How GEC works:

Grammatical error correction, or GEC, is a type of a Natural Language Processing problem that involves various kinds of errors in writing – be it a simple spelling mistake or incorrect tense. GEC models typically use a lot of training data – particularly, annotated data. Training a model to be able to correct, just for English, it would need to be able to understand a lot of specific syntax and generalizations or common rules of English. Some important and simple rules would be that the subject and verb have the same quantity, an article (such as "the") comes before nouns, or that it is uncommon for a word to appear twice in a row (and especially for certain words). However just understanding these kinds of statistics are not foolproof, as language is not just a simple set of rules and structure – this kind of understanding is known as "shallow." The model would not understand the sentence, just that words are typically arranged in a certain way.
https://www.grammarly.com/blog/engineering/under-the-hood-at-grammarly-leveraging-transformer-language-models-for-grammatical-error-correction/

Data Used to Train GEC models:

CoNLL's 2014 task set is the most commonly and often used dataset for testing the aptitude and quality of a given GEC model – it contains 1414 English essays covering various topics with explicit errors denoted by experts. The 28 sectioned error types include run-ons, incorrect noun count, missing a verb, redundancy, and even generally unclearness. Early attempts at training models on this data were only able to earn F0.5-scores of around 40%, which has since greatly improved since 2014. https://aclanthology.org/W14-1701.pdf

Performance of GEC models:

One of the highest achieved F-scores (F0.5) a model could achieve on this data was 71.12%, from a model made in 2023 – Muhammad Reza Qorib and Hwee Tou Ng's GRECO model, which finds the best subset of all edits made upon a given sentence. This dataset is useful as it provides a baseline amount of data that can be used to compare the relative effectiveness of multiple different models and approaches. http://nlpprogress.com/english/grammatical_error_correction.html

Another model by another team of Zuchao Li, Kevin Parnow, and Hai Zhao, made syntax parse trees which were then given to the model as input data. This approach has seen merit before with other NLP tasks – be it role labeling, sentiment analysis or machine reading comprehension. As syntax and grammar are closely linked to each other, they could be used as a basis for a GEC model that uses an encoder-decoder architecture. The encoder creates a vectorized representation of the input text – this being the parse tree that the decoder will use to create the final, corrected sentence. This model achieved an F score of 60.30%, which is higher than the baseline, but does have some room for improvement. Future development and expansion of using parse trees may lead to more results – though they do mention the usage of dependency parse trees may be a better fit for this kind of model. Using better parsers to make better parse trees results in better performance of the model.
https://www.sciencedirect.com/science/article/pii/S0306457322000206

A separate model made by Avinash Anand and several others finished training around a similar time was able to earn an F-score of 77% on the same training data which aims to better use

context as a part of the model – it uses a Dynamic Context Learner model to identify individual segments of a sentence in order to understand context and intent of a sentence.
https://link.springer.com/chapter/10.1007/978-3-031-49601-1_7