokens in N_n must remain contiguous and in their original order. No external tokens may be inserted into the span of a noun-headed phrase.

Step 3 - Output. The final output is a collection of verb-headed phrases $\{V_{v_1}, V_{v_2}, \ldots, V_{v_i}\}$, each rooted at a full verb. Within each V_v, zero or more noun phrases $\{N_{n_1}, N_{n_2}, \ldots, N_{n_j}\} \subseteq V_v$ are identified as impermeable subunits. This structure supports the linguistically-informed manipulation of the sentence while preserving core syntactic boundaries. Figure 3 illustrates how, with the help of UD's dependency structure, we are able to isolate syntactic structures that often correspond to main and subordinate clauses. This allows us to inject the data with more targeted corruptions, which aim to reproduce learner errors more closely.

Step 4 - Corruption of Verb-Headed Phrases. Each extracted verb-headed phrase V_v is corrupted by permuting the positions of a randomly selected subset of its verb tokens. Specifically:

- Identify all verb tokens $\tau \in V_v$ such that $\text{POS}(\tau) \in \{\text{VERB}, \text{AUX}\}$. Each such token is selected for permutation with a probability of roughly $p = 0.5$, resulting in the dataset containing approximately as many correctly placed as incorrectly placed verbs.
- The selected verb tokens may be relocated to any position within V_v, either before or after any other token, **provided that the relative order of all non-verb tokens is preserved**.
- Noun-headed phrases $N_n \subseteq V_v$ remain contiguous and untouched; verb tokens may move around them but not split them.
- If $V_v = V_{v_1}$, i.e., it is the first verb-headed phrase in the sentence, no verb token may occupy the first position in V_v, unless a verb originally located in this position remains in it. This constraint is imposed to avoid the accidental generation of valid polar question syntax, which would undermine the goal of synthetically generating syntactically perturbed structures.
- All verb tokens that are moved from their original positions are automatically labeled as syntactically incorrect (`F`), enabling the generation of training data for GED models. This labeling strategy is justified by the relatively rigid verb placement rules found in Germanic languages, as explained earlier. Verbs that remain in their original position are labeled as correct (`C`), all non-verb tokens are labeled as other (`O`).

Example (9) illustrates how misplacing a verb in the first position is not permitted by our algorithm. This is to avoid generating well-formed polar questions that would be incorrectly labeled as ungrammatical (cf. Table 1).[12]

(9) ik **weet** | dat hij een_hond heeft gekocht
weet ik* | dat hij een_hond heeft gekocht

Example (10) showcases all possible corruptions when we reduce the number of verbs in the subordinate clause of our running example to one, resulting in *ik weet dat hij een hond heeft* "I know that he has a dog". There are 5 positions that *heeft* can theoretically take. One of them corresponds to the correct placement, while the last corruption in

[12]In addition to the standard *left asterisk indicating linguistically unacceptable examples, we use the asterisk on the right* side to indicate examples not permitted by our algorithm.

the example is not permitted due to the impermeable noun-headed phrase constraint. If a verb in the phrase is selected to be permuted, a permitted position is randomly chosen.

(10) ik weet | dat hij een_hond **heeft**
ik weet | ***heeft** dat hij een_hond
ik weet | *dat **heeft** hij een_hond
ik weet | *dat hij **heeft** een_hond
ik weet | *dat hij een **heeft** hond*

Finally, example (11) illustrates what a labeled corrupted sentence could look like if we reintroduced the second verb. Note that the relative order of non-verb tokens always remains the same but the relative order of verb tokens to one another can change.

(11)	*ik*	***weet***	|	**dat*	***gekocht***	*hij*	***heeft***	*een*	*hond*
	`O`	`C`	|	`O`	`F`	`O`	`F`	`O`	`O`

The GermDetect algorithm makes it possible to insert errors that specifically target verb placement while making sure to exclude misplacements that are unlikely, such as breaking up noun phrases, or that would result in a well-formed sentence despite the change of verb placement, such as in polar questions. The full extent of our dataset is summarized in Appendix A.

4 Benchmark Results

In this section, we present our experimental setup and analyze the results of evaluating various mBERT configurations trained on GermDetect.

4.1 Experimental Setup

Following the creation of the GermDetect dataset, we train and evaluate multiple mBERT models using various training dataset configurations and their combinations (Devlin et al., 2019). We use mBERT as our base model primarily due to computational constraints and methodological considerations. All experiments are conducted locally on a MacBook Pro with an Apple M4 Pro chip and 24 GB of RAM (macOS 15.4.1), which makes training larger models such as XLM-R impractical (Conneau et al., 2020). In addition to its lighter memory footprint, mBERT's relatively lower performance ceiling provides a clearer basis for analyzing the impact of training data composition. Specifically, we compare training on the target language alone with training on the target language and related Germanic languages, as explained in Section 4.3. Since stronger models like XLM-R often achieve robust performance regardless of training data composition, they may obscure more subtle transfer effects that are easier to detect with a smaller model. This choice also supports the growing emphasis on environmentally responsible NLP, as smaller models require significantly less energy to train and deploy.

To implement our approach, we add a `BertForTokenClassification` head to mBERT and fine-tune the sequence tagger using Hugging Face's Trainer API. Appendix C summarizes the parameters we use to train our models. For each input sentence, the model generates one of three labels for each token: `O` for tokens that are not verbs, and `C` (correct) or `F` (false) for verb tokens, depending on whether their placement in the sentence is correct or incorrect.

We retain the original dataset splits provided by UD for training, development, and testing, and we do not make any further modifications beyond removing sentences without any verb tokens, as explained in Section 3.1. During training, the models are evaluated based on their loss on the development set. At inference time, in line with earlier works in GED (Bell et al., 2019; Yuan et al., 2021; Volodina et al., 2023), the models are evaluated using the macro-averaged $F_{0.5}$ score, which we compute exclusively over the `C` and `F` categories. This metric places greater emphasis on precision than recall, which aligns with standard practice in intelligent computer-assisted language learning applications where false positives, i.e., flagging correct structures as incorrect, are especially undesirable as they risk demotivating learners.

4.2 Monolingual Baseline Configurations

Table 4 presents the $F_{0.5}$ performance results for different configurations of the mBERT model on the GermDetect test data.

We evaluate three baseline configurations: `target`, `random`, and `adjacent`. The `target` configuration trains models exclusively on GermDetect data from the target language. The `random` configuration trains models on data in which half of the verbs assume any position within a sentence other than their original position, while the other half remain in their original positions. This corruption strategy represents generic verb placement errors,

mBERT Configuration	F$_{0.5}$ Score by Language								
	af	de	nl	fo	is	da	nb	nn	sv
`random`	0.74	0.80	0.72	0.63	0.74	0.76	0.80	0.79	0.79
`adjacent`	0.71	0.67	0.69	0.68	0.64	0.72	0.74	0.75	0.75
`target`	0.82	**0.94**	0.88	0.72	0.82	0.85	0.90	0.89	0.86
`all`	**0.88**	**0.94**	**0.89**	**0.85**	**0.84**	**0.90**	**0.93**	**0.92**	**0.91**
`all-balanced`	0.86	**0.94**	**0.89**	0.79	0.83	0.88	0.92	0.91	0.90
`west`	0.87	0.93	0.88	–	–	–	–	–	–
`west-balanced`	0.84	**0.94**	0.88	–	–	–	–	–	–
`north`	–	–	–	0.84	0.83	0.89	0.92	0.91	0.90
`north-balanced`	–	–	–	0.78	0.83	0.86	0.92	0.91	0.89
`island`	–	–	–	0.82	0.83	–	–	–	–
`island-balanced`	–	–	–	0.75	0.83	–	–	–	–
`mainland`	–	–	–	–	–	0.88	0.91	0.90	0.89
`mainland-balanced`	–	–	–	–	–	0.87	0.92	0.90	0.89

Table 4: Macro-averaged F$_{0.5}$ performance scores (computed over the `C` and `F` categories only) of different configurations of the mBERT model across the Germanic languages covered by GermDetect; models ablate the influence of training classifiers based on different data corruption strategies and by combining the training data of structurally related groups of languages; `balanced` indicates that the target language sets the upper limit for how many sentences of each language are used to train the classifier.

i.e., linguistically uninformed verb placement errors. For the `adjacent` configuration, half of the verbs switch positions with one of their adjacent tokens (both left and right swaps are equally frequent), approximating the word-order error induction mechanism described in Nielsen (2023), but applied to verbs only.

As shown in the results, the `target` configuration significantly improves performance across all languages compared to both the `random` and `adjacent` configurations, indicating that training directly on GermDetect data effectively enhances the detection of naturalistic verb placement errors.

4.3 Ablation of Multilingual Configurations

Next, we assess the effects of training models on configurations that include structurally related languages.

For the `all` configuration, one model is trained on all available data. Similarly, the `west`, `north`, `island`, and `mainland` configurations each include all data from the languages within their respective groups.[13] In the `balanced` configurations, the number of sentences from each language is capped at the level of the target language, ensuring that the target language contributes at least as many sentences as any other included language. This means that, e.g., in the `north-balanced` configuration with Faroese as the target language, if the Faroese dataset contains *X* sentences, then each of the other North Germanic languages can contribute at most *X* sentences to the training set, ensuring that Faroese is not underrepresented in the training data.

The results demonstrate that training on all available data yields the best-performing models, consistent with the expectation that more data generally improves performance. However, training solely on West Germanic data yields performance scores nearly equivalent to those obtained by using all data when tested on West Germanic languages, a trend also observed among North Germanic languages. Furthermore, training exclusively on mainland Scandinavian languages results in only a minor performance reduction in these languages, compared to training on all North Germanic languages. These observations suggest that, while incorporating more diverse data is generally beneficial, the models effectively exploit structural similarities among related languages, achieving performance scores close to the best-performing configurations. Balancing the representation of languages within the training sets does not provide additional benefits; thus, it is preferable to utilize all available data.

[13]West Germanic: af, de, nl. North Germanic: fo, is, da, nb, nn, sv. Island Scandinavian: fo, is. Mainland Scandinavian: da, nb, nn, sv.

It is equally unsurprising that German benefits the least from the inclusion of related languages in training, as it is by far the most well-represented language in GermDetect. Similarly, Icelandic and Dutch also exhibit only minor performance improvements. In contrast, Afrikaans and Faroese – being the languages with the least available data – benefit the most from the inclusion of data from related languages in training.

5 Conclusion

We have introduced **GermDetect**: UD-based, linguistically informed verb placement error **detect**ion datasets for learners of **Germ**anic languages, designed as a token classification task. As our datasets are UD-based, we can provide them in most major Germanic languages: Afrikaans, German, Dutch, Faroese, Icelandic, Danish, Norwegian (Bokmål and Nynorsk), and Swedish. Unlike existing resources, GermDetect targets a specific and pedagogically relevant error type that is underrepresented in current benchmark datasets and goes undetected by most existing GED/GEC tools. Our results show that multilingual models trained on data corrupted by the GermDetect algorithm outperform models trained on naively corrupted data. Furthermore, while training on all data consistently yields the highest performance, models trained on structurally related languages perform nearly as well – demonstrating the benefits of typological similarity. Crucially, the amount of available training data strongly influences the degree to which models benefit from multilingual training: high-resource languages such as German, Icelandic, and Dutch see marginal improvements, whereas low-resource languages such as Afrikaans and Faroese benefit substantially. These findings highlight the importance of both linguistic structure and data quantity in training robust GED/GEC models and suggest that targeted, linguistically informed error induction can support the development of systems capable of providing fine-grained feedback on complex syntactic phenomena such as verb placement in Germanic languages.

Limitations

In its current implementation, the verb placement error generation algorithm is subject to several limitations. It assumes UD sentences to be grammatically well-formed and their annotations to be accurate. However, this is not always the case, as data quality can vary and annotation practices can differ across datasets. To address this, future iterations of the algorithm should include more robust checks to ensure that all tokens are treated correctly, even in the presence of inconsistent or imperfect annotations. Another limitation lies in the assumption that every corruption introduced by the algorithm results in an incorrect sentence. In theory, however, some corruptions can still yield well-formed or acceptable constructions. Although restrictions are currently in place to prevent the generation of polar question syntax, and the relatively rigid word order of Germanic languages minimizes the number of such cases, they are not entirely eliminated. A manual inspection of 100 German sentences revealed an error rate of 2.9% in which verbs were incorrectly labeled. For Dutch, this error rate was 2.1%. Future developments should aim to reduce these accidentally grammatically sound relocations of verbs even further. In the Dutch sentences we manually checked, we also noticed that, for very long sentences, the algorithm sometimes did not restore the correct order of the extracted verb-headed phrases. This affected the soundness of the respective sentence in 4% of the sentences we checked. In a future iteration, it should be examined whether this was caused by an algorithmic shortcoming or by insufficient data and annotation quality.

Additionally, while keeping noun phrases intact represents a step toward introducing linguistically informed errors, further restrictions could be added – for instance, preserving additional syntactic structures or avoiding verb placements that would misrepresent subordinate clause boundaries, such as placing verbs in front of subjunctions. Future versions of the algorithm could also take further advantage of the rich linguistic information already present in UD annotations to better approximate a wider range of learner phenomena. Another issue arises when the root of a sentence is not a verb but a noun, which can occur in some UD treebanks when the main clause contains a copula verb. In such cases, the corresponding noun phrase is currently not impermeable. Moreover, although the current implementation provides feedback on the position of verb placement errors, it does not yet explain why a given verb placement is correct or incorrect. Providing this type of explanatory feedback in the form of a feedback message would offer more meaningful support to learners. Lastly, it goes without saying that it would be desirable to evaluate our models on actual learner data once natural learner data annotated for verb placement errors become available.

Ethics Statement

We do not see any major ethical concerns in the context of this work. As always, to promote diversity, it would be desirable to extend the coverage of our datasets to more Germanic languages, in particular, lower-resourced ones, minority languages, and regional language varieties such as Luxembourgish, Frisian, Yiddish, Low German, Swiss German, etc. As our algorithm operates on UD data, this would be implementable as soon as sufficient UD-annotated data for these languages become available. Of course, implementing an API demonstration of our trained verb placement error detection tools is a natural next step to ensure that language learners can directly benefit from the models we developed.

Acknowledgments

This research was supported by TrustLLM funded by Horizon Europe GA 101135671. We would like to thank Marco Kuhlmann and Marcel Bollmann for their support and guidance during the earlier stages of this project.

References

Ásgrímur Angantýsson. 2021. English-like V3-orders in matrix clauses in Icelandic. *Working Papers in Scandinavian Syntax*, 106:17–46.

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.

Adriane Boyd, Jirka Hana, Lionel Nicolas, Detmar Meurers, Katrin Wisniewski, Andrea Abel, Karin Schöne, Barbora Štindlová, and Chiara Vettori. 2014. The MERLIN corpus: Learner language and the CEFR. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 1281–1288, Reykjavik, Iceland. European Language Resources Association (ELRA).

Christopher Bryant, Mariano Felice, Øistein E. Andersen, and Ted Briscoe. 2019. The BEA-2019 shared task on grammatical error correction. In *Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 52–75, Florence, Italy. Association for Computational Linguistics.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online. Association for Computational Linguistics.

Robert Dale and Adam Kilgarriff. 2011. Helping our own: The HOO 2011 pilot shared task. In *Proceedings of the 13th European Workshop on Natural Language Generation*, pages 242–249, Nancy, France. Association for Computational Linguistics.

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

Isidora Glisic and Anton Ingason. 2022. The Nature of Icelandic as a Second Language: An Insight from the Learner Error Corpus for Icelandic. *CLARIN Annual Conference*, pages 23–33.

Anton Karl Ingason, Þórunn Arnardóttir, Lilja Björk Stefánsdóttir, and Xindan Xu. 2021. The Icelandic Child Language Error Corpus (IceCLEC) Version 1.1. CLARIN-IS.

Anke Lüdeling, Seanna Doolittle, Hagen Hirschmann, Karin Schmidt, and Maik Walter. 2008. Das Lernerkorpus Falko. *Deutsch als Fremdsprache*, 2:67–73.

Arianna Masciolini, Andrew Caines, Orphée De Clercq, Joni Kruijsbergen, Murathan Kurfalı, Ricardo Muñoz Sánchez, Elena Volodina, and Robert Östling. 2025. The MultiGEC-2025 Shared Task on Multilingual Grammatical Error Correction at NLP4CALL. University of Tartu Library.

Courtney Napoles, Keisuke Sakaguchi, and Joel Tetreault. 2017. JFLEG: A fluency corpus and benchmark for grammatical error correction. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 229–234, Valencia, Spain. Association for Computational Linguistics.

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*, pages 1–14, Baltimore, Maryland. Association for Computational Linguistics.

Hwee Tou Ng, Siew Mei Wu, Yuanbin Wu, Christian Hadiwinoto, and Joel Tetreault. 2013. The CoNLL-2013 shared task on grammatical error correction. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning: Shared Task*, pages 1–12, Sofia, Bulgaria. Association for Computational Linguistics.

Dan Nielsen. 2023. ScandEval: A benchmark for Scandinavian natural language processing. In *Proceedings of the 24th Nordic Conference on Computational Linguistics (NoDaLiDa)*, pages 185–201, Tórshavn, Faroe Islands. University of Tartu Library.

Antje Orgassa. 2009. *Specific Language Impairment in a Bilingual Context: The Acquisition of Dutch Inflection by Turkish-Dutch Learners*. Ph.D. thesis, University of Amsterdam. Published by UtrechtLOT.

Sarah Schimke and Christine Dimroth. 2018. The influence of finiteness and lightness on verb placement in L2 German: Comparing child and adult learners. *Second Language Research*, 34(2):229–256.

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.

Alex Warstadt, Alicia Parrish, Haokun Liu, Anhad Mohananey, Wei Peng, Sheng-Fu Wang, and Samuel R. Bowman. 2020. BLiMP: The benchmark of linguistic minimal pairs for English. *Transactions of the Association for Computational Linguistics*, 8:377–392.

Alex Warstadt, Amanpreet Singh, and Samuel R. Bowman. 2019. Neural network acceptability judgments. *Transactions of the Association for Computational Linguistics*, 7:625–641.

Marit Westergaard and Terje Lohndal. 2019. *Verb Second Word Order in Norwegian Heritage Language: Syntax and Pragmatics*. Georgetown University Press.

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.

Daniel Zeman et al. 2024. Universal Dependencies 2.15. LINDAT/CLARIAH-CZ digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

A Dataset Specifications

Language	Dataset	Split	# Sents	# C	# F	# O
af	AfriBooms	Train	1,300	2,617	2,615	25,341
		Dev	192	404	402	3,966
		Test	425	776	776	7,637
de	GSD	Train	9,710	9,728	9,726	129,050
		Dev	613	697	694	6,730
		Test	648	737	737	7,740
	HDT	Train	104,322	120,674	120,674	1,391,353
		Dev	12,234	14,105	14,104	159,956
		Test	12,758	14,497	14,497	167,172
	LIT	Test	1,784	2,657	2,656	27,315
	PUD	Test	738	1,053	1,052	10,864
nl	Alpino	Train	11,604	14,167	14,165	132,951
		Dev	697	777	775	8,536
		Test	557	827	827	7,947
	LassySmall	Train	10,389	14,063	14,060	167,415
		Dev	1,252	1,790	1,788	19,352
		Test	1,272	1,719	1,718	19,241
fo	FarPaHC	Train	1,015	2,284	2,283	13,993
		Dev	292	728	726	5,676
		Test	299	760	760	5,845
	OFT	Test	1,203	770	770	6,869
is	GC	Train	3,908	6,707	6,706	58,746
		Dev	495	941	938	7,972
		Test	529	860	861	7,755
	IcePaHC	Train	32,505	62,068	62,066	475,106
		Dev	4,579	12,379	12,375	94,062
		Test	4,965	11,982	11,983	95,484
	Modern	Train	2,659	5,660	5,660	45,435
		Dev	383	745	746	6,141
		Test	432	883	882	7,620
	PUD	Test	995	1,484	1,481	14,091
da	DDT	Train	3,989	6,186	6,188	55,403
		Dev	518	825	823	7,167
		Test	532	803	803	6,870
nb	Bokmaal	Train	14,120	19,775	19,773	171,489
		Dev	2,178	2,966	2,966	25,653
		Test	1,810	2,526	2,526	21,095
nn	Nynorsk	Train	12,718	18,464	18,461	177,112
		Dev	1,685	2,429	2,425	22,475
		Test	1,337	1,858	1,858	18,115
sv	LinES	Train	3,008	4,672	4,669	39,408
		Dev	989	1,562	1,559	13,327
		Test	969	1,438	1,437	11,969
	PUD	Test	992	1,343	1,341	14,502
	Talbanken	Train	3,909	4,769	4,768	47,995
		Dev	478	709	708	7,263
		Test	1,140	1,573	1,571	14,798
Total			**275,607**	**381,884**	**381,816**	**3,795,723**

Table 5: Datasets per language and number of sentences (# Sents) as well as number of labels per data split.

B MERLIN and Falko Search Queries

Examples (12), (13), and (14) show the queries we used when looking for all errors, movement errors, and movement errors involving verbs in both MERLIN and Falko, respectively. Table 6 shows the exact error numbers. We show the MERLIN queries. The Falko queries are essentially equal with only minor differences in category names.

(12)

```
TH1Diff!=/(CHA)|(CHA\x2FSPLIT)/
```

(13)

```
TH1Diff=/(MOVS)|(MOVT)|(MOVS\x2FCHA)|
(MOVT\x2FMERGE)|(MOVS\x2FSPLIT)|
(MOVT\x2FCHA)/
```

(14)

```
TH1Diff=/(MOVS)|(MOVT)|(MOVS\x2FCHA)|
(MOVT\x2FMERGE)|(MOVS\x2FSPLIT)|
(MOVT\x2FCHA)/
& tok_pos=/(VAFIN)|(VAIMP)|(VAINF)|
(VMFIN)|(VMINF)|(VVFIN)|(VVIMP)|
(VVINF)|(VVPP)|(VVIZU)|(VAPP)/
& #1_=_#2
```

Errors / Corpus	**MERLIN**	**Falko**
All grammatical errors	14,366	8,535
Movement errors	4,880	2,526
Movement errors with verb	591	315

Table 6: Movement errors across MERLIN and Falko corpora.

C Model Training Parameters

Hyperparameter	**Value**
Number of training epochs	3
Training batch size	16
Evaluation batch size	16
Weight decay	0.01
Learning rate	5e-5
Checkpoint saving strategy	Per epoch
Evaluation strategy	Per epoch
Max number of saved checkpoints	1
Metric for best model selection	Evaluation loss
Load best model at end	True

Table 7: Model training parameters.