Error Detection and Correction for English Learners using Neural Models

1st Zuoheng Dai
Instituto de Computación
Facultad de Ingeniería
Universidad de la República
Montevideo, Uruguay
zuoheng.dai@fing.edu.uy

2nd Martín Manitto
Instituto de Computación
Facultad de Ingeniería
Universidad de la República
Montevideo, Uruguay
martin.manitto@fing.edu.uy

3rd Luis Chiruzzo
Instituto de Computación
Facultad de Ingeniería
Universidad de la República
Montevideo, Uruguay
luischir@fing.edu.uy

4th Aiala Rosá
Instituto de Computación
Facultad de Ingeniería
Universidad de la República
Montevideo, Uruguay
aialar@fing.edu.uy

***Abstract*—In recent years, the importance of English language learning in Uruguay has been steadily growing. However, the country faces a significant challenge: a shortage of teachers, which makes it difficult for students to achieve an adequate English level. This problem particularly affects children at the initial levels, limiting their progress in language learning. In this context, it is crucial to assess student progress in order to define educational policies. This project builds on the work of previous initiatives that seek to support teachers in correcting exercises written by students. The central purpose of the project is to develop a tool that facilitates the automatic correction of texts written by English learners at the initial stages. The tool must receive the texts and return a corrected version, highlighting the identified errors. Initially, we explored the exclusive use of Large Language Models (LLMs) to address this problem. However, after several experiments, the results were not satisfactory. Given this limitation, we opted for an alternative solution that combines the use of LLMs, a module for detecting differences, and a classifier trained to predict error types, achieving the same correction objective, developing a system composed of three independent modules. We obtained an error corrector based on the Mistral LLM with an F0.5 score of 0.81, and an error classifier with an F1 score of 0.76. Overall, the system achieved an F0.5 performance of 0.68.**

***Index Terms*—NLP, AI in Education, Grammar Error Correction, English Learning, Language Models**

I. Introduction

In Uruguay, for several years now, increasing importance has been given to both English language teaching and the use of computer tools in education. Two decades ago, not all primary schools had English teachers or access to computers. However, through programs such as Inglés sin Límites 2030[1], CEIBAL[2], and CEIBAL en Inglés[3], universal English language education, supported by various technological solutions, is being achieved in both urban and rural schools.

However, more work is still needed to fully reach students in rural schools, where human resources are limited. Teachers are responsible for students of different grades, and English teachers are often in short supply. Therefore, to partially reduce this gap, the use of Artificial Intelligence tools, especially those based on Natural Language Processing techniques, can supplement some of the tasks typically carried out by teachers.

In this paper, we describe a tool that can identify, correct, and classify errors in texts written by English students, aiming to simplify or reduce the workload of English teachers. There have been attempts in the past at creating hand-crafted rules to identify and correct errors in English texts written by Uruguayan students [1]. This paper, however, presents a strategy based on Large Language Models (LLMs) to correct texts, combined with an alignment module to identify corrected spans, and a final logistic regression classifier to identify the error types.

The paper is organized as follows. Section II introduces concepts and related work on the field of GED and GEC; Section III describes our initial experiments using bare LLMs and prompting to solve this problem and the shortcomings of this approach; Section IV described our proposed solution to overcome these issues; Section V shows the experimental results obtained with our method; and finally Section concludes and presents the future work.

II. Related work

The problem addressed in this paper falls within the field of Grammatical Error Detection (GED) and Grammatical Error Correction (GEC). Here, the term "grammatical" is not used in the descriptive sense but in the traditional sense, as it aims to correct all types of errors, including lexical, spelling, or syntactic errors [2], [3]. Regarding both tasks, GED and GEC, several competitions have been conducted in the last decade.

CoNLL-2013 [4] and CoNLL-2014 [5] Shared Tasks on GEC provided datasets for training error detection and correction tools. In the 2013 edition, a set of five different errors was defined, extended to 28 errors for the 2014 edition.

The BEA-2019 [6] Shared Task on GEC. Given a sentence written by a student, the participants had to generate a corrected version of the sentence. The task did not include error classification.

MultiGED-2023 [2] addressed only error detection, but MultiGEC-2025 [7] extended the task to error correction. Unlike the previously described competitions, no specific types of errors are defined; instead, two variants of text correction

[1]https://www.anep.edu.uy/codicen/politicas-linguisticas/focus-on-first
[2]https://ceibal.edu.uy/
[3]https://ceibal.edu.uy/plataformas-y-programas/ceibal-en-ingles/

are proposed: a "minimal edits" track, where texts are corrected preserving as much as possible their original grammar, vocabulary, and style; and a "fluency edits" track, where more changes are allowed on the original text, aimed at producing more idiomatic language. In the 2025 edition, the majority of approaches were based on LLMs, but the official results are far from solving the problem.

Concerning the usage of LLMs for GEC, [8] evaluates the LLM's tendency to overcorrection. The authors experiment with fine-tuning, and zero-shot and few-shot prompting for correcting texts written by learners of English as a foreign language. They find that overcorrection occurs primarily in the writing of advanced language learners, rather than at the beginner and intermediate levels.

When considering English texts written by Uruguayan students, [1] uses a dataset of approximately 54,000 written by Uruguayan learners of English with a basic level of proficiency. This dataset includes typical errors of local students that might not necessarily be found in other regions, which makes it especially relevant for the construction of an effective correction tool. All the texts in the dataset were manually classified by English teachers on a scale from 0 to 6, but only a very small subset (95 texts) was manually annotated and expanded with marks indicating segments containing errors and their corresponding error types. In total 15 error types were considered, but many of them did not have a large number of examples. They created a set of heuristics for detecting each error type, ranging from rule-based heuristics to a combination of spellcheckers and BERT-based models. In contrast, in this work we use the same dataset but we focus mainly on the use of more modern LLMs, which have proven to be very useful for many NLP tasks.

III. Initial experiments

In recent years, the use of LLMs has revolutionized many aspects of NLP, improving the state of the art in most of the tasks and even allowing to explore new previously unexplored tasks. We first wanted to analyze if these powerful LLMs were able to solve the GED and GEC problems by themselves. In this section we show how different models behave when prompted to solve this task, either by only correcting the texts, or by providing details of what errors were found.

Example (1) shows a text like the ones that can be found in the original corpus[4]. The dataset is split in a large training set with no annotations (only broad grades from 0 to 6), and much smaller development and test sets, comprising 53 and 42 texts respectively, fully annotated with error spans, correction suggestions, and error types.

(1) she is andrea, she has 14 years old. She eats apple. i like pizza, and ried and i dont likes singin and eating a candies.

In the manually corrected version shown in example (2), several mistakes of different types have been identified, with

[4]This is not an actual example from the corpus, it is constructed combining different examples to show different error types.

Error Type	Tag	Original	Correction
Spelling	S	ried	reading
		singin	singing
Sentence caps.	CBOS	she is	She is
I pronoun caps.	CPI	i like	I like
		i dont	I don't
Proper name caps.	CPN	andrea	Andrea
Verb form	VF	she has 14	she is 14
Subject-Verb agreement	SVA	I dont likes	I don't like
Missing article	MA	eats apple	eats an apple
Unnecessary article	UA	eating a candies	eating candies

TABLE I: Types of errors marked in the original corpus that we consider for this work, and the original text and correction suggestions corresponding to the errors in our example.

the error types and correction suggestions shown in Table I. Out of the 15 error types marked in the corpus, we only considered the most common 8 error types.

(2) She is Andrea, she is 14 years old. She eats an apple. I like pizza, and reading and I don't like singing and eating candies.

When asked to give a correct version of the text, GPT yields the version shown in example (3). Note that many of the mistakes detected in the manual annotation were also corrected by GPT (the capitalization, spelling, and verb errors), but it also tends to overcorrect, like splitting sentences and changing an 'and' to an 'or'. These could be considered improvements on the original text, but in an educational context it is often preferable not to change the original text too much. LLMs also make some guesses that are not necessarily correct, for example GPT suggested changing 'ried' for 'rice', while the manual annotation chose 'reading', which might be better suitable in context. Also, there is sometimes no unique answer as to what corrections have to be made: the annotator chose to change "eats apple" to "eats an apple", while the LLM chose to change it to "eats apples", and both changes are valid (this is often observed in the context of GEC as is an issue for automatic metrics [9]).

(3) She is Andrea. She is 14 years old. She eats apples. I like pizza and rice, and I don't like singing or eating candy.

The results when asking an LLM to write a correct version of a text with errors are generally very good, but when asking the model to give a detail of which errors were detected and their types, the performance is considerably lower.

We asked different LLMs either to give a correct version of a text or to detect, classify and correct the errors in a text, for all the texts in the dev corpus. In these initial experiments we iteratively tried different prompts to solve the task, all the prompts are described in Appendix A, and in this section we will show the results only of the best one. Table II shows the results for this experiment, in which we used models GPT 3.5 [10], LLaMA 2 70B [11], and Mistral 7B [12]. The rows marked with labels "No" mean only correcting the text, while the ones marked with labels "Yes" mean that the model had also to detect and classify the errors. Following [5] we use $F_{0.5}$ as the main metric because it emphasizes precision more than recall, and in an educational context it is very important that the errors shown are accurate, even if not all errors are detected. Note that the performance when the model has to predict the labels for the error types is always much lower. There is also an 'Extra' column which shows the average number of errors per sample that were corrected by the LLM but that were not considered during the manual annotation.

TABLE II: Results using a prompting only solution for GPT 3.5, LLaMA 2 and Mistral over the dev set.

Model	Labels	Precision	Recall	$F_{0.5}$	Extra
ChatGPT	No	0.851	0.873	0.848	1.512
ChatGPT	Yes	0.514	0.460	0.460	1.070
LLaMA	No	0.900	0.843	0.880	0.860
LLaMA	Yes	0.588	0.422	0.508	0.582
Mistral	No	0.815	0.820	0.804	1.698

There are several reasons why the LLMs have lower performance when trying to predict at the same time the error span, correction suggestion, and error type. First of all, it is difficult just to explain all the task in the prompt, as the descriptions of the error types are lengthy, and including some few-shot examples the prompt could be considerably long. This is not a problem for modern models to handle, but still they might ignore parts of the prompt when it is too long. On top of this, the models tend not to reproduce a text if it has errors, preferring to give a corrected version instead, which hinders the capability to reproduce the original text adding error tags where appropriate. Finally, the models might create the output in a different format than expected, for example a list of errors (similar to Table II) instead of a unique text with tags. During our experiments, both types of outputs were observed, and we had to manually analyze how correct they were independently of the output format.

These issues might be at least partially fixed by doing fine-tuning instead of few-shot prompting, but this was not possible given the very small number of annotated examples we had, and also given our computational constraints (more on this in section IV-B). Because of this, we decided to go for the two-stage approach described in the next section.

IV. Proposed solution

The implemented solution is a system that works in two steps (see Fig. 1): It first uses a LLM to create a corrected version of an input text, given the good capabilities of modern language models to create well-written texts that preserve the meaning of the input. Then both the original input and the corrected version are submitted to an alignment process that detects which parts of the original text were changed, framing the modifications introduced by the model as "change suggestions", with pairs containing an original fragment and its corrected version. These pairs of fragments are then run through a classifier that labels which type of error was likely the cause of the change suggestion.

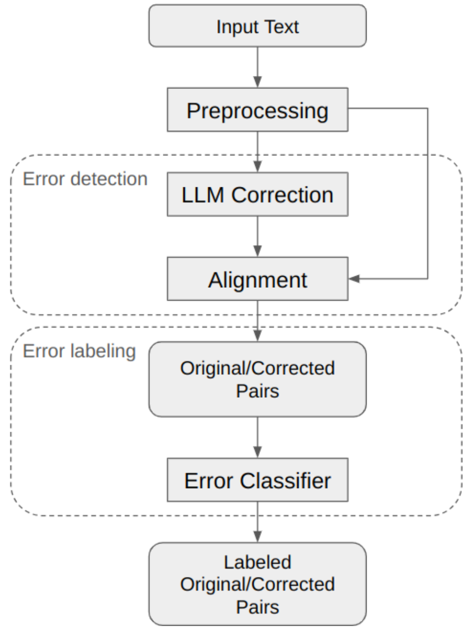

Fig. 1: General architecture of the system.

A. Preprocessing

All texts we used from the corpus were preprocessed in the following way:

- Add a blank space after punctuation symbols.
- Replace multiple consecutive blank spaces with only one blank space.
- Trim blank spaces from the beginning and end of texts.
- Normalize non-standard uses of accents or other symbols for apostrophes.
- Replace line breaks for blank spaces.

B. Corrections with LLM

The initial experiments shown in section III were done using the web APIs of GPT and LLaMA and using their biggest models. However, this scenario is not ideal for making further experiments that would make use of more data, we needed a model that would run locally in the infrastructure we had, which is the ClusterUY supercomputing environment [13]. Because of this, we started doing experiments with the Mistral 7B model, and seeing that its results for text correction were lower than for the bigger models but not by a large margin (see Table II, we just selected this model for further experiments.

The system prompt asked the model to give a correct version of an input text and also contained three examples. We manually corrected three texts from the training data, which had no manual annotations, and used them as few-shot examples in the prompt.

C. Alignment

Following [14], we used a method based on the Levenshtein edition distance to align the original and corrected texts. In this method, we calculate the distance at token level, with the options to insert, remove or substitute a token at each step, and using a dynamic programming transversal to find the best configuration, i.e. the one with the lowest cost.

However, just considering all tokens equal would result in many cases where there is no clear way of aligning two words. Consider for example the alignment between the source text “fishing” and the target text “to fish”. There are two possible alignments here: substitute “fishing” with “to” and insert “fish”; or insert “to” and substitute “fishing” with “fish”. The decision would be easy for a human annotator, as “fishing” and “fish” are two semantically related words, but if we just use the same cost for all token substitutions, the algorithm would not have a way to decide which is the best option. Because of this, we calculate the substitution cost as a function of the character-based Levenshtein edition distance of the tokens. In this case, the distance between “fishing” and “to” would be 1.0, and between “fishing” and “fish” would be 0.43, so the algorithm would prefer the second option.

After detecting all insertions, removals, and substitutions, they are inserted in the original text as error tags, with the corresponding substitution value as change suggestion. We used the heuristic followed by [14] of grouping consecutive cases of the same operation as the same error (e.g. two insertions or two substitutions). This is not ideal in all situations, and better heuristics could be explored in the future, e.g. training a classifier to decide when to merge or split operations as in [15].

There are a few other tweaks to the heuristic to improve handling of some scenarios: we always consider punctuation as error delimiters; we incorporate rules for solving the tokenization of contractions; and we extend the error tag when the added word is an article so it finishes on the next noun, to mimic the way these errors are annotated in the corpus.

If we took example (1) and the corresponding manual correction (2), the algorithm would give example (4) as output. Notice that a better merging heuristic might split ‘I don’t like singing’ as several errors, but the simple heuristic works fine for most of the cases.

(4) `<E t="She">`she`</E>` is `<E t="Andrea">` andrea`</E>`, she `<E t="is">`has`</E>` 14 years old. She eats `<E t="an apple">`apple`</E>`. `<E t="I">`i`</E>` like pizza, and `<E t="reading">`ried`</E>` and `<E t="I don’t like singing">`i dont likes singin`</E>` and eating `<E t="">`a`</E>` candies.

D. Error type classifier

We trained two logistic regression classifiers that try to predict the type of error in each case. The classifiers were trained using the data from the dev set, as we did not have manually annotated data from the (much larger) training set. We split the dev set in 80% for training the classifiers and 20% for testing them.

First, we tried an approach similar to [14], extracting linguistic features from the data. We used spaCy [16] to POS-tag and lemmatize the texts, and an English dictionary from the NLTK library [17], and obtained the following features:

- Type of alignment operation (insertion, removal, substitution).
- POS of the tokens immediately before and after the error.
- Longest prefix between original text and correction.
- If both the original and the corrected version have the same lemma.
- If the original text words are in the dictionary.
- Token substitution cost as calculated in section IV-C.
- POS of the corrected text (for insertion and substitution).

In the second experiment, we used the BERT [18] embeddings corresponding to the original and corrected texts as features for the classifier.

Table III shows the results for the two classifiers on the 20% split used for testing. As can be seen in the table, the embeddings classifier had a better overall performance. However, in the rest of the paper we will use the features-based classifier (see section V).

Tag	Features	Embeddings
S	0.82	0.86
CBOS	1.0	0.97
CPI	1.0	1.0
CPN	1.0	0.80
VF	0.58	0.86
SVA	0.93	0.77
MA	0.20	0.86
UA	0.50	1.0
Global	0.76	0.88

TABLE III: Comparison of the F1 obtained for each classifier, broken down by error type.

After using the classifier, the final output of the system would look as in example (5), including the type labels in each of the errors.

(5) `<CBOS t="She">`she`</CBOS>` is `<CPN t="Andrea">`andrea`</CPN>`, she `<VF t="is">`has`</VF>` 14 years old. She eats `<MA t="an apple">`apple`</MA>`. `<CPI t="I">`i`</CPI>` like pizza, and `<S t="reading">`ried`</S>` and `<S t="I don’t like singing">`i dont likes singin`</S>` and eating `<UA t="">`a`</UA>` candies.

V. Experimental results

Table IV shows the results of our system over the dev and test sets. The upper half of the table shows the results of only the first part of the process: using the LLM to correct the texts and running the alignment process to detect the atomic errors. The lower half of the table includes the use of the classifier to identify the error types as well, so it is expected that all these results are lower than the ones in the upper half.

Although in this work we only considered eight error types, there were many more errors marked in the corpus. The LLMs did indeed correct many of these errors that we were not focusing on, and it would not make sense to count these

Experiment	Dataset	Precision	Recall	$F_{0.5}$
Only text correction and alignment	Dev	0.815	0.820	0.804
	Test	0.873	0.725	0.813
Correction, alignment and tagging of errors	Dev	0.726	0.511	0.609
	Test	0.782	0.549	0.682

TABLE IV: Final results using Mistral for correcting the text and the feature based classifier for predicting the error type.

good corrections as false positives, so during the evaluation we had to manually analyze the corrections to see if they were appropriate and count them as true positives. This was more complex in the tagged evaluation (lower half of the table), because our classifier was not shown any examples of the other types of errors during training, so it could not predict any of them. This part of the evaluation had to be done manually, analyzing each case to see if the errors were correctly classified.

Unfortunately, we trained the embeddings classifier (which had better intrinsic performance) after we performed the end-to-end evaluation of the system with the feature-based classifier, and due to time constraints we could not complete the same evaluation with the embeddings classifier, which we think would give even better results.

VI. Conclusions

We presented an experiment on automatic GED and GEC for English texts written by Uruguayan students. Our approach has the following steps: first we use the Mistral 7B model to create a corrected version of the text, then we align it with an adapted edition distance algorithm to detect error spans and their correction suggestions, and finally we use a logistic regression classifier to identify the most likely type of error for each span. This approach obtained interesting performance on the test set, with a $F_{0.5}$ of 0.81 (only for the correction process, without tags), and a $F_{0.5}$ of 0.68 for the whole process including tags.

We also experimented with another type of classifier that had better performance on the held-out corpus, but due to time constraints we could not finish the evaluation of this approach integrated to the rest of the system, but we are planning to do this evaluation in the future.

Furthermore, we are currently creating an improved corpus, obtaining texts written by secondary school students that are learning English from all across the country. These texts are written as responses to several different exercises, so as to include a much needed variety of answers, and are being carefully annotated with many types of errors. We plan to use this corpus to train better systems for GEC in our context, and one of the planned experiments is to test the approach described in this work with this new corpus to see how the performance compares with more varied data.

References

[1] R. Brown, S. Paez, G. Herrera, L. Chiruzzo, and A. Rosá, "Experiments on automatic error detection and correction for uruguayan learners of english," in *Swedish Language Technology Conference and NLP4CALL*, 2023, pp. 45–52.

[2] E. Volodina, C. Bryant, A. Caines, O. De Clercq, J.-C. Frey, E. Ershova, A. Rosen, and O. Vinogradova, "MultiGED-2023 shared task at NLP4CALL: Multilingual grammatical error detection," in *Proceedings of the 12th Workshop on NLP for Computer Assisted Language Learning*, D. Alfter, E. Volodina, T. François, A. Jönsson, and E. Rennes, Eds. Tórshavn, Faroe Islands: LiU Electronic Press, May 2023, pp. 1–16. [Online]. Available: https://aclanthology.org/2023.nlp4call-1.1/

[3] C. Bryant, Z. Yuan, M. R. Qorib, H. Cao, H. T. Ng, and T. Briscoe, "Grammatical error correction: A survey of the state of the art," *Computational Linguistics*, vol. 49, no. 3, pp. 643–701, 09 2023. [Online]. Available: https://doi.org/10.1162/coli_a_00478

[4] H. T. Ng, S. M. Wu, Y. Wu, C. Hadiwinoto, and J. Tetreault, "The CoNLL-2013 shared task on grammatical error correction," in *Proceedings of the Seventeenth Conference on Computational Natural Language Learning: Shared Task*, H. T. Ng, J. Tetreault, S. M. Wu, Y. Wu, and C. Hadiwinoto, Eds. Sofia, Bulgaria: Association for Computational Linguistics, Aug. 2013, pp. 1–12. [Online]. Available: https://aclanthology.org/W13-3601/

[5] H. T. Ng, S. M. Wu, T. Briscoe, C. Hadiwinoto, R. H. Susanto, and C. Bryant, "The CoNLL-2014 shared task on grammatical error correction," in *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task*, H. T. Ng, S. M. Wu, T. Briscoe, C. Hadiwinoto, R. H. Susanto, and C. Bryant, Eds. Baltimore, Maryland: Association for Computational Linguistics, Jun. 2014, pp. 1–14. [Online]. Available: https://aclanthology.org/W14-1701/

[6] C. Bryant, M. Felice, Ø. E. Andersen, and T. Briscoe, "The bea-2019 shared task on grammatical error correction," in *Proceedings of the fourteenth workshop on innovative use of NLP for building educational applications*, 2019, pp. 52–75.

[7] A. Masciolini, A. Caines, O. De Clercq, J. Kruijsbergen, M. Kurfalı, R. Muñoz Sánchez, E. Volodina, and R. Östling, "The MultiGEC-2025 shared task on multilingual grammatical error correction at NLP4CALL," in *Proceedings of the 14th Workshop on Natural Language Processing for Computer Assisted Language Learning*, R. Muñoz Sánchez, D. Alfter, E. Volodina, and J. Kallas, Eds. Tallinn, Estonia: University of Tartu Library, Mar. 2025, pp. 1–33. [Online]. Available: https://aclanthology.org/2025.nlp4call-1.1/

[8] M. Zeng, J. Kuang, M. Qiu, J. Song, and J. Park, "Evaluating prompting strategies for grammatical error correction based on language proficiency," in *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, N. Calzolari, M.-Y. Kan, V. Hoste, A. Lenci, S. Sakti, and N. Xue, Eds. Torino, Italia: ELRA and ICCL, May 2024, pp. 6426–6430. [Online]. Available: https://aclanthology.org/2024.lrec-main.569/

[9] C. Bryant and H. T. Ng, "How far are we from fully automatic high quality grammatical error correction?" in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, C. Zong and M. Strube, Eds. Beijing, China: Association for Computational Linguistics, Jul. 2015, pp. 697–707. [Online]. Available: https://aclanthology.org/P15-1068/

[10] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray *et al.*, "Training language models to follow instructions with human feedback," *Advances in neural information processing systems*, vol. 35, pp. 27 730–27 744, 2022.

[11] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale *et al.*, "Llama 2: Open foundation and fine-tuned chat models," *arXiv preprint arXiv:2307.09288*, 2023.

[12] A. Q. Jiang, A. Sablayrolles, A. Mensch, C. Bamford, D. S. Chaplot, D. de las Casas, F. Bressand, G. Lengyel, G. Lample, L. Saulnier, L. R. Lavaud, M.-A. Lachaux, P. Stock, T. L. Scao, T. Lavril, T. Wang, T. Lacroix, and W. E. Sayed, "Mistral 7b," 2023. [Online]. Available: https://arxiv.org/abs/2310.06825

[13] S. Nesmachnow and S. Iturriaga, "Cluster-uy: collaborative scientific high performance computing in uruguay," in *International Conference on Supercomputing in Mexico*. Springer, 2019, pp. 188–202.

[14] B. Swanson and E. Yamangil, "Correction detection and error type selection as an esl educational aid," in *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2012, pp. 357–361.

[15] H. Xue and R. Hwa, "Improved correction detection in revised esl sentences," in *Proceedings of the 52nd Annual Meeting of the Association*

for Computational Linguistics (Volume 2: Short Papers), 2014, pp. 599–604.

[16] M. Honnibal, “spacy 2: Natural language understanding with bloom embeddings, convolutional neural networks and incremental parsing,” *(No Title)*, 2017.

[17] S. Bird, E. Klein, and E. Loper, *Natural language processing with Python: analyzing text with the natural language toolkit.* ” O’Reilly Media, Inc.”, 2009.

[18] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “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)*, 2019, pp. 4171–4186.

APPENDIX

A. LLM Prompts

During the initial experiments, we followed an iterative approach to refine the prompt used to test the models. The first prompt gives a lengthy description of the task and the error types, and includes an example of expected output:

I’m going to give you texts written by an ESL student whose mother tongue is Spanish. Mark and correct the errors in the texts.
There are different types of errors:
When the spelling is wrong, the error is Spelling. Mark these errors with <E_S>and the ending with </E_S>. For example, ”<E_S>Thero</E_S>are three dogs.”
When the capital letter at the beginning of a sentence is used, the error is Beginning of Sentence. Mark these errors with <E_BOS>and </E_BOS>. For example, ”I like dogs. <E_BOS>she</E_BOS>likes cats.”
When the pronoun ”I” is written in lowercase, the error is Pronoun I. Mark these errors with <E_PI>and </E_PI>. For example, ”<E_PI>i</E_PI>like dogs.”
When a proper name is written without capitalization, the error is Proper Noun. Mark these errors with <E_PN>and </E_PN>. For example, ”I live in <E_PN>montevideo</E_PN>”.
When an article is missing, the error is Missing Article. Mark these errors with <E_MA>and </E_MA>. For example, ”She eats <E_MA>apple</E_MA>”.
When there is an unnecessary article like ”the” or ”a”, the error is Unnecessary Article. Mark the noun that has the unnecessary article with <E_UA>and </E_UA>. For example, ”She like a <E_UA>books</E_UA>.”
When an incorrect verb form is used, the error is Verb Form. Mark the wrong verb form with <E_VF>and </E_VF>. For example, ”She likes <E_VF>study</E_VF>.
When the error is agreement between subject and verb, the error is Subject-Verb Agreement. Mark these errors with <E_SVA>and </E_SVA>. For example, ”He <E_SVA>eat</E_SVA>apples.”
Here is an example how it should work: Original phrase: {{one_shot_example_original}}
The correction: {{one_shot_example_result}}
Mark the errors in the original sentence when there’s a difference between the original sentence and the corrected sentence.
This is the first text: {{input_text_to_correct}}
Mark the sentence with the errors you find and list the errors.

Given the large number of mistakes in the model results, the second prompt focused only on detecting the errors, without explaining the different types:

I’m going to give you texts written by an ESL student whose mother tongue is Spanish. List all the errors you find and what type of error they are and thengive me the sentence with the errors marked with the <error>and </error>tag.
This is the first text, mark all the errors you find:
Original phrase: {{one_shot_example_original}}
The correction with error tags: {{one_shot_example_result}}
Please mark the error tag as in the previous example in this original phrase: {{input_text_to_correct}}

The second approach had even worse results, so the next two approaches included a simplified version of the error types description in the prompt. In this occasion we tried with or without the inclusion of a one-shot example:

I’m going to give you texts written by an ESL student whose mother tongue is Spanish to correct. There are different types of errors:
1. <E_S>: Spelling Error. For example, ”<E_S>Thero</E_S>are three dogs.”
2. <E_BOS>: Beginning of Sentence Error(when sentences start with lowercase). For example, ”<E_BOS>she</E_BOS>likes cats.”
3. <E_PN>: Pronoun Error(when pronouns are in lowercase). For example, ”I live in <E_PN>montevideo</E_PN>”.
4. <E_MA>: Article Mistake. For example, ”She eats <E_MA>apple</E_MA>”.
5.<E_UA>: Unnecessary Article. For example, ”She like a <E_UA>books</E_UA>.”
6.<E_VF>: Verb Form Error. For example, ”She likes <E_VF>study</E_VF>.
7.<E_SVA>: Subject-Verb Agreement Error. For example, ”He <E_SVA>eat</E_SVA>apples.”
Here is an example how it should work: Original phrase: {{one_shot_example_original}}
The correction: {{one_shot_example_result}}
This is the first text: {{input_text_to_correct}}
Mark the sentence with the errors in the original phrase you find and list the errors.

The fifth iteration of the prompt simplifies the task description by removing the examples from all error types. This was the version that worked best at trying to solve the whole problem of detecting and classifying errors in one step:

This is a text written by an ESL student whose mother tongue is Spanish. There are different types of errors:
1. <E_S>: Spelling Error.
2. <E_BOS>: Beginning of Sentence Error(when sentences start with lowercase).
3. <E_PN>: Pronoun Error(when pronouns are in lowercase).
4. <E_MA>: Article Mistake.
5.<E_UA>: Unnecessary Article.
6.<E_VF>: Verb Form Error.
7.<E_SVA>: Subject-Verb Agreement Error.
This is the first text: {{input_text_to_correct}}
Give me the correct text first and then the list of errors.

When using the model just for creating a correct version of the text instead of trying to detect and classify the errors (as is the case with the Mistral experiments), we used a simple few-shot approach:

original: <start>{{few_shot_example_1_original}}<end>
corrected: <start>{{few_shot_example_1_result}}<end>
original: <start>{{few_shot_example_2_original}}<end>
corrected: <start>{{few_shot_example_2_result}}<end>
original: <start>{{few_shot_example_3_original}}<end>
corrected: <start>{{few_shot_example_3_result}}<end>
original: <start>{{input_text_to_correct}}<end>
corrected: <start>