Grammatical Error Correction Using Large Language Models: A Case Study on Universal Dependencies Treebanks

University of Turku
Department of Computing
Master of Science (Tech) Thesis
Data Analytics
June 2025
Arvin Jalali

UNIVERSITY OF TURKU
Department of Computing

Arvin Jalali: Grammatical Error Correction Using Large Language Models: A Case Study on Universal Dependencies Treebanks

Master of Science (Tech) Thesis, 82 p., 11 app. p.
Data Analytics
June 2025

This thesis addresses Grammatical Error Correction (GEC) through two phases. The first phase investigates the use of Universal Dependencies (UD), a cross-linguistically consistent framework for syntactic annotation, particularly focusing on the `Typo=Yes` feature, to support error analysis in GEC. Tokens marked with `Typo=Yes` were extracted from three UD treebanks, including *UD English EWT*, *UD English GUM*, and *UD Finnish TDT*, and manually annotated based on the criteria of the ERRANT framework, which is designed to classify grammatical errors consistently. This enabled detailed cross-dataset and cross-linguistic error analysis.

The second phase evaluates the ability of a Large Language Model (LLM) to classify grammatical errors using structured prompts based on the ERRANT framework. Both zero-shot and few-shot prompting techniques were applied, and the LLM's performance was compared against manually annotated gold standards developed during the first phase. This work aims to bridge linguistic annotation frameworks and neural language models to advance GEC systems.

Keywords: Grammatical Error Correction, Universal Dependencies, ERRANT Framework, Large Language Models, Prompt Engineering

Preface

This thesis was completed as part of the MSc in Information and Communication Engineering, Data Analytics track, at the Department of Computing, University of Turku, Finland. It presents *Grammatical Error Correction Using Large Language Models: A Case Study on Universal Dependencies Treebanks.*

First and foremost, I would like to express my sincere gratitude to my supervisor, Dr. Jenna Kanerva, for her excellent guidance and support throughout this thesis project. Her supervision exceeded my expectations and made this research journey truly enriching. Working on such a novel topic under her direction allowed me to explore a wide range of valuable concepts and methodologies relevant to NLP.

I am also grateful to Professor Filip Ginter, whose NLP courses at the TurkuNLP group provided strong motivation and a solid foundation for this thesis.

My thanks also go to my friend, Dr. Behnam Badihi Olyaei, whose consistent encouragement served as a continuous source of motivation and trust during my studies.

Above all, I am sincerely thankful to my family for their unconditional emotional support across every stage of my life. I especially wish to thank my aunt, Farnaz Pashmforoosh, my uncle, Dr. Alireza Pashmforoosh, and my mother, Khadijeh Pashmforoosh, for their constant support and sacrifices.

Finally, I acknowledge the financial support provided by my late father, Hamid Jalali, which contributed to the completion of my academic studies in Turku.

Arvin Jalali

Turku, Finland, 6 June 2025

Contents

List of Figures

List of Tables

1 Introduction

1.1 Background

Grammatical Error Correction (GEC) is a vital subfield of Natural Language Processing (NLP) that focuses on automatically identifying and correcting grammatical errors in written text. Over the past decade, GEC methods have evolved from rule-based and classifier-based systems to statistical machine translation, and later to neural and transformer-based architectures. The emergence of Large Language Models (LLMs) has significantly improved the fluency and adaptability of GEC systems. However, these models often lack linguistic transparency and interpretability, making it difficult to trace how grammatical corrections are made or which linguistic rules are being applied. [1]

Meanwhile, Universal Dependencies (UD) provide a cross-linguistically consistent framework for dependency-based syntactic annotation, enabling the analysis of grammatical relations between words in a sentence [2]. This thesis explores how UD annotations, particularly the `Typo=Yes` feature [3], used to mark tokens with typographical or spelling errors, can support and enhance GEC, both in terms of error analysis and classification. Using UD as part of a GEC pipeline brings a layer of linguistic structure that many neural models currently lack, offering a more interpretable and linguistically grounded approach to error analysis.

This research is organized into two phases: the first focuses on error analysis using manually annotated error types from UD treebanks, and the second on prompt-based error classification with LLMs.

1.2 Research Questions and Motivation

Phase One

The first phase investigates the potential value of UD treebanks as a resource for GEC. The main research question is:

> *Can UD, particularly the* `Typo=Yes` *feature, be effectively utilized in GEC?*

To explore this, tokens marked with `Typo=Yes` were extracted from three UD treebanks (*UD English EWT*, *UD English GUM*, and *UD Finnish TDT*) and manually annotated according to the ERRANT framework. ERRANT is an automatic toolkit for identifying and classifying grammatical errors in English text, supporting consistent and efficient error annotation [4]. The manual annotations allowed for detailed cross-dataset and cross-linguistic error analysis on the selected UD treebanks. In addition, the inclusion of Finnish, with its rich morphology and syntactic variation, helps evaluate whether annotation frameworks and error classification strategies developed for English remain effective in a typologically different language. Overall, this phase highlights the value of UD annotations as a complementary resource for understanding and organizing grammatical errors in multilingual contexts.

Phase Two

Building on the manually annotated datasets from the first phase, the second phase evaluates the classification capabilities of an LLM. The research question guiding this phase is:

> *Can an LLM classify grammatical errors from the selected UD treebanks, based on structured prompts informed by the ERRANT framework, and how does its performance compare to manually annotated gold standards?*

This phase involves a prompt engineering experiment, applying both zero-shot and few-shot prompting techniques to guide an LLM in classifying grammatical errors inspired by the ERRANT framework.

1.3 Achievements of the Thesis

This thesis makes several key contributions and achievements. It conducts an in-depth literature review covering the foundations of GEC, key GEC datasets, the ERRANT framework, and the UD annotation scheme. Additionally, it manually annotates 542 tokens marked with the `Typo=Yes` feature, randomly selected from three UD treebanks (*UD English EWT*, *UD English GUM*, and *UD Finnish TDT*), according to the criteria of the ERRANT framework. While ERRANT was originally designed for English, it is adapted for Finnish to support cross-linguistic annotation. Furthermore, the thesis performs a cross-linguistic and cross-dataset analysis of grammatical error types, highlighting how language morphology and domain-specific features influence error distribution in GEC.

It also reviews the foundations of LLMs in the context of GEC, including core prompting strategies and relevant GEC-specific LLM methods. Moreover, a prompt-based experimental setup was designed and executed using zero-shot and few-shot prompting to evaluate how well an LLM can classify grammatical errors based on the ERRANT framework. Finally, LLM-based error classifications are compared against the manually annotated gold-standard datasets, assessing performance and identifying strengths and limitations of prompt-based error classification with LLMs.

These contributions aim to bridge the gap between linguistic annotation frameworks and neural language models, supporting the development of more interpretable and linguistically grounded GEC systems.

1.4 Structure of the Thesis

This thesis is organized into two main phases. The first phase investigates the use of UD as a resource for grammatical error analysis. It begins by reviewing the foundations of GEC, covering its evolution and challenges. Then, it introduces the `ERRANT` framework, which standardizes error classification for consistent analysis. This is followed by a survey of key GEC datasets. Next, the study examines the UD framework and its `Typo=Yes` annotation, which marks tokens with typographical and spelling errors. Using these annotations, an

overall 542 tokens from three UD treebanks were manually annotated with ERRANT error types. Finally, the phase evaluates the effectiveness of UD annotations for GEC tasks.

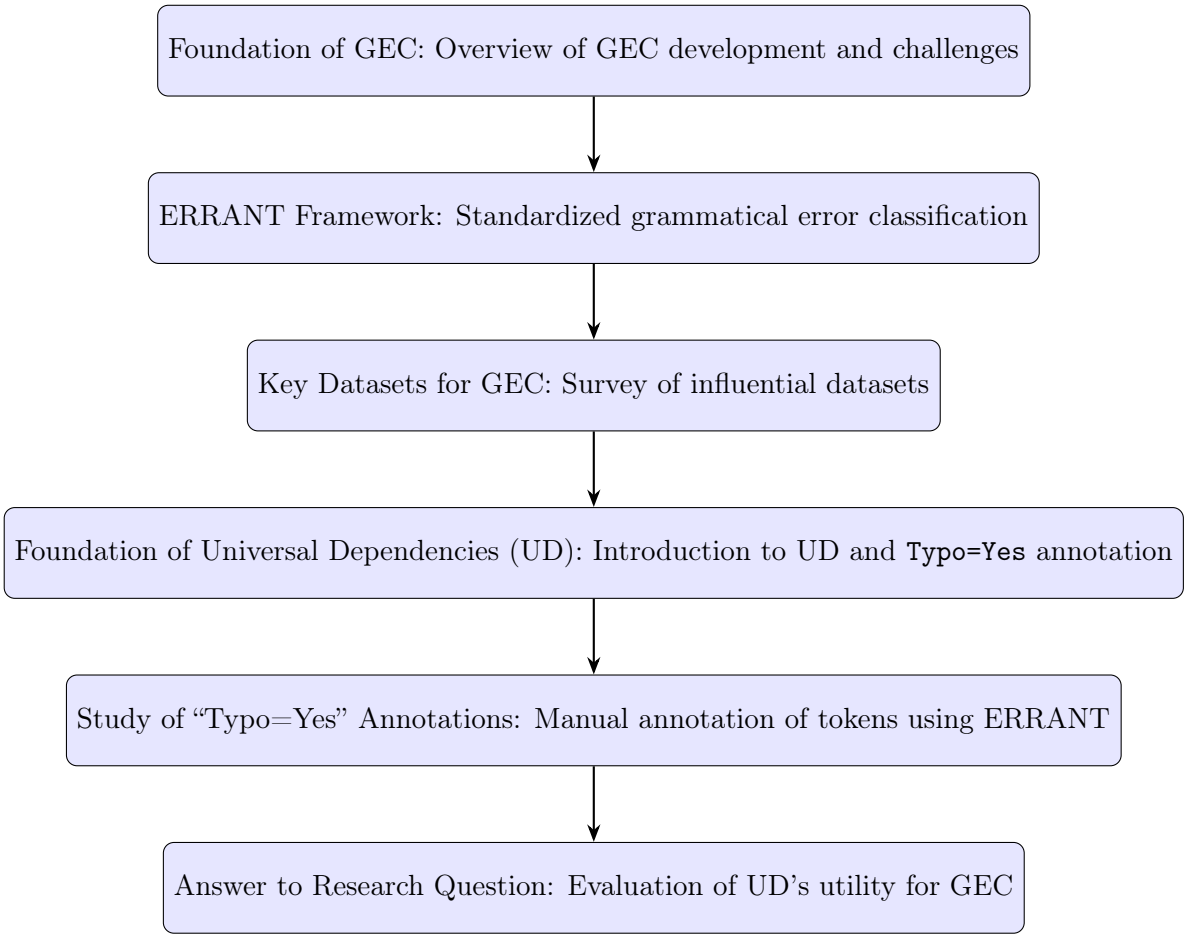

Figure 1.1: Structure and flow of the first phase of the thesis

The second phase focuses on LLMs and their capacity to classify grammatical errors. It begins with an overview of LLM architectures and recent GEC methods using these models. Then, it reviews prompting methods tailored to GEC. This leads to a prompt engineering experiment where zero-shot and few-shot prompts are tested for error classification accuracy.

Together, these phases integrate linguistic annotation and neural modeling approaches to advance the field of GEC.

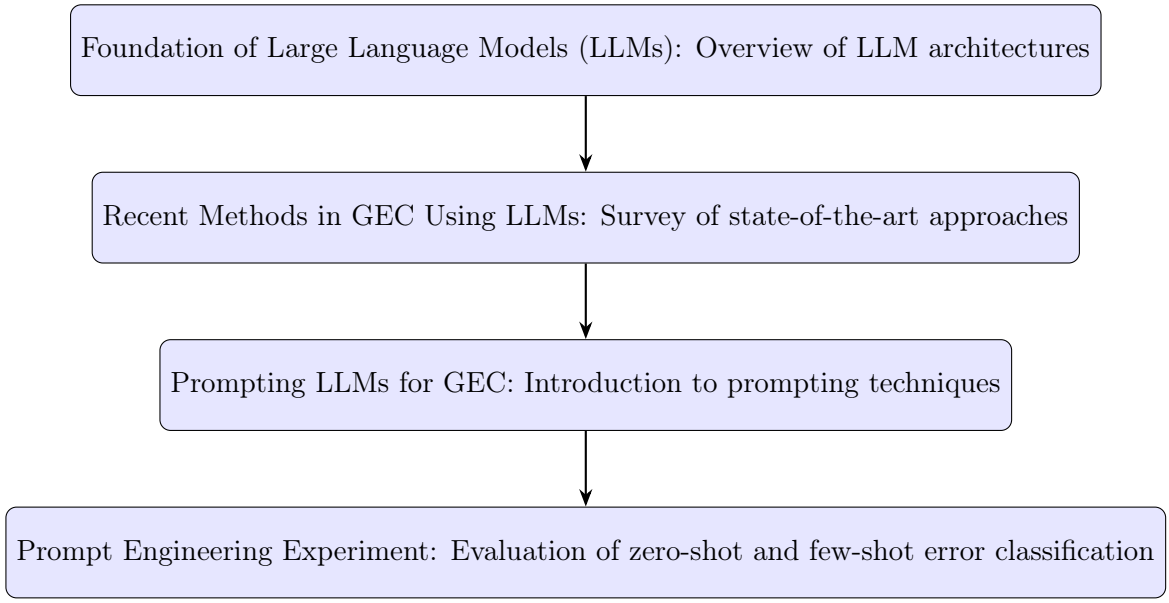

Figure 1.2: Structure and flow of the second phase of the thesis

2 Foundations of Grammatical Error Correction (GEC)

2.1 Evolution of GEC Approaches

This section is based on the work of Bryant et al. [1], who presented a comprehensive survey of Grammatical Error Correction (GEC), covering its evolution. GEC is the task of automatically detecting and correcting errors in written text. These errors can range from grammatical mistakes, such as incorrect verb tenses, subject-verb agreement errors, and article usage, to spelling mistakes, punctuation errors, and inappropriate word choices.

Over the past decade, GEC has evolved considerably, moving beyond traditional rule-based approaches toward more advanced neural network models capable of handling a broader variety of language errors. The key milestones in the evolution of GEC can be summarized as follows, based on the survey by Bryant et al. [1]:

Classifiers were among the earliest popular approaches in GEC, focusing on specific error types like articles, prepositions, and verb forms. They work by predicting corrections based on context features and using machine learning algorithms like naive Bayes, decision trees, or neural networks. While effective for errors with clear and limited correction options, classifiers struggle with errors related to open-class words (e.g., nouns, verbs, adjectives) and mistakes that depend on the context. They also assume only one error at a time, limiting their scalability for complex errors. This led to a shift towards more advanced methods.

Statistical Machine Translation (SMT) was the next significant advancement, treating

GEC as a machine translation task. SMT-based systems modeled the correction of errors as transforming incorrect sentences into grammatically correct ones. These approaches had a stronger ability to capture long-range dependencies compared to classifiers but still struggled with errors requiring more context or deeper understanding.

Neural Machine Translation (NMT) refers to the use of deep learning methods in GEC. With the growth of deep learning, Recurrent Neural Networks (RNNs), Convolutional Neural Networks (CNNs), and eventually Transformers became widely adopted for GEC tasks. These neural models demonstrated a strong capacity to learn long-range dependencies in text and handle complex errors. However, they require large parallel corpora for training and can suffer from overcorrection, generating unnatural sentences.

Edit-based approaches focus on applying a sequence of edits to the original sentence, rather than generating full sentences. This includes sequence tagging and sequence-to-sequence approaches. In the sequence tagging approach, each token in a sentence is assigned a tag specifying the required edit (e.g., "REPLACE", "KEEP"). This method is faster and more efficient than generating full sentences but can struggle with complex, multi-token edits. In the sequence-to-sequence approach, the model generates a series of specific edits, each identifying which part of the sentence to change and what to replace it with. However, it is slower than sequence tagging, though still faster than full sentence generation models. It offers better transparency and interpretability.

Language models for low-resource and unsupervised GEC have recently gained attention. Language models are algorithms trained on large corpora of text that learn to predict the next word in a sequence, making them effective for tasks like GEC. Unlike traditional approaches, these models do not require large parallel data but instead learn from monolingual or multilingual text. They can function as discriminators, where sentences with low probability (i.e., those that are grammatically incorrect) are corrected to form higher-probability, grammatically correct sentences. They can also function as generators, with modern Large Language Models (LLMs) such as GPT-3 [5] and PaLM [6] directly generating grammatically correct sentences from a given prompt.

Each GEC approach has contributed to the field's evolution, with newer methods

like NMT and language models addressing limitations of earlier techniques. However, these advances bring challenges such as data requirements, complexity, and interpretability. The current trend focuses on balancing these factors for more generalized, efficient systems.

2.2 Key Datasets Supporting GEC

The development and evaluation of GEC systems rely heavily on high-quality annotated datasets. These resources provide the foundation for training models, benchmarking performance, and ensuring consistent comparisons across studies. Key datasets have played an essential role in shaping the progress and direction of GEC research.

The CoNLL-2014 dataset [7], which consists of essays written by non-native English speakers and annotated with grammatical corrections, has served as a standard benchmark for many studies. Moreover, the BEA-2019 dataset [8] offers a broader range of error types as well as native and learner proficiency levels, making it suitable for evaluating systems across diverse contexts. Additionally, the JFLEG corpus [9], developed to complement existing GEC benchmarks, offers fluency-oriented corrections for sentences written by English learners, emphasizing both grammaticality and naturalness in the revised text.

These datasets have been valuable not only for training GEC models but also for establishing consistent benchmarks for system evaluation. The next chapter of this thesis provides a detailed overview of several important standard datasets and their roles in advancing GEC research.

2.3 Data Augmentation in GEC

This section is based on the work of Bryant et al. [1], who presented a comprehensive survey of GEC. A major challenge in GEC is the limited availability of large, high-quality parallel corpora. Existing resources are often either small to train robust models or contain noise, which can negatively affect performance. To address this challenge, data

augmentation methods have been proposed to generate synthetic data from monolingual corpora, which reduces the need for costly manual annotations. Two main methods for generating synthetic data are noise injection and back-translation.

In noise injection, errors are added to clean text using predefined rules or patterns from real error data. Rule-based methods include deleting, inserting, or changing words, while error pattern methods focus on mistakes commonly made by learners, making the errors more realistic.

Back-translation treats clean sentences as the correct version and generates incorrect ones by reversing GEC data. This approach, inspired by neural machine translation, often yields better performance than rule-based methods. Different techniques like sampling, noisy beam search, and edit-based generation have been used to improve data quality, and combining various model types (such as CNNs, LSTMs, and Transformers) can lead to even better performance.

In addition, round-trip translation can be used as a data augmentation method, where sentences are translated through a bridge language (e.g., English to Chinese and back) to introduce naturalistic errors.

Besides creating new data, the quality of existing datasets can be improved by removing or reducing the effect of incorrect corrections (noise reduction) and helping models handle errors more effectively (model enhancement). For example, some methods train the model by challenging it with difficult examples (adversarial training), or by teaching it to correct sentences in multiple steps, which can lead to higher accuracy.

In summary, data augmentation has played a key role in improving GEC systems by addressing the problem of limited data and enhancing model performance.

2.4 Evaluation Metrics in GEC

A crucial part of developing GEC systems is being able to measure how well they perform. In the prompt engineering part of this thesis, model performance was evaluated using four standard metrics: *accuracy*, *precision*, *recall*, and F_1*-score*. *Accuracy* measures the

proportion of correctly classified instances out of all cases. *Precision* is the ratio of true positive predictions to all positive predictions, reflecting the model's reliability in its positive outputs. *Recall* (sensitivity) is the ratio of true positives to all actual positives, indicating the model's ability to identify all relevant instances. F_1*-score* is the harmonic mean of precision and recall, balancing both metrics into a single performance indicator. [10]

In addition, several commonly used evaluation metrics in GEC are presented below.

MaxMatch ($\mathbf{M^2}$) is one of the most widely used evaluation metrics used in GEC. It compares the edits made by the system to those made by human annotators. It calculates precision (how many of the system's corrections are actually correct), recall (how many real errors the system managed to fix), and an F-score that combines both. In GEC, researchers often use the $F_{0.5}$ score, which gives more weight to precision. This is because it's usually more important for the system to avoid making unnecessary changes than to catch every possible error. [11]

ERRANT is another evaluation method that builds on the $\mathbf{M^2}$ scorer but adds more detail. It not only determines whether a correction is accurate but also identifies the type of error, like verb tense, spelling, or missing words. This allows researchers to better understand where a system is strong and where it needs improvement. ERRANT also supports different levels of detail, so evaluations can be broad or very specific, depending on what's needed. [4]

GLEU is a metric inspired by methods used in machine translation. Instead of focusing on individual edits, it compares the system's entire corrected sentence to human-corrected versions. It rewards overlap with the reference sentences and penalizes any unnecessary changes. GLEU is especially useful when there are multiple valid ways to fix a sentence, which is common in real-world language use. [12]

Overall, while these metrics are essential for evaluating GEC systems, they do have limitations. Human language is complex and subjective, so there's often more than one correct way to rewrite a sentence. As a result, some useful corrections might be scored unfairly. Still, these metrics offer a practical way to track improvements and compare

different systems.

2.5 Current Challenges and Future Directions of GEC

This section is based on the work of Bryant et al. [1], who presented a comprehensive survey of GEC. Despite major advances, GEC still faces several key challenges. A major challenge is domain generalization, where systems trained on one type of text often struggle to perform well on others. Developing systems that can adapt to a wider range of contexts remains a critical goal.

Another challenge is personalization. Learners have different native languages and proficiency levels, which influence the kinds of errors they make. Current systems are often trained on specific learner groups and may not generalize well to others. Building adaptable models that can serve diverse users is an important direction for future research.

Educational applications of GEC also highlight the need for systems to provide feedback, not just corrections. Users benefit more when they understand why something is wrong. This ties into the broader issue of interpretability; systems should be able to justify their suggestions in a way that humans can understand and trust.

There are also technical challenges. GEC systems often struggle with complex semantic errors, like misuse of idioms or collocations (words that naturally go together, like “make a decision” or “heavy rain”). Additionally, most systems operate at the sentence level, limiting their ability to fix context-dependent mistakes. Moving towards document-level models could significantly improve performance.

Current evaluation practices for GEC systems face limitations, as most metrics rely on exact matches to reference corrections, which may not fully capture system output quality. As a result, there is a growing need for more flexible and comprehensive evaluation methods that better align with real-world applications and user expectations.

In the future, developing multilingual and spoken GEC systems will be essential for broadening the applicability of these tools. While much of the research has focused on English, applying GEC techniques to other languages with distinct linguistic structures

presents a key opportunity for progress. Addressing these challenges will be essential for building more robust, inclusive, and effective GEC systems.

3 Key Datasets and Error Classification for GEC

3.1 Understanding ERRANT: A Framework for GEC

This section is based on the work of Bryant et al. [4], who introduced the ERRANT (ERRor ANnotation Toolkit) for automatic error annotation and detailed evaluation in GEC. The ERRANT was created to address a significant shortcoming in GEC systems which is the lack of detailed error type annotations. ERRANT identifies automatically edits between original and corrected sentences and classifies them according to a flexible and rule-based framework.

It is important to note that ERRANT was originally developed for English, with rules and classifiers specifically tailored to the syntactic and morphological structure of the English language. ERRANT is not tied to any specific dataset, allowing for error analysis at multiple levels of detail. It also reduces the workload for human annotators and brings more consistency to GEC datasets.

During the testing of ERRANT by experts, they rated the system's automatically generated edits as "Good" or "Acceptable" over 95% of the time. ERRANT was also successfully used in the CoNLL-2014 shared task [7], where it enabled a comprehensive first-time analysis of error types.

The toolkit sorts errors into 25 main categories, which are represented in Table 3.1. These 25 categories encompass a wide range of grammatical errors, each classified according to specific linguistic rules that consider part of speech, morphological patterns,

and other language properties.

Additionally, each error type can be categorized further with specific prefixes to indicate the nature of the error. "M" denotes a missing element, "R" represents a replacement error, and "U" identifies unnecessary elements.

This structured system enables more detailed analysis at various levels. For example, noun replacement errors are labeled as R:NOUN, indicating that an incorrect noun has been used and should be replaced with the correct one. Similarly, missing subject-verb agreement errors are marked as M:VERB:SVA, indicating that a necessary grammatical agreement between the subject and verb is absent.

3.2 An Overview of Key Datasets for GEC

GEC is an important area of natural language processing (NLP) that focuses on detecting and correcting errors in written text. Several standard datasets have already been developed specifically to train and evaluate GEC systems. Each dataset covers various error categories and certain languages. The following is an overview of some key datasets for GEC.

3.2.1 C4_200M Synthetic Dataset

This subsection is based on the work of Stahlberg and Kumar [13], who proposed using tagged corruption models for generating diverse and realistic synthetic data to improve GEC performance. The C4_200M synthetic dataset was developed by Google Research. It contains 200 million sentences generated through tagged corruption models. The main purpose is to generate synthetic data for GEC tasks to address the limitation of available annotated data. Previous synthetic data generation methods often fail to manage the diversity and complexity of errors in text. In contrast, the tagged corruption models used in this dataset utilize automatic error annotation tools such as ERRANT to ensure a diverse range of ungrammatical sentences generated from grammatically correct ones.

This approach involves giving a grammatically correct sentence and a specific er-

Code	**Meaning**	**Description / Example**
ADJ	Adjective	big → wide
ADJ:FORM	Adjective Form	Comparative or superlative adjective errors. goodest → best, bigger → biggest, more easy → easier
ADV	Adverb	speedily → quickly
CONJ	Conjunction	and → but
CONTR	Contraction	n't → not
DET	Determiner	the → a
MORPH	Morphology	Tokens have the same lemma but nothing else in common. rapid (adj) → rapidly (adv)
NOUN	Noun	person → people
NOUN:INFL	Noun Inflection	Count-mass noun errors. informations → information
NOUN:NUM	Noun Number	dog → dogs
NOUN:POSS	Noun Possessive	teachers → teacher's
ORTH	Orthography	Case and/or whitespace errors. Bestfriend → best friend
OTHER	Other	Errors that do not fall into any other category (e.g. paraphrasing). at his best → well, job → professional
PART	Particle	(look) in → (look) at
PREP	Preposition	of → at
PRON	Pronoun	ours → ourselves
PUNCT	Punctuation	! → .
SPELL	Spelling	genectic → genetic, color → colour
UNK	Unknown	The annotator detected an error but was unable to correct it.
VERB	Verb	ambulate → walk
VERB:FORM	Verb Form	Infinitives (with or without "to"), gerunds (-ing) and participles. to walk → walking, dancing → danced
VERB:INFL	Verb Inflection	Misapplication of tense morphology. getted → got, fliped → flipped
VERB:SVA	Subject-Verb Agreement	(She) have → (She) has
VERB:TENSE	Verb Tense	Includes inflectional and periphrastic tense, modal verbs and passivization. eats → ate, eats → has eaten, eats → can eat, eats → was eaten
WO	Word Order	only can → can only

Table 3.1: List of 25 main error types in ERRANT with examples and explanations, adapted from [4].

ror type tag to the model to generate a corresponding ungrammatical sentence. The C4_200M synthetic dataset covers 25 distinct categories of error types defined by the ERRANT annotation toolkit, which was explained earlier in this chapter. This results in the creation of a significant synthetic pre-training dataset with a balanced error tag frequency distribution. Hence, it reflects the variety and frequency of errors encountered in the actual text. In machine learning, a pre-training dataset typically refers to a dataset used for training a model in an unsupervised or semi-supervised manner before fine-tuning it on a task-specific dataset.

Experiments by Stahlberg and Kumar demonstrate considerable improvements in GEC performance when pre-training on the C4_200M synthetic dataset. Their work achieves state-of-the-art results on benchmark datasets such as CoNLL-2014, meaning that models pre-trained on the C4_200M perform very well when tested on CoNLL-2014. These synthetic datasets are publicly available, with the original English dataset hosted on GitHub and multilingual datasets described in Stahlberg and Kumar's 2024 publication.

3.2.2 CoNLL-2014 Shared Task

This subsection is based on the work of Ng et al. [7], who organized the CoNLL-2014 shared task to benchmark GEC systems through standardized evaluation.The CoNLL-2014 shared task dataset is a benchmark for GEC, meaning it is a standard reference used to evaluate the performance of GEC systems. This dataset consists of English essays written by non-native speakers and is used to test how well systems can identify and correct grammatical errors. It expands on CoNLL-2013 by increasing the error types from five to 28, covering verb tense, noun number, subject-verb agreement, prepositions, pronouns, word choice, sentence structure, and more.

The training data originates from the NUS Corpus of Learner English [14]. It includes 1,414 essays written by students at the National University of Singapore (NUS) who are non-native speakers of English. The essays are annotated manually by English instructors. The test data consists of 50 essays written by 25 non-native English speakers at NUS,

with each student contributing two essays. These essays are independently annotated for errors by two native English-speaking experts, providing a more robust and reliable evaluation standard.

CoNLL-2014 uses the F0.5 score, while CoNLL-2013 [15] uses the F1 score. The F0.5 score gives more weight to precision than recall, emphasizing the importance of making accurate corrections rather than just finding many possible errors. Additionally, an important part of the CoNLL-2014 evaluation is the MaxMatch (M^2) scorer. This metric compares system-generated edits to human corrections. Because there can be several valid ways to fix a sentence, the M^2 scorer provides a flexible way to judge how well the system matches human edits.

3.2.3 BEA-2019 Shared Task

This subsection is based on the work of Bryant et al. [8], who organized the BEA-2019 Shared Task, introducing a new evaluation benchmark and dataset to advance GEC research. The BEA-2019 Shared Task on GEC introduces several advancements, including the creation of a new dataset, the Write&Improve+LOCNESS corpus. This corpus represents a wider variety of native and learner English levels, allowing for more comprehensive evaluations. The task also introduces different tracks that control the amount of annotated data available to participants. Systems are evaluated using the ERRANT F0.5 score, providing detailed performance statistics.

The results of the BEA-2019 task show significant improvements in GEC over the past few years, particularly with the adoption of transformer-based models, compared to earlier tasks like CoNLL-2014. The top-performing systems demonstrate that competitive GEC models can be built even with limited annotated training data, achieving high performance in both tracks with restricted data and fewer resources. These findings, along with the new corpus, provide valuable benchmarks for ongoing and future GEC research.

3.2.4 JFLEG Corpus

This subsection is based on the work of Napoles et al. [9], who introduced the JHU Fluency-Extended GUG (JFLEG) corpus as a fluency-oriented benchmark for evaluating GEC systems. Their primary goal was to provide fluency edits that not only correct grammatical mistakes but also make sentences sound more native-like.

The corpus consists of 1,511 sentences from the GUG corpus. Each sentence was corrected four times by human annotators. These corrections cover a wide range of proficiency levels. They enable a more comprehensive evaluation of GEC systems by incorporating both minimal and fluency edits. JFLEG is considered a significant standard for developing GEC systems capable of making more extensive edits to produce fluent and natural-sounding English.

The JFLEG corpus categorizes errors into three types:

- **Awkwardness**: sentences that sound unnatural or unclear to native speakers.
- **Orthographic Errors**: mistakes in spelling and capitalization.
- **Grammatical Errors**: issues with number agreement, tense, word choice, etc.

In addition, edits are categorized as:

- **Minimal Edits**: small changes to correct specific grammatical or orthographic errors.
- **Fluency Edits**: broader sentence-level improvements to enhance naturalness and clarity.

Evaluations using the JFLEG corpus have shown that while modern GEC systems, especially those based on neural machine translation, perform better than traditional approaches, they still face challenges in producing fluent and natural-sounding text. Many systems tend to focus on minimal edits and struggle with making broader fluency improvements. This highlights the ongoing difficulty of achieving true fluency.

3.2.5 GitHub Typo Corpus

This subsection is based on the work of Hagiwara and Mita [16], who introduced the GitHub Typo Corpus. The GitHub Typo Corpus, originated from GitHub, is a comprehensive multilingual dataset. It consists of more than 350,000 edits and 65 million characters. The dataset covers more than 15 languages, with the majority in English, Chinese (Simplified) and Japanese. The GitHub Typo Corpus includes not only spelling errors but also naturally-occurring grammatical errors.

The types of edits covered in this dataset are presented in the following.

- Mechanical Errors: issues with punctuation or formatting that affect sentence structure.
- Spell Errors: mistakes in spelling due to incorrect or missing letters in words.
- Grammatical Errors: violations of language rules.
- Semantic Errors: changes in meaning caused by incorrect word choices or context.

A logistic regression classifier was used to identify typo edits (mechanical, spelling, and grammatical) versus semantic edits, achieving an F1 score of about 0.9. Considerably, many edits in non-English languages, such as Chinese and Japanese, involve English words. It indicates the significance of the dataset in multilingual and code-related error detection.

What distinguishes the GitHub Typo Corpus is its complexity. The existing spell checkers, such as Aspell (a free, open-source spell checker [17]) and Enchant (a generic wrapper library that provides a unified interface to multiple spelling engines[18]), struggle to perform effectively on the dataset. The average F0.5 score for these checkers is around 0.5, indicating the challenge of the typo edits.

In summary, the GitHub Typo Corpus goes beyond simple spelling errors by including a diverse mix of spelling, grammatical, and punctuation mistakes. Hence, it is considered to be an invaluable resource for advancing GEC systems.

3.2.6 cLang-8 Dataset

This subsection is based on the work of Rothe et al. [19], who introduced a simple multilingual GEC approach using large language models and released the cLang-8 dataset for improved training. The cLang-8 dataset, a modified version of the widely-used Lang-8 corpus [20], is essential for advancing multilingual GEC tasks. It uses a simple, language-independent method to generate synthetic examples and relies on large-scale multilingual models (up to 11 billion parameters). It achieves new state-of-the-art results in GEC benchmarks for four languages including English, Czech, German, and Russian. cLang-8 cleans the noisy Lang-8 dataset using the gT5 model, which simplifies the GEC training process. It only requires a single fine-tuning step with standard language models, resulting in improved accuracy.

The training set includes 2.4 million English examples, along with smaller portions for German (114K) and Russian (45K). Models trained on cLang-8 significantly outperform those trained on the original Lang-8, with larger models showing consistent improvement across all error types. By releasing cLang-8, the dataset provides researchers with a cleaner, more reliable resource, making the training of GEC models more efficient and effective.

3.2.7 MultiGEC-2025 Shared Task

This subsection is based on the work of Masciolini et al. [21], who organized the MultiGEC-2025 shared task aimed at advancing multilingual GEC across a diverse set of languages. As part of this initiative, the authors introduced the MultiGEC-2025 dataset, which focuses on GEC at the text level rather than the sentence level. Unlike earlier efforts that primarily targeted isolated sentence corrections, this task emphasizes the evaluation of GEC systems on full-length texts, thus reflecting more realistic and complex language use scenarios.

The dataset includes twelve European languages: Czech, English, Estonian, German, Greek, Icelandic, Italian, Latvian, Russian, Slovene, Swedish, and Ukrainian, making it one of the most linguistically diverse resources in the field of GEC. It brings together

seventeen subcorpora from various sources, such as learner essays and online content, and organizes them into a consistent format for training and evaluating GEC systems.

The shared task offers two different aspects: one for minimal edits focusing strictly on grammatical correctness, and another for fluency edits aiming for more natural and idiomatic phrasing.

The MultiGEC-2025 shared task uses three core metrics to evaluate system output: ERRANT [4], GLEU [12], and Scribendi [22]. ERRANT measures grammatical accuracy using the F0.5 score, which favors precision over recall to discourage unnecessary corrections. GLEU assesses how closely the system output matches reference texts in terms of both content and fluency. Scribendi estimates the overall fluency and correctness of the text using a language model, making it suitable for evaluating outputs across different languages.

To conclude, the MultiGEC-2025 shared task represents a significant advancement in the multilingual GEC field, offering valuable resources and evaluation methods for diverse languages and correction objectives.

3.2.8 ErAConD

This subsection is based on the work of Yuan et al. [23], who introduced ErAConD (Error Annotated Conversational Dialog Dataset), the first GEC dataset designed for conversational dialogue with error impact annotations. ErAConD aims for GEC in open-domain chatbot conversations. While the existing GEC datasets focus mostly on essays, ErAConD focuses on human-machine dialogues, particularly for second language learners. The dataset consists of 186 dialogues with 1,735 user turns, and it contains various grammatical errors.

Errors are categorized into three levels based on how much they impact comprehension. This categorization assists in assessing the severity of grammatical errors and their effect on written communication.

- The first level includes minor mistakes, such as incorrect capitalization and punctuation (except apostrophes), which generally have slight to no impact on under-

standing.

- The second level covers errors with a moderate impact, like the improper use of acronyms, abbreviations, non-English internet slang, and apostrophes. While these can cause slight confusion, they do not change the intended meaning.

- The third level consists of more serious mistakes, such as subject-verb agreement errors, incorrect verb forms, and word confusion, which can significantly alter meaning and lead to misunderstandings.

In order to evaluate the effectiveness of ErAConD, GECToR, as a leading GEC model, was fine-tuned on this dataset with a focus particularly on third-level errors. The results indicated that this fine-tuning considerably improved the model's precision, reducing false positives and providing more accurate corrections. In conclusion, ErAConD represents a progressive step for GEC models, particularly for conversational contexts.

4 Foundations of Universal Dependencies (UD)

4.1 Introduction to UD

Universal Dependencies (UD) [2] is an open community project aimed at developing a cross-linguistic and consistent framework for dependency-based syntactic annotation. UD can be considered as a method of representing the grammatical structure of a sentence by showing how words relate to one another. The project standardizes linguistic annotations across various languages. It also enhances multilingual NLP and facilitates broader linguistic research. UD annotations include tokenization (splitting text into smaller units like words or subword units), morphological tagging (assigning grammatical features like tense, number, or part-of-speech to words), and syntactic dependencies (relationships between words that represent their roles in sentences, such as subject, object, etc). UD ensures a universal yet adaptable structure. The goal is to facilitate multilingual parser development (creating software that automatically analyzes and parses the syntactic structure of sentences), cross-lingual learning (learning from data in one language and applying it to another), and parsing research from the perspective of language typology (the study of cross-linguistic patterns and structures). UD aims to provide a universal set of categories and guidelines in order to guarantee consistent annotation among similar linguistic structures across languages. However, UD allows some necessary adaptations for specific languages.

UD is considered effective if it balances the following key factors: [24]

- It must be reliable for linguistic analysis of all individual languages.
- It should provide a solid foundation for linguistic typology (the classification of languages based on common features), enabling the identification of cross-linguistic parallels across languages and language families.
- The annotation process needs to be fast, efficient, and consistently applied by human annotators.
- The framework should be understandable enough for non-linguists such as language learners or engineers by applying familiar grammatical concepts, terminology, and structures.
- It must achieve high-accuracy computational parsing (using algorithms to analyze sentence structures with great precision).
- It should strongly support downstream NLP tasks, including relation extraction (identifying and classifying relationships between entities in text), reading comprehension, and machine translation.

Improving UD in one of the above key factors is relatively simple; however, the real challenge is making improvements without disrupting the developed balance between them.

A major step in the project's development was the transition from version 1 to version 2 of UD [25]. This update improved annotation consistency and expanded language coverage. Key changes included better tokenization rules for multiword expressions (phrases that are treated as a single unit, like "New York" or "high school") and clearer grammatical distinctions in morphological annotations. Moreover, syntactic dependency representation (the way relationships between words are represented syntactically) was improved. Furthermore, UD version 2 introduced guidelines for enhanced dependencies, aiming to provide richer syntactic details to support semantic understanding (understanding the meaning of sentences) in NLP. Despite these advancements, the project continues to grow. It focuses on expanding language representation, increasing dataset sizes, and

maintaining consistency across languages. Today, the UD framework is a significant resource in computational linguistics, supporting multilingual text analysis and AI-driven language processing.

4.2 Understanding the "Typo=Yes" Annotation in UD Version 2

The `Typo=Yes` annotation in UD is used to mark words that contain spelling mistakes, typographical errors, or unexpected character usage. This includes common mistakes such as misspelled words (e.g., Barak Obama instead of Barack Obama), errors in word choice (lesser instead of fewer), and mispronunciations that are transcribed as they sound (e.g., shilly instead of silly in spoken language data). [3]

It can also include cases where special characters or spaces are used in an unusual and creative way for the purpose of visual effect, as well as errors caused by incorrect character encoding. However, abbreviations, informal spellings, and stylistic variations are not necessarily considered typos unless they are truly unexpected in the context. The handling of unexpected capitalization and stylistic elements depends on the language and the specific treebank. For example, in social media text, inconsistent capitalization may not always be considered a typo due to the inherently informal way of writing. [3]

Furthermore, when a word is mistakenly split into several parts due to extra spaces, the goeswith relation is used to connect parts of the word together, and the `Typo=Yes` annotation is assigned to the main part of the word. [26]

The `Typo=Yes` annotation is focused specifically on spelling errors and does not apply to cases where an extra word is mistakenly added to a sentence, or even when deliberately altering the spelling for stylistic reasons such as using symbols to censor profanity. [3]

UD handles typos and other textual errors in a way to preserve original errors for linguistic modeling purposes while marking them for correction. When a word is misspelled, the correct spelling is stored in the MISC column as "CorrectForm", while the `Typo=Yes` annotation is assigned to mark an error. [26]

4.3 Understanding the CoNLL-U Format

The CoNLL-U format [27], a modified version of the CoNLL-X format, is a standardized text format used to represent linguistic annotations in UD treebanks. Each word in a sentence is represented by a row, with ten fields separated by tabs, containing key linguistic features such as the word's form, lemma, part-of-speech tag, and syntactic relationships. These fields are defined as the following:

1. **ID**: A unique identifier indicating the word's position in the sentence. Multiword tokens use a range, while empty nodes use decimal values (lower than 1 and greater than 0).
2. **FORM**: The word as it appears in the sentence.
3. **LEMMA**: The dictionary or base form of the word (e.g., studying → study).
4. **UPOS**: The universal part-of-speech tag (e.g., noun, verb, adjective).
5. **XPOS**: A language-specific part-of-speech tag (optional).
6. **FEATS**: Additional grammatical features (e.g., number, tense, case).
7. **HEAD**: The ID of the word's syntactic parent in the sentence (0 if it is the root).
8. **DEPREL**: The grammatical relationship between the word and its parent (e.g., subject, object).
9. **DEPS**: Enhanced dependency relations, allowing for more complex syntactic representations (optional).
10. **MISC**: Miscellaneous annotations, such as metadata or comments.

Sentences in a CoNLL-U file are separated by blank lines, and comments (marked with a hash) typically appear at the beginning of a sentence. Most fields (except FORM, LEMMA, and MISC) cannot contain spaces, and some fields, such as UPOS, HEAD, and DEPREL, must always be specified except in multiword tokens or empty nodes. The

format also supports multi-word tokens and empty nodes, allowing for flexible representation of linguistic structures across different languages. A notable feature of CoNLL-U is its ability to annotate typographical errors using the `Typo=Yes` label in the FEATS field. This enables straightforward identification and analysis of such errors within a UD treebank.

4.4 Previous Research on the Use of UD Treebanks for GEC

The Treebank of Learner English (TLE) [28] is the first publicly available syntactic treebank designed for English as a Second Language (ESL). It includes over 5,124 sentences from the Cambridge First Certificate in English (FCE) corpus [29], each annotated with part-of-speech tags and UD structures. Importantly, each sentence is provided in both its original, potentially ungrammatical form and its corrected version, allowing for detailed comparison. It introduces annotation guidelines tailored to learner language, ensuring consistency in handling ungrammatical constructions.

In addition, it presents parsing benchmarks on the dataset and examines how grammatical errors impact parsing accuracy. This resource is intended to support research in second language acquisition and the computational analysis of ungrammatical language.

TLE does not primarily aim to use UD for GEC, nor does it include tokens annotated with `Typo=Yes`. However, it remains highly relevant to GEC research for several reasons:

- It contains parallel annotations of both original (ungrammatical) and corrected learner English sentences, providing valuable data for studying grammatical errors.
- The availability of UD syntactic structures for both versions allows analysis of how errors impact sentence syntax, which is crucial for developing syntax-aware GEC models.
- Although not designed to propose or evaluate GEC methods directly, the dataset offers a foundational resource that supports research into improving grammatical

error correction through syntactic information.

In summary, while TLE is not a GEC system itself, its parallel UD annotations of learner language make it a valuable resource for exploring syntactic approaches to grammatical error correction.

5 A Study of "Typo=Yes" Annotations in UD Treebanks Using the ERRANT Framework

5.1 Automated Detection of "Typo=Yes" Tokens in UD Treebanks

A Python code was developed to download and analyze CoNLL-U files from various repositories hosted on GitHub by UD. The primary objective was to identify tokens annotated with `Typo=Yes` to assess the prevalence of them in UD treebanks.

The code starts by authenticating with the GitHub API using a personal token. It then retrieves a list of UD repositories, filtering for those with names containing "UD_", which typically indicate UD treebanks. For each relevant repository, the script checks for CoNLL-U files, downloads those that are available, and processes them to identify and count tokens marked with `Typo=Yes`. The results are stored in a Comma-Separated Values (CSV) file for further analysis.

The analysis identified 302 repositories with downloadable files, of which 95 contained at least one instance of the `Typo=Yes` annotation. The 10 UD treebanks with the highest number of tokens marked as `Typo=Yes` are listed in Table 5.1.

The results reveal significant variation in the prevalence of `Typo=Yes` annotations across different UD treebanks, highlighting areas where further attention may be required to improve data quality within the UD framework.

Treebank	Typo=Yes Tokens
UD_English-EWT	1279
UD_Estonian-EWT	1110
UD_Russian-Taiga	1102
UD_Portuguese-DANTEStocks	1088
UD_Spanish-GSD	982
UD_French-GSD	930
UD_Latvian-LVTB	431
UD_Romanian-TueCL	430
UD_Finnish-TDT	396
UD_English-GUM	362

Table 5.1: Top 10 UD treebanks with the highest number of `Typo=Yes` tokens

5.2 Selection of UD Treebanks for "Typo=Yes" Annotation Analysis

The author of this thesis is a native Persian speaker with an advanced proficiency in English and basic understanding of Finnish grammar. Given this linguistic background, the analysis is limited to languages for which adequate linguistic knowledge is available, specifically English and Finnish. Since the Persian UD treebanks do not contain any tokens marked with the `Typo=Yes` annotation, the analysis focuses on English and Finnish.

The UD treebanks selected for further analysis were prioritized based on two factors: the languages that can be worked with, and the number of tokens annotated with `Typo=Yes`. This prioritization ensures that the analysis is both feasible and meaningful, focusing on languages with sufficient data to support the study. The following three UD treebanks were chosen for analysis: UD English EWT, UD Finnish TDT, and UD English GUM, with 1279, 396, and 362 tokens annotated with `Typo=Yes` respectively. The following subsections offer a short overview of each UD treebank to give some context about their content and structure.

5.2.1 UD English EWT

The UD English EWT treebank [30] contains 1,279 tokens annotated with `Typo=Yes`, making it the top priority for this analysis. It is a gold-standard UD corpus built on the English Web Treebank LDC2012T13 [31] and comprises 254,820 words across 16,624 sen-

tences. The dataset includes texts from five types of online sources (weblogs, newsgroups, emails, reviews, and Yahoo! answers) to provide diverse linguistic material. While most dependency annotations were single-annotated, a smaller portion received double annotation and further revision to enhance consistency. This treebank offers a rich and varied dataset for investigating typographical annotation in English.

5.2.2 UD Finnish TDT

The UD Finnish TDT treebank [32] contains 396 tokens annotated with `Typo=Yes` and is based on the Turku Dependency Treebank (TDT) [33], a broad-coverage resource for general Finnish. It features a wide range of genres, including Wikipedia articles, Wikinews articles, university online news, blog entries, student magazine articles, grammar examples, Europarl speeches, JRC-Acquis legislation, financial news, and fiction, all sourced from 674 individual documents. The conversion to UD was carried out with precise manual checks and corrections to ensure consistency with UD guidelines. This makes UD Finnish TDT a valuable resource for investigating typographical errors in Finnish as a language characterized by its rich morphology and syntactic complexity.

5.2.3 UD English GUM

The UD English GUM treebank includes 362 tokens annotated with `Typo=Yes` and is part of the Georgetown University Multilayer (GUM) corpus. This corpus is developed by students in the Computational Corpus Linguistics course at Georgetown University. It features a variety of text types including academic, blog, email, fiction, government, legal, news, nonfiction, social, spoken, web, and wiki. The content is sourced from publicly available materials under Creative Commons licenses. The corpus is designed to reflect different communicative purposes in modern English, especially in informal contexts such as social media and blogs. The GUM treebank thus offers a valuable dataset for studying typographical errors in contemporary, informal English. [34]

5.3 Adopting an Error Classification Scheme for Annotating "Typo=Yes" Tokens in UD Treebanks

To deepen the study and analyze the `Typo=Yes` annotations in the three selected UD treebanks, we extend the investigation by manually annotating each `Typo=Yes` token with a specific grammatical error type. This requires a consistent error classification scheme to guide the annotation process. The selection of an appropriate scheme is crucial to ensure that errors are categorized consistently with the linguistic structure and annotation goals of UD.

In Chapter 3, two error classification schemes relevant to this study were introduced:

- **ERRANT**: A tool for automatically identifying and categorizing grammatical errors at the sentence level. It defines 25 distinct error categories and is particularly suited for dependency-based syntactic analysis. [4]
- **CoNLL 2014 Shared Task**: A benchmark for evaluating GEC systems, especially in texts written by second language learners. It defines 28 error categories focused on a wide range of learner language issues. [7]

While both are practical for GEC, they differ in focus and application, as discussed below.

For the purposes of this study, ERRANT is chosen as the most suitable error classification scheme, particularly when working with the UD treebanks. In UD, the `Typo=Yes` annotation is specifically used to mark spelling mistakes, typographical errors, and character-level issues such as misused or missing characters, incorrect word forms due to typing errors, or unexpected spacing. These are closely aligned with the types of surface-level errors ERRANT is capable of classifying through its 25 error types. Moreover, ERRANT's focus on sentence-level analysis and syntactic structure makes it a compatible tool for use with UD treebanks. In addition, ERRANT is widely adopted as an evaluation metric in GEC systems, further demonstrating its reliability and relevance. In contrast, the CoNLL 2014 Shared Task, while comprehensive with its 28 error categories,

is primarily designed for learner language and is less suitable for handling typographical annotations found in UD.

5.4 Sentence Extraction and Manual ERRANT-Based Annotation of "Typo=Yes" Tokens

A Python code was developed to automatically identify and extract sentences annotated with `Typo=Yes` from the selected UD treebanks. The code fetches .conllu files from the treebanks including UD English EWT, UD English GUM, and UD Finnish TDT. It extracts sentences that include tokens annotated with the `Typo=Yes` feature and collects both their metadata and token lines. The resulting typo-annotated sentences are shuffled to ensure randomization and stored in three separate .conllu files corresponding to each treebank, serving as the basis for subsequent manual ERRANT annotation.

As a result, 240 tokens were selected for English EWT, 202 tokens for English GUM, and 100 tokens for Finnish TDT. These tokens were randomly sampled and manually annotated according to the 25 error types and three core operation labels, M (Missing), U (Unnecessary), and R (Replacement), as defined by the ERRANT framework. The resulting annotations were stored in three separate `.conllu` files, each corresponding to one of the UD treebanks.

For the task of annotation, Visual Studio Code (VS Code) was used, supported by several extensions including vscode-conllu, NLP Visualization, and Rainbow CSV, which facilitated efficient inspection and editing of the .conllu files.

An example of a sentence extracted from the UD English EWT and annotated for this study is shown below:

```
# sent_id = reviews-343813-0002
# newpar id = reviews-343813-p0002
# text = Definately won't be returning.
1    Definately    definitely    ADV    RB    Typo=Yes    5    advmod
5:advmod  CorrectForm=Definitely|ERRANT=R:SPELL
```

```
2-3    won't    _    _    _    _    _    _    _    _
2    wo    will    AUX    MD    VerbForm=Fin    5    aux    5:aux
3    n't    not    PART    RB    Polarity=Neg    5    advmod    5:advmod
4    be    be    AUX    VB    VerbForm=Inf    5    aux    5:aux
5    returning    return    VERB    VBG    Tense=Pres|VerbForm=Part    0
root    0:root    SpaceAfter=No
6    .    .    PUNCT    .    _    5    punct    5:punct
```

In this example, the first token, `Definately`, is annotated with the `Typo=Yes` feature, indicating a typographical error. The `CorrectForm` field specifies the intended form as `Definitely`. According to the UD, the token itself is categorized as an adverb (`UPOS=ADV`) with the lemma `definitely`. Additionally, a manual annotation field `ERRANT=R:SPELL` is added to align with the ERRANT framework. This label indicates that the error falls under the category of `Replacement` (`R`) and specifically identifies a `Spelling` (`SPELL`) error, which is one of the 25 core error types defined by ERRANT. This enriched annotation scheme facilitates detailed error analysis in the context of GEC.

5.5 Creating a Structured Dataset from the Annotated UD Treebanks

This section focuses on converting the ERRANT-annotated `.conllu` files into structured datasets, designed to offer insights into the distribution of various ERRANT error categories among tokens annotated with the `Typo=Yes` feature.

The datasets are constructed by extracting useful information from the `.conllu` files. For each token annotated with the `Typo=Yes` feature, the following data are extracted and organized into the dataset:

- **sent_id**: The sentence identifier in a UD treebank, extracted from the metadata in the `.conllu` (e.g., `# sent_id = ...`).
- **token_id**: The unique identifier of the token, corresponding to its position in the sentence.

- **token_form**: The ungrammatical form of the token (e.g., `reccommend`).
- **CorrectForm**: The intended correction of the token, specified in the `CorrectForm` attribute (e.g., `CorrectForm=Definitely`).
- **ERRANT**: Manual annotation based on the ERRANT framework (e.g., `ERRANT=R:SPELL`).

This dataset serves as the foundation for further analysis, enabling the examination of the frequency and distribution of various ERRANT error types.

5.6 Categorization of ERRANT Error Types into Broader Categories

The 25 error types defined in the ERRANT framework have been re-categorized into six broader categories. This categorization, carried out by the author of this thesis, was done to provide a more general and insightful overview of the errors, grouping them according to their linguistic relevance. The six categories are as follows:

- **Nominal Errors**: Includes noun, noun inflection, noun number, and noun possessive.
- **Verbal Errors**: Includes verb, verb form, verb inflection, subject-verb agreement, and verb tense.
- **Modifier and Morphological Errors**: Includes adjective, adjective form, adverb, and morphology.
- **Function Word Errors**: Includes pronoun, conjunction, preposition, determiner, and particle.
- **Orthographic and Structural Errors**: Includes spelling, orthography, and punctuation.
- **Miscellaneous and Undefined Errors**: Includes other and unknown.

This categorization allows for a structured analysis of the ERRANT error types found in the annotated treebanks. It enables deeper insights into the types of errors that occur across different languages and UD treebanks.

5.7 Distribution of ERRANT-Annotated Errors across UD Treebanks

With the structured datasets derived from the ERRANT-annotated `.conllu` files, the next step is to examine the distribution of error types across three selected UD treebanks: `UD English EWT`, `UD English GUM`, and `UD Finnish TDT`.

The analysis reveals that error distribution varies across the three datasets. In the case of the UD English EWT dataset, there is a high frequency of **Orthographic and Structural Errors** (186 instances), with frequent occurrences of spell (`SPELL`: 88), contraction (`CONTR`: 65), and orthography (`ORTH`: 29). This dataset also contains a noticeable amount of **Nominal Errors** (22 instances), driven primarily by noun possession (`NOUN:POSS`: 21). Meanwhile, **Verbal Errors** are relatively limited (16 instances), appearing in smaller quantities across subtypes such as verb form (`VERB:FORM`: 7) and subject-verb agreement (`VERB:SVA`: 5). These patterns suggest a concentration of structural errors within UD English EWT.

The UD English GUM dataset, by comparison, presents a more balanced and varied error profile. It includes a substantial number of **Verbal Errors** (34 instances), with common subtypes including subject-verb agreement (`VERB:SVA`: 10) and verb form (`VERB:FORM`: 11). **Function Word Errors** are also prominent (37 instances), with significant counts for pronoun (`PRON`: 13), determiner (`DET`: 10), and preposition (`PREP`: 11). Although **Orthographic and Structural Errors** (95 instances) occur less frequently than in EWT, they remain a key category, especially in spell (`SPELL`: 38) and orthography (`ORTH`: 28). **Nominal Errors** (24 instances) are also present, primarily reflecting noun number (`NOUN:NUM`: 19). This distribution indicates a broader range of grammatical errors within UD English GUM.

In contrast, the UD Finnish TDT dataset reveals a distinct distribution of errors. **Orthographic and Structural Errors** are again dominant (58 instances), especially in spell (`SPELL`: 38) and orthography (`ORTH`: 17). A noteworthy aspect of this dataset is the presence of **Modifier and Morphological Errors** (5 instances), led by adjective form (`ADJ:FORM`: 3). **Nominal Errors** (14 instances) are largely due to noun inflection (`NOUN:INFL`: 10), which reflects the morphological complexity of Finnish. **Verbal Errors** appear in a volume comparable to the English datasets (16 instances), mainly involving verb form and verb inflection. Overall, the TDT dataset highlights language-specific challenges, particularly those arising from the complex morphology of Finnish.

To visually summarize these findings, the following figures illustrate the frequency of ERRANT error types and broader ERRANT error categories across the three UD treebanks. Figure 5.1 displays the frequency of specific error types, while Figure 5.2 groups these into higher-level error categories for comparative analysis.

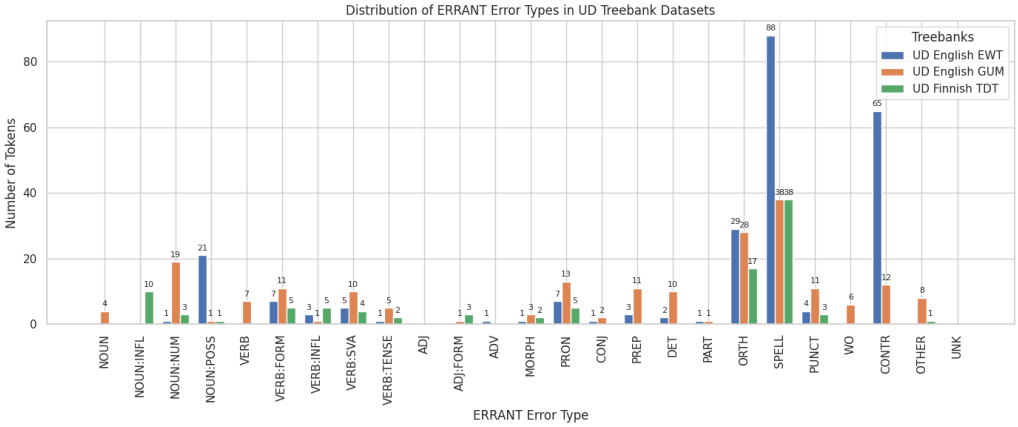

Figure 5.1: Distribution of ERRANT errors in UD treebank datasets

5.8 Linguistic Justification for Different Error Patterns in UD Finnish TDT

The distribution of errors observed in the UD Finnish TDT dataset can be explained by the typological and morphological features of the Finnish language. In this context, morphology refers to the structure and form of words, specifically, how they change

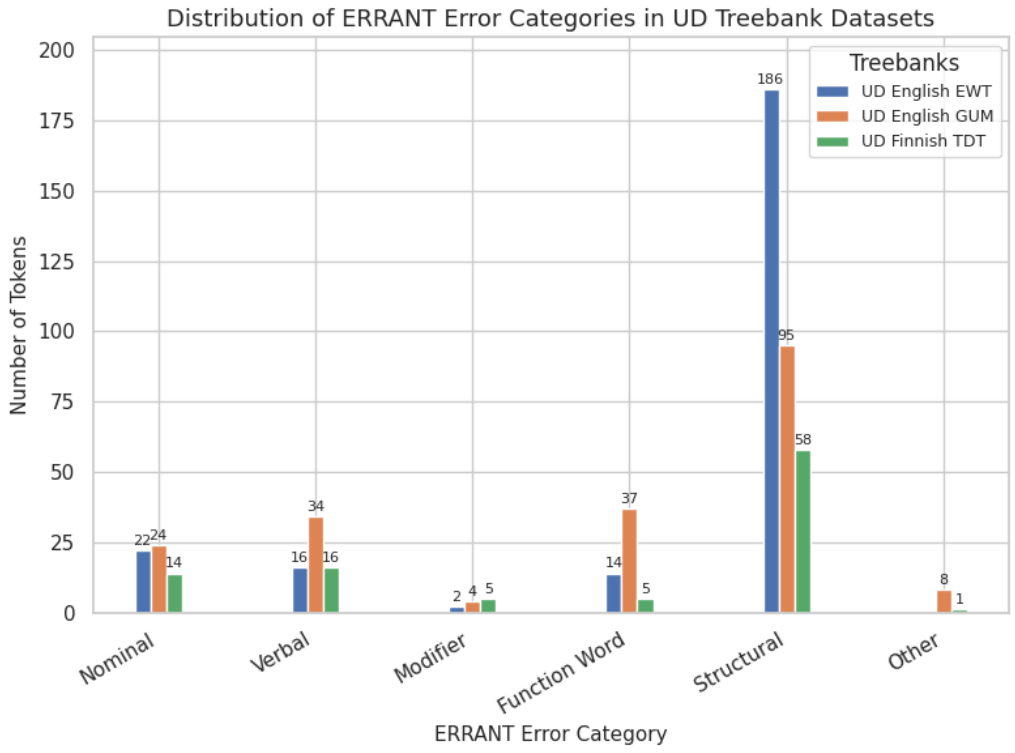

Figure 5.2: Distribution of ERRANT error categories in UD treebank datasets

to express different grammatical meanings such as tense, case, number, or possession. Finnish is a morphologically rich and agglutinative language, meaning that words are often formed by adding a series of suffixes to a root word to indicate various grammatical functions [35]. This results in long, complex word forms that can pose challenges for language learners and users. Compared to languages like English, which have simpler morphological structures, Finnish tends to produce distinct error patterns.

Nominal Errors: The relatively high number of Nominal Errors (14 instances) stems mainly from noun inflection (`NOUN:INFL`: 10). Finnish nouns are inflected for more than fifteen grammatical cases, each marked with specific suffixes [35]. This highly inflected case system, along with additional markings for number and possession, places a significant cognitive burden on writers and increases the likelihood of inflectional mistakes, particularly in written text.

Orthographic and Structural Errors: These are the most prevalent in this dataset (58 instances), especially spell (`SPELL`: 38) and orthography (`ORTH`: 17). These errors are caused by the agglutinative nature of Finnish, where words are built by combining

smaller parts, making them longer and more complex [35]. This morphological complexity increases the probability of typographical mistakes, especially in case endings and compound formation.

Verbal Errors: Despite the inherent complexity of Finnish verb conjugation, the dataset shows a moderate number of Verbal Errors (16 instances), most of which relate to tense and verb form. These errors are maybe due to the difficulty of using less common or irregular verb forms. However, the regular patterns of Finnish verbs help reduce the chances of more errors in this area.

Methodological limitation: The author of this thesis has only basic proficiency in Finnish. This posed difficulties in accurately distinguishing between verb form, verb tense, and verb inflection. In cases where it was unclear, the sentences in question were excluded from the dataset. This compromise introduces a degree of bias into the reported number of Verbal Errors and should be taken into account when interpreting these results.

In conclusion, the error patterns observed in the UD Finnish TDT dataset can be attributed to the complex morphological structure of Finnish, with its rich system of noun inflection and agglutinative word formation. Moreover, the methodological limitations due to the author's basic proficiency in Finnish also contributed to some bias in the classification of Verbal Errors, which should be considered when interpreting the results.

5.9 Distribution of Error Correction Action Types in UD Treebanks

The ERRANT framework categorizes the GEC of an item into three main action types: `R` (Replacement), `M` (Missing), and `U` (Unnecessary). Analyzing the distribution of these actions across those three datasets provides insight into the nature of the required corrections.

In the UD English EWT and UD Finnish TDT datasets, all identified errors are classified as `R` (Replacement), with 240 and 100 instances respectively. This means every correction involves substituting one token with another, without any insertions or dele-

tions. Such a pattern suggests a narrower range of error types, where all corrections are only replacements.

By contrast, UD English GUM dataset displays a more balanced mix of action types: `R`: 159, `M`: 22, and `U`: 21. This distribution indicates a richer variety of correction types, including not only replacements but also additions and removals of tokens. The presence of all three action types in substantial numbers highlights the linguistic diversity and complexity captured in this dataset.

From a GEC perspective, the English GUM dataset is particularly valuable because it reflects a wider spectrum of real-world language issues. Its inclusion of missing and unnecessary token edits, alongside replacements, makes it a more comprehensive resource for GEC.

To complement this analysis, the following figure visualizes the distribution of ERRANT action types across the three UD treebank datasets. As shown in Figure 5.3, the contrast between the datasets is clearly illustrated, with the English GUM dataset displaying a more diverse correction action types.

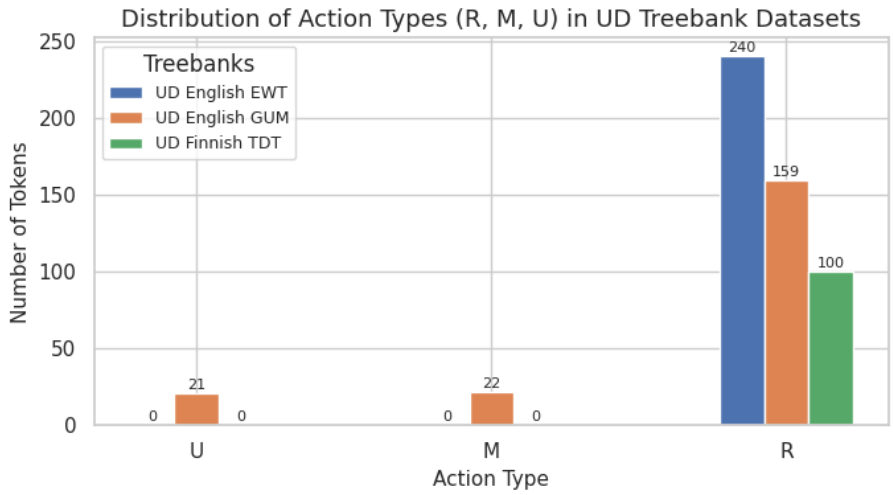

Figure 5.3: Distribution of action types (R, M, U) in UD treebank datasets

5.10 Overall Insights: Cross-Dataset and Cross-Linguistic Variability

The analysis of errors across UD English EWT, UD English GUM, and UD Finnish TDT revealed how dataset-specific characteristics and language-specific features influence grammatical error distribution patterns.

Cross-Dataset Patterns: Even though UD English EWT and UD English GUM both represent English language, their error patterns differ significantly, likely due to variations in domain, annotation practices, or register (i.e., the social context, tone, or formality of the language). UD English EWT, for instance, exhibited high frequencies of Orthographic and Structural Errors, especially related to spell and contraction. In contrast, UD English GUM demonstrated a more balanced error distribution, including a higher proportion of Verbal Errors and Function Word Errors, along with a richer variety of ERRANT action types including Replacement, Missing, and Unnecessary. These qualities make GUM particularly valuable for modeling real-world GEC scenarios.

Cross-Linguistic Patterns: UD Finnish TDT, meanwhile, displayed a grammar profile shaped by its complex morphology and challenges posed by Finnish's agglutinative nature.

These patterns suggest that using UD annotations can help identify language-specific trends in grammatical errors, which may be useful for developing GEC systems tailored to different languages.

6 Universal Dependencies (UD) as a Resource for Grammatical Error Correction (GEC)

6.1 Motivation and Overview of UD-Based GEC Approach

The first phase of this thesis focused on Grammatical Error Correction (GEC), with particular attention to exploring whether Universal Dependencies (UD) can be effectively utilized within GEC tasks.

The work began with an exploration of various standard GEC datasets, including the C4_200M Synthetic Dataset [13], CoNLL 2014 Shared Task Dataset [7], BEA-2019 Shared Task [8], JFLEG Corpus [9], GitHub Typo Corpus [16], cLang-8 Corpus [19], MultiGED Dataset [21], and ErAConD [23]. This process led to the selection of the ERRANT framework [4] as a powerful and widely adopted tool for grammatical error classification, which defines 25 grammatical error types and is commonly used for annotation and evaluation in GEC systems.

Next, the focus shifted to the UD framework [36], specifically to the `Typo=Yes` feature. A detailed understanding of how UD operates, its format, and the role of the `Typo=Yes` annotation was developed. The frequency of `Typo=Yes` tokens was analyzed across all UD treebanks, resulting in the selection of three of them for in-depth analysis: UD English EWT, UD English GUM, and UD Finnish TDT.

From these treebanks, sentences containing `Typo=Yes` tokens were randomly sampled. A total of 542 such tokens were manually annotated by determining both their correct forms and the corresponding ERRANT error types. This process resulted in the creation of three new datasets, each for one UD treebank, which includes information such as sentence ID, token ID, the original (incorrect) token, its lemma, universal part-of-speech tag, the correct form, and the ERRANT error classification.

Analyzing three annotated datasets provided valuable insights regarding the frequency and distribution of grammatical errors in the selected treebanks. Comparisons were made across UD English EWT, UD English GUM, and UD Finnish TDT to assess both dataset-specific characteristics and language-specific grammatical features. Notably, differences in grammatical structure between English and Finnish became evident through this analysis.

To conclude this phase of the thesis, the rest of this chapter directly addresses the following **Research Question:** *Can UD, particularly the `Typo=Yes` feature, be effectively utilized in GEC? Furthermore, what does UD offer that commonly-used GEC datasets lack, and how might future research integrate UD into practical GEC pipelines?*

The short answer is yes: UD can be highly beneficial for GEC, especially when combined with structured error annotations like those provided by the ERRANT framework. Although UD treebanks were not originally developed for GEC, their detailed grammatical structure offers an untapped resource for GEC applications. In the following sections of this chapter, these points will be discussed in more detail.

6.2 Syntactic Benefits of UD for GEC

A core strength of UD is its rich syntactic representation [2], which is particularly useful for identifying and correcting grammatical errors involving syntax, such as subject-verb agreement, article usage, and prepositions. UD's dependency relations and universal part-of-speech tags provide a strong structural overview of sentence composition [2], which can significantly support error detection and correction. The manual mapping of Typo=Yes tokens to ERRANT error types in this study was an attempt to link surface-level errors

to underlying syntactic structures. This approach helps interpret grammatical mistakes more precisely through the framework of UD.

Beyond syntactic utility, prior analysis of English and Finnish UD treebanks revealed meaningful cross-dataset and cross-linguistic error patterns. These findings, discussed in the previous chapter, underscore UD's potential to surface language-specific error trends that can inform more adaptable GEC systems.

6.3 Potential and Limitations of the "Typo=Yes" Feature

The `Typo=Yes` feature in UD offers a unique, though rarely used, opportunity for GEC research. It is sparsely and inconsistently annotated across treebanks. In fact, many UD treebanks either do not include any `Typo=Yes` annotations at all or contain only a very small number. While it has not previously been the focus of systematic GEC efforts, the present work demonstrates that it can be manually refined and aligned with detailed error types using frameworks like ERRANT. This alignment effectively transforms `Typo=Yes` from a general-purpose flag into a meaningful annotation layer for GEC, enabling the construction of lightweight yet structurally enriched datasets.

Although ERRANT can automatically classify grammatical edits between original and corrected sentences [4], the manual annotation of 542 `Typo=Yes` tokens in this thesis provided several advantages. Manual annotation ensures high accuracy and consistency, which is especially important when dealing with non-standard sources like UD, where error annotations were not initially intended for GEC. Furthermore, ERRANT's rules are predominantly tailored to English, reducing their applicability to morphologically rich languages like Finnish. In such cases, human interpretation is necessary to accurately classify ambiguous or structurally complex errors.

Another advantage of the `Typo=Yes` feature is that it marks real, naturally occurring errors, unlike synthetic datasets such as C4_200M, which rely on artificially generated mistakes. This makes `Typo=Yes` a valuable resource.

In conclusion, this thesis treated the `Typo=Yes` annotation not just as a way to flag errors, but as a starting point for deeper analysis. By linking it with syntactic structures and detailed error types, it becomes possible to create more meaningful GEC resources.

6.4 Leveraging UD for Multilingual GEC

One of the key strengths of UD is its ability to support cross-linguistic applications. Most GEC resources, including ERRANT, are primarily built for English, which makes it challenging to expand GEC research to other languages. For languages with complex grammar, like Finnish, developing GEC systems similar to ERRANT requires a deep understanding of their unique structures and common error patterns. Finnish, for example, has an agglutinative structure and complex morphology, which can make error correction more difficult.

What sets UD apart is that it is inherently multilingual, covering a wide variety of languages. In this study, the inclusion of the UD Finnish TDT treebank alongside English treebanks highlights how UD can play a crucial role in multilingual GEC. UD's part-of-speech tags and dependency relations provide language-agnostic structural insights, which can then be combined with language-specific error classifications. This combination makes it easier to build GEC systems for languages that are under-resourced or have different linguistic structures, like Finnish, helping create more inclusive and effective GEC models across multiple languages.

6.5 Challenges and Future Directions for Using UD in GEC

While UD offers many benefits, there are still challenges in using it directly for GEC. One major issue is the inconsistency in the annotation of the `Typo=Yes` feature across treebanks. Additionally, most UD treebanks do not include fully corrected sentence pairs, making it harder to apply UD data directly to GEC tasks.

Despite these challenges, the manual annotation of 542 `Typo=Yes` tokens in this study along with their correct forms and error classifications demonstrates that it is possible to adapt UD resources for GEC. This opens up several opportunities for future research. For example, it could be useful to develop a system that automatically aligns `Typo=Yes` tokens with their corrected forms and assigns ERRANT error types. Moreover, integrating UD's syntactic information into GEC pipelines could improve error classification and correction, as syntactic features such as dependency relations and part-of-speech tags might provide valuable context for detecting and correcting grammatical errors.

In summary, although there are challenges in using UD for GEC, the potential to refine and apply UD treebanks for GEC presents valuable directions for future work.

6.6 Conclusion

In conclusion, although UD treebanks were not originally designed for GEC, they provide significant value due to their syntactic richness, including universal part-of-speech tags and dependency relations, as well as their consistent structure across languages. This thesis has demonstrated that `Typo=Yes` tokens, when systematically aligned with detailed language-specific error types like those in the ERRANT framework, can effectively adapt UD treebanks for GEC resources.

However, several challenges remain. The `Typo=Yes` feature is inconsistently annotated, many UD treebanks lack fully corrected sentence pairs, and the effective application of UD for non-English languages requires the development of language-specific error classification frameworks. Despite these limitations, UD treebanks present a promising foundation for the development of multilingual GEC systems and offer valuable directions for future research.

7 Foundations of Large Language Models

7.1 Introduction to Large Language Models (LLMs)

Large Language Models (LLMs) are advanced deep learning models trained on vast corpora of text to understand, generate, and manipulate human language. They are typically based on the Transformer architecture, which introduced the concept of self-attention [37], a mechanism that allows the model to weigh the importance of different words in a sequence relative to each other. Earlier architectures relied on recurrence, where sequences were processed step by step, as seen in Recurrent Neural Networks (RNNs). Some also used convolution, which applies fixed filters to capture local patterns, as in Convolutional Neural Networks (CNNs). In contrast, the Transformer uses self-attention, allowing it to process the entire sequence simultaneously.

LLMs are capable of performing a wide range of NLP tasks, including text completion, machine translation, summarization, and question answering. Their effectiveness is largely attributed to their scale, in terms of model parameters and training data, and their ability to generalize from patterns in natural language. [38]

7.2 Core Architectures and Representative Systems in LLMs

While LLMs, in general, represent a powerful class of models capable of diverse natural language tasks, different architectures have been developed to optimize performance across various use cases. Among the most influential of these models are GPT, BERT, and T5. Each of these models builds upon the Transformer architecture, but with distinct approaches and innovations tailored to different types of language understanding and generation tasks. In addition to these foundational models, this section also discusses ChatGPT, an application built on GPT. The following subsections examine the structure and characteristics of these models and systems, along with their contributions to the development of LLMs. [38]

7.2.1 GPT (Generative Pretrained Transformer)

GPT is a model developed to improve the performance of language models by pre-training on large unlabeled datasets and fine-tuning for specific tasks. While there are large amounts of unlabeled text available for training, labeled datasets specific to these tasks are limited, making it difficult for traditionally discriminative models to perform well. To overcome this challenge, GPT was designed to use a two-stage approach: it first undergoes generative pre-training on a large and diverse corpus of unlabeled text, and is then fine-tuned discriminatively on specific tasks. GPT uses task-aware input transformations during fine-tuning, which enables it to effectively transfer its learned knowledge while requiring minimal changes to the model architecture. [39]

This strategy has proven highly effective across a broad range of natural language benchmarks, as demonstrated in the original evaluations. GPT's task-agnostic design not only simplifies adaptation but also outperforms models that were specifically crafted for individual tasks.

The GPT model is primarily designed for text generation tasks, but it can also be applied to a variety of natural language understanding tasks such as question answer-

ing, textual entailment, and document classification. However, unlike models like BERT, which process text in both directions (bidirectionally), GPT processes text in a unidirectional manner, meaning it reads from left to right. This unidirectional processing limits its ability to fully capture the context of a word from both sides simultaneously, which can impact performance on certain tasks that require deeper contextual understanding.

7.2.2 ChatGPT

ChatGPT is not a model itself in the same way that GPT or BERT are. Instead, it is a chat-based application built on top of a GPT model, designed specifically for conversational interactions. It incorporates fine-tuning through human feedback, a process known as Reinforcement Learning from Human Feedback (RLHF). In this approach, the model improves its responses based on human evaluations rather than relying only on labeled datasets. [38]

ChatGPT is a strong example of how pre-trained foundation models (PFMs) are being used to create more interactive AI systems. It demonstrates notable progress in zero-shot and few-shot learning. Zero-shot learning refers to the model's ability to handle tasks without having seen any examples during training, while few-shot learning involves solving tasks after being shown only a small number of examples.

ChatGPT has been specifically fine-tuned to follow user instructions more effectively, produce coherent and contextually relevant responses, and engage in multi-turn conversations, where it maintains a dialogue across several exchanges instead of responding in isolation. Its development highlights the increasing importance of aligning LLMs with human intentions, which is becoming a central goal in the evolution of PFMs.

7.2.3 BERT (Bidirectional Encoder Representations from Transformers)

BERT is a powerful model for language representation designed to learn from both the left and right context of words at the same time. This bidirectional approach allows

BERT to better understand the full meaning of a sentence, as it can look at the words before and after a given word simultaneously, instead of just one direction. [40]

After being trained on large amounts of unlabeled text, BERT can be fine-tuned for specific tasks, such as question answering or language inference, by simply adding one extra output layer to create state-of-the-art models. This means that it can be used across a wide variety of tasks without needing to make big changes to its structure. BERT combines conceptual simplicity with strong empirical performance. It has achieved state-of-the-art results across eleven NLP tasks demonstrating its practicality.

7.2.4 T5 (Text-to-Text Transfer Transformer)

T5 introduces a unified framework that converts all text-based language tasks into a text-to-text format. This means every task, whether it is question answering, summarization, or text classification, is treated as a problem of transforming input text to output text. This approach is different from models like BERT and GPT, which are not inherently text-to-text models. The text-to-text approach simplifies the transfer learning process and makes it easy to apply the same model across different types of tasks. Transfer learning, the process of pre-training a model on a data-rich task and then fine-tuning it for a specific downstream task, has become a powerful approach in NLP. [41]

T5 uses an encoder-decoder architecture where the encoder is bidirectional and the decoder is unidirectional. This allows it to effectively perform text-to-text transformations. T5 was pre-trained on a massive, cleaned dataset called the Colossal Clean Crawled Corpus (C4), which was built by filtering and cleaning the Common Crawl web data to remove noise and low-quality content. After pre-training, T5 is fine-tuned on specific downstream tasks, achieving state-of-the-art results across a wide range of NLP tasks. Despite the benefits of scaling up models, the creators also emphasize the importance of research into smaller, more efficient models for low-resource scenarios.

7.3 Training Paradigms and Challenges of LLMs

The development of LLMs depends on large-scale datasets and substantial computational resources, guided by several key training paradigms. One of the most fundamental paradigms in LLM training is self-supervised learning, which allows models to learn representations from unlabeled data. This paradigm forms the foundation of the pre-training phase, where models are exposed to vast and diverse corpora to acquire broad, generalizable linguistic knowledge. [38]

Another key training paradigm is transfer learning, which allows the model to rapidly adapt to specific downstream tasks with limited labeled data. This process involves fine-tuning learned representations and further specialization through techniques like prompt tuning and instruction tuning, enabling the model to efficiently adjust to particular domains or task requirements. [38]

Despite their impressive advancements, LLMs continue to face several challenges. The resource-intensive nature of LLMs raises concerns about their accessibility in low-resource settings and their environmental sustainability. Training and deploying these models require substantial computational power and energy, often resulting in significant carbon emissions. This prompts critical reflection on how to develop and use LLMs in a more energy-efficient and environmentally responsible way. Moreover, LLMs can unintentionally learn and reproduce sensitive information from training data, raising concerns about data privacy and model misuse. Furthermore, bias and fairness remain significant issues, as LLMs can inherit and even amplify societal biases embedded in the training data. Lastly, improving the interpretability of LLMs is crucial, as understanding their decision-making processes remains a complex and ongoing challenge. [42] [38]

8 Overview of Recent Methods in GEC Using LLMs

8.1 The Effectiveness of Transformer Language Models in GEC

This section is based on the work of Alikaniotis and Raheja [43], who explored the use of Transformer language models in GEC. Traditionally, GEC has relied on rule-based systems or statistical models that required extensive annotated datasets. In contrast, recent developments in Transformer-based architectures, such as BERT and GPT, have introduced promising alternatives. These models, pre-trained on large-scale natural language corpora, can be used for GEC without the need for explicitly labeled correction data.

The method explored in this approach uses a pre-trained Transformer language model to generate multiple candidate corrections for a given sentence and then score these candidates based on their probability, selecting the most likely sentence as the final correction. This strategy relies on language modeling alone, without any supervised training.

Although these models are not fine-tuned on GEC-specific data, they can perform surprisingly well, in some cases approaching the performance of supervised systems. The simplicity and flexibility of the method are key strengths, as it reduces dependence on annotated training data and facilitates adaptation to different languages or domains. However, the approach still faces limitations, particularly with intensive sentence-level changes or errors requiring complex contextual understanding. In conclusion, Trans-

former language models offer a strong unsupervised baseline for GEC, highlighting new possibilities for developing GEC in low-resource settings.

8.2 Pillars of GEC

This section is based on the work of Omelianchuk et al. [44], who conducted a comprehensive evaluation of contemporary approaches to GEC in the era of LLMs. The authors explore different methods for applying LLMs to GEC. Instead of focusing on a single method, they systematically compare prompting, fine-tuning, ensembling, and ranking to determine which approaches get the best results.

Zero-shot prompting involves asking the model to correct sentences without any further training. The model, such as GPT-4, is given a prompt like "Correct this sentence" and responds directly. This method is fast and easy to deploy, but its success heavily depends on the quality of the prompt, motivating additional work on prompt engineering.

Fine-tuning focuses on training pre-trained language models, such as T5 or BERT-based variants, on specific GEC datasets to improve their task-specific performance. Although this process requires more computational resources, the authors demonstrate that fine-tuned models can achieve competitive results, particularly when large, high-quality datasets are available.

Ensembling uses multiple models together rather than relying on a single one. Different models may correct different types of errors better, so combining their outputs, for example through majority voting, can improve overall performance. Majority voting involves selecting the correction that is suggested by the majority of the models.

Ranking focuses on selecting the best correction from multiple generated candidates, using a separate model (sometimes GPT-4 itself) to make the selection. This strategy helps refine the results and reduce erroneous corrections.

The authors evaluate their systems on standard GEC benchmarks (CoNLL-2014 and BEA-2019), achieving new state-of-the-art results, with $F_{0.5}$ scores of 72.8 on CoNLL-2014 and 81.4 on BEA-2019, improvements of +1.7 and +0.6 points over previous bests,

respectively. The key findings of the study reveal that zero-shot prompting with GPT-4 delivers surprisingly strong results. Also, fine-tuning smaller models can achieve competitive performance while being more cost-effective to run. Combining multiple models through ensembling further improves results compared to relying on a single model. Moreover, ranking candidate corrections enhances the overall quality of the output. Notably, GPT-4 can serve dual roles, both as a corrector and as a judge for ranking the best corrections, highlighting its versatility in the GEC process.

8.3 LM-Critic: Language Models for Unsupervised GEC

This section is based on the work of Yasunaga et al. [45], who introduced an unsupervised approach to GEC, called LM-Critic, using language models as grammaticality critics, eliminating the need for labeled data. Traditional GEC models depend on large annotated datasets, which are costly and time-consuming to produce. LM-Critic overcomes this limitation by using pre-trained language models to evaluate the grammaticality of sentences.

This method builds on the *Break-It-Fix-It* (BIFI) framework which consists of three components: the Fixer, which attempts to correct errors in sentences; the Critic, a pre-trained language model that evaluates whether the Fixer's correction is more grammatical; and the Breaker, which generates new training examples by introducing errors into sentences.

When applied to datasets like CoNLL-2014, BEA-2019, GMEG-wiki and GMEG-yahoo, LM-Critic demonstrated significant improvements. In unsupervised settings, it achieved an average increase of $+7.7$ $F_{0.5}$ over existing methods, and in supervised settings, it provided an additional $+0.5$ $F_{0.5}$ improvement over strong baseline models. These results highlight the effectiveness of LM-Critic, especially in scenarios where labeled data is limited.

8.4 Grammar Error Explanation (GEE) with LLMs

This section is based on the work of Song et al. [46], who proposed a novel task of Grammar Error Explanation (GEE) using LLMs. Their approach goes beyond GEC by providing natural language explanations for each error, helping users better understand the rules behind corrections. GEE is designed to complement traditional GEC tools by not only correcting errors but also providing explanations essential for language learners to grasp underlying grammar rules. A two-step pipeline is introduced to achieve the purpose as the following.

Atomic Token Edit Extraction, as the first step, focuses on applying LLMs like GPT-4 to identify the minimal edits required to fix a sentence. These edits fall into four categories: Insert (adding a missing word), Delete (removing an unnecessary word), Replace (substituting an incorrect word with the correct one), and Relocate (moving a word to a different position). For example, in "She go to school.", the correction would involve replacing "go" with "goes."

Explanation Generation, as the second step, focuses on applying GPT-4 to provide a one-sentence explanation for each edit. For the earlier example, it would explain: "The verb 'go' is changed to 'goes' to agree with the third-person singular subject 'She'."

When tested on datasets from language learners in German, Chinese, and English, the pipeline showed strong performance. The atomic edit extraction achieved F1 scores of 0.93 for German, 0.91 for Chinese, and 0.891 for English. Human evaluations confirmed the explanations' accuracy with correctness rates of 93.9% for German, 96.4% for Chinese, and 92.2% for English. Initially, GPT-4 was evaluated using one-shot prompting: it detected only **60.2%** of true errors and correctly explained **67.5%** of the errors it detected. Overall, it explained just **40.6%** of all errors, leaving **39.8%** entirely unexplained. This highlights the challenge of generating accurate explanations without improved GEE models.

8.5 Efficient and Interpretable GEC with Mixture of Experts

This section is based on the work of Qorib et al. [47], who proposed MoECE, a mixture-of-experts model for GEC that specializes sub-networks in different error types. Unlike traditional methods relying on multiple models, MoECE combines specialized sub-networks (experts) within a single model. Each expert is trained to handle a specific grammatical error type, such as verb tense or article usage, while shared layers capture general language understanding.

A key feature of MoECE is its router mechanism, which directs input to the most relevant expert based on the error type, improving computational efficiency. Despite having fewer parameters than larger models, MoECE achieves T5-XL level performance with three times fewer parameters, faster inference times, and lower computational costs. Additionally, it provides interpretability by identifying the type of error corrected, which can assist language learners in understanding grammatical concepts. The model employs an error type loss that helps routing to experts and enhances interpretability by producing error-type-aware corrections. Overall, MoECE offers an efficient and interpretable approach to GEC by combining specialized experts within a single model.

8.6 LLMs as Annotators for Type-Aware Data Augmentation in GEC

This section is based on the work of Li and Lan [48], who proposed TypeDA, a method that uses LLMs as annotators for type-aware data augmentation in GEC to generate grammatical errors aligned with specific error types. Unlike previous data augmentation methods, which often lack diversity or introduce in-distribution noise, TypeDA focuses on generating controlled errors related to grammatical issues.

The method decomposes the augmentation process into two steps: masking parts of a correct sentence and filling them with type-specific incorrect tokens. This structured

prompting enables LLMs to generate diverse and realistic training examples while preserving the intended error type. Experimental results show that models trained with TypeDA-augmented data outperform baseline models, making the method especially valuable when annotated data is limited.

8.7 LLMs as Evaluators for GEC

This section is based on the work of Kobayashi et al. [49], who showed that LLMs, particularly GPT-4, are very effective at evaluating GEC systems. By prompting GPT-4 to assess aspects like fluency, grammatical accuracy, and meaning preservation, they found that its evaluations align much better with human judgments than traditional metrics such as ERRANT and GLEU.

Their comprehensive experiments showed that GPT-4 reached a Kendall's rank correlation of 0.662 with human evaluations, a measure of how closely the model's rankings match human rankings, outperforming all previously known methods. This means GPT-4's assessments closely reflect human preferences when ranking corrected sentences. The study also suggests that larger LLMs and well-crafted prompts play an important role in improving evaluation quality. For future work, the authors plan to explore how few-shot learning and better prompt design can further enhance performance. They also aim to investigate evaluating longer texts using GPT's expanded context window, which current metrics do not fully address.

9 Foundations of Prompting LLMs for GEC

9.1 Prompt Engineering Techniques for LLMs

This section builds upon the comprehensive survey by Sahoo et al. [50], which analyzes key prompt engineering techniques for LLMs. Prompt engineering has become an essential method for effectively utilizing LLMs in a wide range of NLP tasks. Rather than retraining models or modifying their parameters, prompt engineering focuses on designing input instructions, called prompts, that guide the model to perform specific tasks. This approach enables rapid adaptation of pre-trained models like GPT-3 and GPT-4 to new challenges without extensive fine-tuning, making it highly efficient and flexible.

One of the most basic techniques is zero-shot prompting, where a model is asked to perform a task without any prior examples. The prompt simply describes the task, relying on the model's pre-trained knowledge to generate a response. A more effective variation is few-shot prompting, which includes a small number of input-output examples within the prompt to demonstrate the task. This helps the model better understand the required format or logic, often leading to improved results.

To support tasks involving reasoning or multiple steps, Chain-of-Thought (CoT) prompting was introduced. This method encourages the model to show intermediate reasoning steps, similar to how a human would think before reaching a conclusion. Auto-CoT builds on this by automatically generating these reasoning steps, reducing the need for manual prompt construction. Another enhancement is self-consistency, which asks the

model to produce multiple reasoning paths and then selects the most common answer, improving reliability in complex problems.

For even more structured reasoning, Tree-of-Thoughts (ToT) and Graph-of-Thoughts (GoT) have been developed. ToT organizes reasoning as a tree, allowing the model to explore multiple branches of thought and backtrack if necessary. GoT further advances this idea by allowing non-linear, graph-based reasoning, mimicking how humans consider various interconnected ideas during problem-solving.

Another important direction in prompt engineering is reducing hallucinations, incorrect or fabricated outputs, by using techniques such as Retrieval-Augmented Generation (RAG). RAG improves the quality of LLM outputs by incorporating relevant information retrieved from external sources directly into the prompt, which helps ground the model's response in factual content. Similarly, ReAct combines reasoning with task-specific actions, such as deciding when to retrieve additional information, resulting in more accurate and transparent outputs.

In conclusion, prompt engineering has become a practical and versatile approach for adapting LLMs to a wide range of tasks. Techniques like zero-shot, few-shot, CoT, ToT, GoT, and RAG demonstrate how carefully designed prompts can guide models toward better reasoning and more accurate outputs. As models become more capable, prompt engineering will continue to play a central role in guiding their application across diverse domains.

9.2 Prompting Open-Source and Commercial Language Models for GEC of English Learner Text

Recent advances in LLMs have enabled impressive fluency and grammaticality in generated text, and they show potential for GEC when prompted with ungrammatical sentences. This section builds upon the work of Davis et al. [51], which investigates how both open-source and commercial LLMs perform on GEC tasks. The authors evaluate ten LLMs using four English benchmark datasets, and measure performance with the

ERRANT toolkit, which provides detailed error-type classification and computes precision, recall, and $F_{0.5}$ scores. The benchmark datasets used in the study include FCE, CoNLL-14, W&I+LOCNESS, and JFLEG. The evaluated models also include commercial systems such as GPT-3.5 and GPT-4, and open-source models such as LLaMA-2 Chat and Falcon-40b-Instruct.

The study shows that LLMs do not consistently outperform supervised GEC systems, except in specific cases such as commercial models performing well on fluency-focused benchmarks. Several open-source models were found to surpass commercial ones on tasks requiring minimal edits. Although efforts were made to encourage minimal edit corrections through prompt design, LLMs tend to favor fluency rewrites, which explains their stronger performance on fluency-annotated datasets compared to minimal-edit datasets where supervised models still lead.

The authors also compare zero-shot and few-shot prompting strategies, observing that well-designed zero-shot prompts often perform as well as or better than few-shot prompts. This emphasizes that the effectiveness of GEC from LLMs depends more on the quality of prompts than on the quantity or number of examples provided.

Furthermore, the results indicate that, in general, LLMs demonstrate strong performance on error types such as spelling, missing determiners, replacement subject–verb agreement, replacement noun number, and orthography. However, they perform less effectively on open-class replacement of nouns and verbs, as well as the broad "other" error type. This suggests that LLMs are more successful with morphological or character-level corrections that remain close to the original text, whereas lexical or phrasal replacements within the minimal edit paradigm pose greater difficulty.

9.3 Analyzing the Performance of GPT-3.5 and GPT-4 in GEC

This section builds upon the work of Coyne et al. [52], which provides a detailed evaluation of GPT-3.5 and GPT-4 models on GEC tasks. The authors test both models on

established benchmarks like BEA-2019 and JFLEG, exploring how different prompting strategies, zero-shot and few-shot, affect their ability to correct grammar and improve sentence fluency.

Their experiments reveal that prompt design plays a crucial role in model performance. Among zero-shot prompts, some doubled the performance of others. Both GPT-3.5 and GPT-4 performed consistently better with lower temperature settings. Temperature is a parameter that controls randomness in language generation, where lower values produce more precise and predictable outputs, and higher values generate more varied and creative responses. Few-shot prompting modestly improved GPT-3.5's performance, peaking with two examples. For GPT-4, however, adding few-shot examples had little or slightly negative effect. Interestingly, GPT-3.5 sometimes outperformed GPT-4 in zero-shot scenarios, though GPT-4 generally performed better when using the best prompts. Both models achieved their top results with the same carefully engineered prompt including two few-shot examples.

On benchmark tests, GPT-4 reached state-of-the-art scores on JFLEG, demonstrating strong fluency corrections. However, both models performed worse on the BEA-2019 dataset, which rewards minimal edits, reflecting their tendency to perform more extensive sentence revisions rather than minimal corrections. This emphasis on fluency can lead to over-editing or changes that alter the original meaning, explaining the high fluency scores and positive human evaluations alongside lower scores on datasets prioritizing minimal edits. The authors also highlight that GEC is inherently complex: human annotators often disagree on ideal corrections, and the best editing style depends on the text's context and purpose, such as language learning versus academic writing. This underscores the importance of clearly defining goals and evaluation criteria when applying models like GPT-3.5 and GPT-4 to grammatical error correction.

10 A Unique Prompt Engineering Experiment for ERRANT-based Error Classification

10.1 Research Background and Objectives

The first phase of this master's thesis involved work with three Universal Dependencies (UD) treebanks: UD English EWT, UD English GUM, and UD Finnish TDT. In each dataset, a set of tokens annotated with the feature `Typo=Yes` was randomly selected. These tokens were then manually annotated with grammatical error types based on the ERRANT (Error Annotation Toolkit) framework, which offers a standardized approach to classifying grammatical errors.

The current phase of the thesis focuses on evaluating the capability of an LLM to perform grammatical error classification on the same set of tokens, using structured prompts informed by the ERRANT framework. This prompt engineering experiment includes both zero-shot and few-shot configurations. The LLM's predictions are compared against the manual annotations made during the first phase of the thesis, which serve as a gold standard, to assess how effectively prompt design can guide the model in such a task.

10.2 Task Complexity and Classification Challenges

An important consideration in this study is the complexity of the classification task. Under the ERRANT framework, each ungrammatical token can potentially be annotated with one of 75 distinct annotations, formed by combining three edit operations, Replacement (R), Unnecessary (U), and Missing (M), with 25 main grammar error categories.

This leads to a much more detailed classification space compared to binary or standard multi-class tasks. With 75 possible labels, many of which capture subtle linguistic distinctions, the task presents a high risk of ambiguity and misinterpretation. This is particularly true when using LLMs via prompting, as they may struggle with overlapping label definitions, context-dependent interpretations, or aligning their reasoning with the precise constraints imposed by the ERRANT framework.

In conclusion, high accuracy should not be expected, and performance should be interpreted within the context of these structural limitations. The objective is not to achieve flawless classification, but rather to analyze how well prompt-engineered LLMs can approximate human-level annotation in a task that demands linguistic precision and categorical clarity.

10.3 Dataset Preparation

The dataset used in this study closely resembles the one constructed during the first phase of this thesis. During that initial phase, a manual annotation process was carried out on three UD treebanks: UD English EWT, UD English GUM, and UD Finnish TDT. Tokens annotated with the feature `Typo=Yes` in the original CoNLL-U files were identified as candidates for grammatical error classification. Following the manual annotation, relevant information was extracted from the CoNLL-U files and compiled into structured CSV files, which form the core datasets. Each row in the CSV file represents a single token with a grammatical error and contains the following features:

- `sentence_id`: The identifier of the sentence in the original UD treebank.

- `text`: The full sentence containing the erroneous token.
- `token_id`: The position of the ungrammatical token within the sentence.
- `token_form`: The erroneous token.
- `lemma`: The base or dictionary form of the token.
- `CorrectForm`: The intended, grammatically correct version of the token.
- `ERRANT`: The error type assigned to the token according to the ERRANT framework, annotated manually by the author.

In the current phase of the thesis, the same dataset is employed as input for a prompt-based grammatical error classification experiment using an LLM. For each data item, all features except the `ERRANT` are used to construct the input prompt. The LLM is expected to predict the most appropriate ERRANT-style error classification based on the provided context. A new CSV file is then generated that includes the model's predicted `ERRANT` error label. This file allows for a direct, item-by-item comparison between the model-generated labels and the gold standard manual annotations, forming the basis for accuracy evaluation.

Details about the prompt structure, model configuration, and evaluation procedures are presented in the subsequent section.

10.4 Zero-Shot Prompt Design and Model Configuration

In this phase of the thesis, a prompt-based approach was employed to assess the ability of an LLM to perform grammatical error classification according to the ERRANT framework. This experiment is framed as a prompt engineering task, where model responses are evaluated against a manually annotated gold standard. The LLM used in this study was OpenAI's `gpt-4o-mini`, accessed via the OpenAI API key provided by the TurkuNLP Lab at the University of Turku. The evaluation began with a zero-shot

prompt setup, where the model receives no examples but is guided solely by a detailed natural language instruction. The prompt was designed to simulate the perspective of an annotation expert working with ERRANT, a structured framework for grammatical error classification. ERRANT distinguishes between 25 main error types, which are further categorized according to three edit operations including `R`: Replacement, `U`: Unnecessary, and `M`: Missing, resulting in 75 distinct annotation possibilities for each error.

The early runs with less detailed instructions revealed that the model often misunderstood the required ERRANT format. Hence, the prompt was revised and expanded to ensure compliance with ERRANT's structured error type expectations to reduce invalid output formats. The model was explicitly instructed to return error labels in the format `EDIT_OPERATION:ERROR_TYPE` (e.g., `R:SPELL`, `U:DET`, `M:VERB:TENSE`). This strict format was emphasized in the prompt to prevent the model from producing vague or incomplete outputs.

Furthermore, the prompt emphasized the importance of contextual analysis, meaning that the model was instructed to avoid making surface-level comparisons between the incorrect token and its corrected form in isolation. Instead, it was required to consider the full sentence context to determine the nature of the grammatical error, as contextual meaning plays a crucial role in accurate error classification under the ERRANT framework.

The zero-shot prompt design followed the original ERRANT framework, as outlined in Section 3.1. Figure 10.1 provides a simplified overview of the key components and logical flow of the zero-shot prompt design used for ERRANT error classification. Also, the full zero-shot prompt is provided in Appendix.

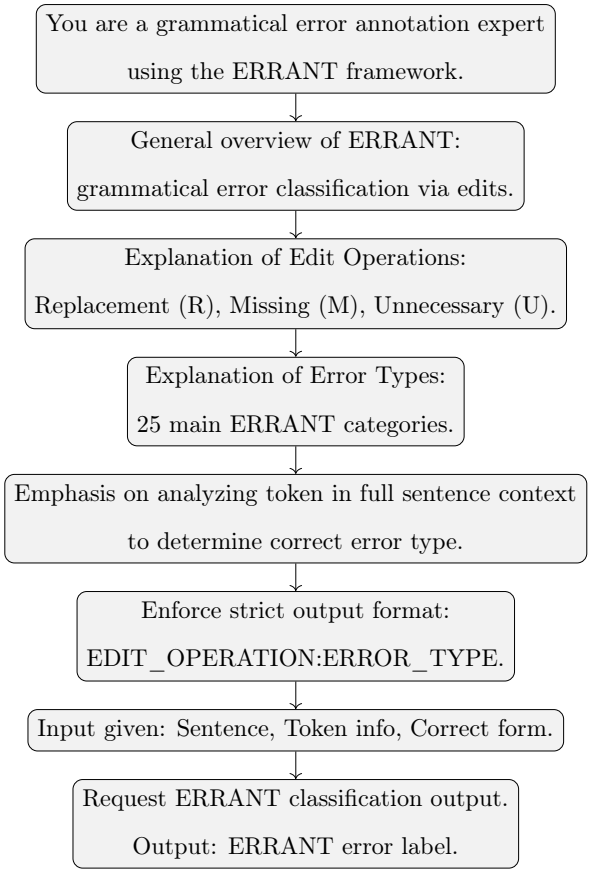

Figure 10.1: Components of the zero-shot prompt for ERRANT error classification

10.5 Zero-Shot Results

10.5.1 UD English EWT

To establish a baseline, the model's ability to classify ERRANT edits was evaluated in a zero-shot setting, using a 240-item subset of the UD English EWT. All edits in this gold-standard set were manually annotated as Replacement (R) operation, making it an ideal setup for testing whether a language model can infer both the operation type and the main error category without any training examples.

In the broader evaluation across all 75 ERRANT categories (derived from 3 operation types $\times$ 25 main error types), the model achieved an accuracy of **49.58%**. The classification report shows strong performance on some of the most frequent categories. For instance, `R:SPELL` achieved a precision of 96.23% and an F1-score of 72.34%, while

`R:CONTR` had nearly perfect precision (98.08%) and a high F1-score of 87.17%. These results indicate that the model can often correctly identify common and surface-level edits. On the other hand, classes like `R:ORTH` and `R:NOUN:POSS`, which are more syntactic or semantic in nature, were significantly harder for the model. `R:ORTH` reached only 3.28% F1-score, and `R:NOUN:POSS` showed zero precision and recall. Figure 10.2 presents a horizontal bar chart of the top 10 ERRANT classes ranked by support, with F1-score indicated through color coding to highlight performance differences across categories.

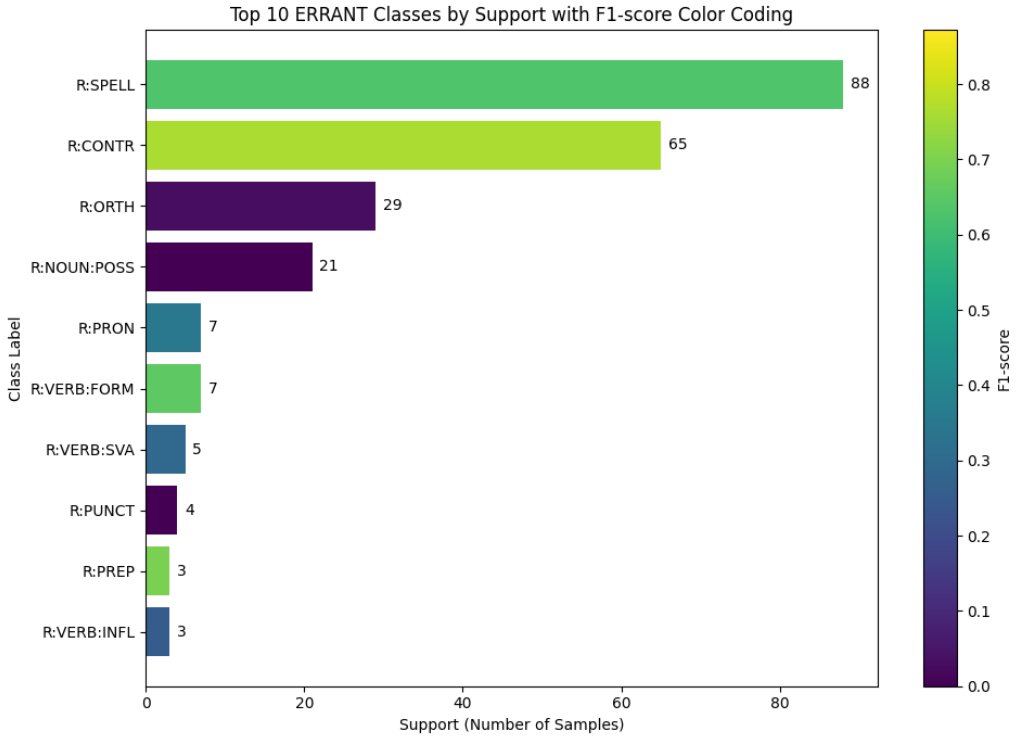

Figure 10.2: EWT zero-shot top 10 ERRANT classes by support with F1-score color coding

At the operation level, the model achieved an accuracy of **78.33%**, correctly predicting the `R` operation for 188 out of 240 edits. However, 52 edits were misclassified: 46 were incorrectly labeled as `Unnecessary (U)`, 3 as `Missing (M)`, and 3 were completely misaligned. This reflects the model's moderate ability to detect that an edit is needed, although it sometimes struggles with identifying the correct type of operation. Figure 10.3 shows a heatmap visualization of the confusion matrix at the edit operation level, illustrating the distribution of correct and incorrect predictions across different operation types.

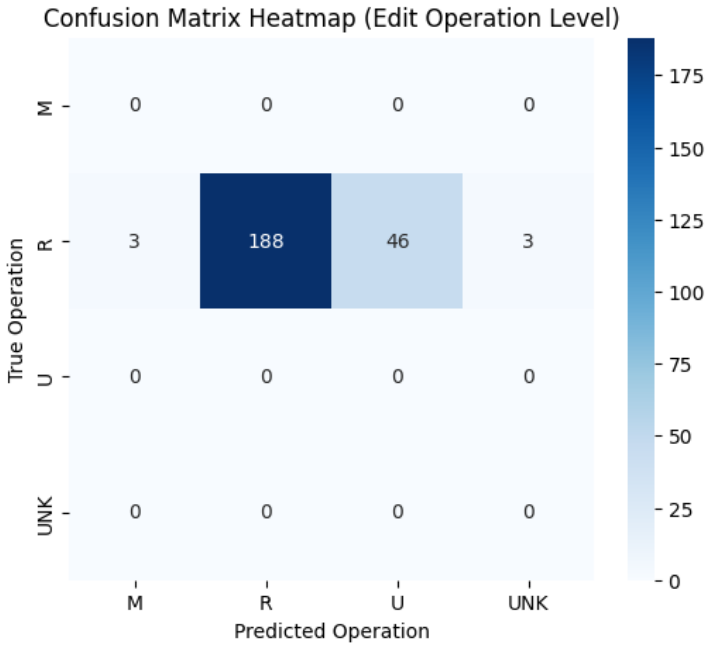

Figure 10.3: EWT zero-shot confusion matrix heatmap (edit operation level)

Finally, when focusing only on predicting the main error types, ignoring the specific operation, the model achieved an accuracy of **57.92%**. This shows a partial advantage in identifying the category of the error even when the operation type is incorrect. These results collectively highlight that while the language model demonstrates solid zero-shot performance on common surface-level errors, it struggles with deeper grammatical categories.

10.5.2 Misclassification Patterns on UD English EWT

A closer look at the 46 `R` → `U` misclassifications reveals two especially common patterns:

14 cases involved edits that should be labeled `R:ORTH` but were misclassified as `U:ORTH`. Most of these errors were due to whitespace in email addresses. For example, a sentence like “arvin jalali@tut.fi” should be corrected to “arvinjalali@tut.fi”. In ERRANT’s logic, this is a replacement, where the token “arvin” is replaced by the full email address. However, the model often interprets the whitespace as an unnecessary token and classifies it as `U:ORTH`. This happens because the correction removes the space, and without understanding the full token replacement, the model sees it as a deletion. This reflects a conventional annotation standard in UD that a language model cannot infer without

learning from examples. It is a known limitation of zero-shot prompting rather than a shortcoming of the UD standard itself.

13 cases involved possessive noun corrections (`R:NOUN:POSS`) misclassified as `U:ORTH`. A typical example is "my horses leg", which should be "my horse's leg". Here, the model mistakenly treats the missing apostrophe-s as a surface-level orthographic mistake, rather than a syntactic possessive structure. This indicates that the model has difficulty distinguishing grammatical corrections from orthographic ones, particularly in possessive constructions.

Overall, these results illustrate a key limitation: even when the model correctly identifies that a token needs to be changed, it often misidentifies the nature of the correction. These findings support the need for few-shot prompting, where just a few examples can guide the model more clearly on both the operation type and main ERRANT error classes.

10.5.3 UD English GUM

To complement the evaluation on EWT, we tested the model's zero-shot classification capabilities on a 202-item subset of UD English GUM. This set includes 159 Replacement (R), 22 Missing (M), and 21 Unnecessary (U) edits, providing a more diverse distribution of operations compared to the EWT subset.

Across the 75 possible ERRANT classes (25 error types $\times$ 3 operations), the model achieved a classification accuracy of **47.52%**. Several categories saw strong performance, particularly those involving surface-level edits. For instance, `R:SPELL` achieved a high precision of 94.44% and an F1-score of 60.71%, while `R:VERB:SVA` stood out with perfect precision and an F1-score of 87.50%. Similarly, `R:NOUN:NUM` and `R:PRON` yielded F1-scores above 68%, indicating solid performance on common agreement and pronoun errors. In contrast, classes like `R:CONTR` (F1 = 14.29%) and `R:PUNCT` (F1 = 0.00%) proved more difficult, suggesting limited generalization for less frequent or more structurally embedded edits. Figure 10.4 presents a horizontal bar chart of the top 10 ERRANT classes ranked by support, with F1-score indicated through color coding to highlight performance differences across categories.

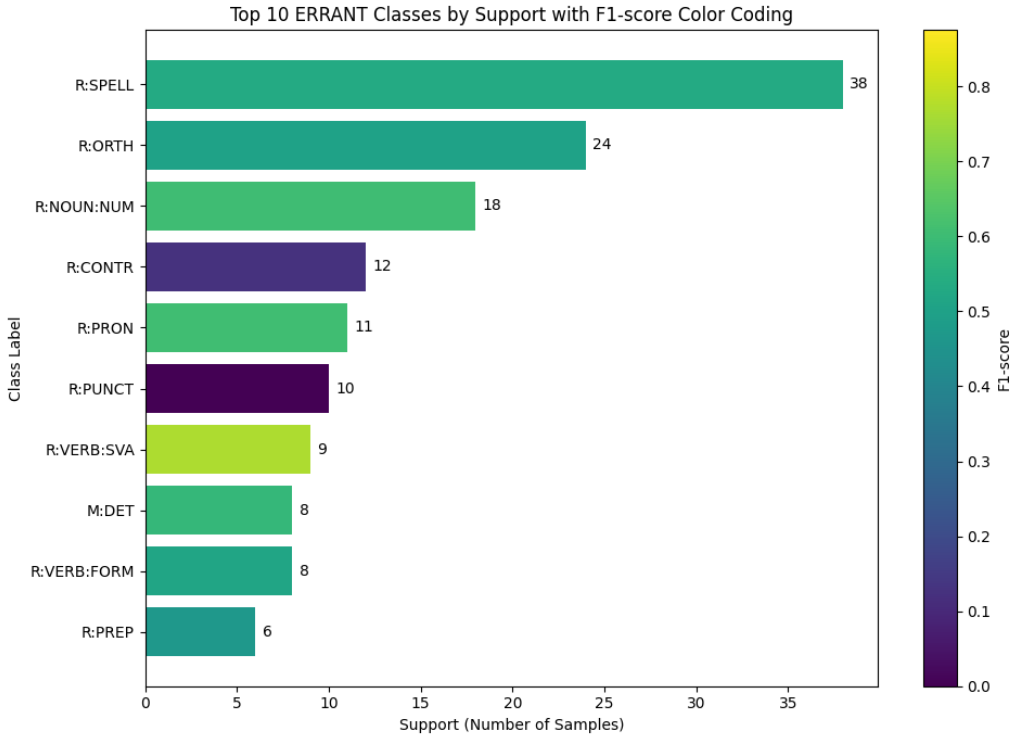

Figure 10.4: GUM zero-shot top 10 ERRANT classes by support with F1-score color coding

At the operation level, the model predicted the correct operation (`R`, `M`, or `U`) in **82.18%** of cases. Out of the 202 edits evaluated, 159 were originally labeled as `R`, of which 130 were correctly predicted. The remaining 29 `R` edits were misclassified: 17 as `Unnecessary (U)` and 12 as `Missing (M)`. Among the 22 edits labeled as `Missing (M)`, 17 were correctly classified, while 5 were misclassified as `Replacement (R)`. Of the 21 edits labeled as `Unnecessary (U)`, 19 were correctly identified, while 1 was misclassified as `Missing (M)` and 1 received an unknown label. Figure 10.5 shows a heatmap visualization of the confusion matrix at the edit operation level, illustrating the distribution of correct and incorrect predictions across different operation types. Accuracy (edit operation only): 82.18

When evaluating only the prediction of the main error type (ignoring operation labels), the model achieved an accuracy of **54.95%**, nearly matching its full-label performance. These findings reflect the overall trend observed in EWT: The model performs well on common, surface-level grammatical edits but still struggles with deeper syntactic or lexical issues. The consistent accuracy across two different English UD corpora suggests a stable

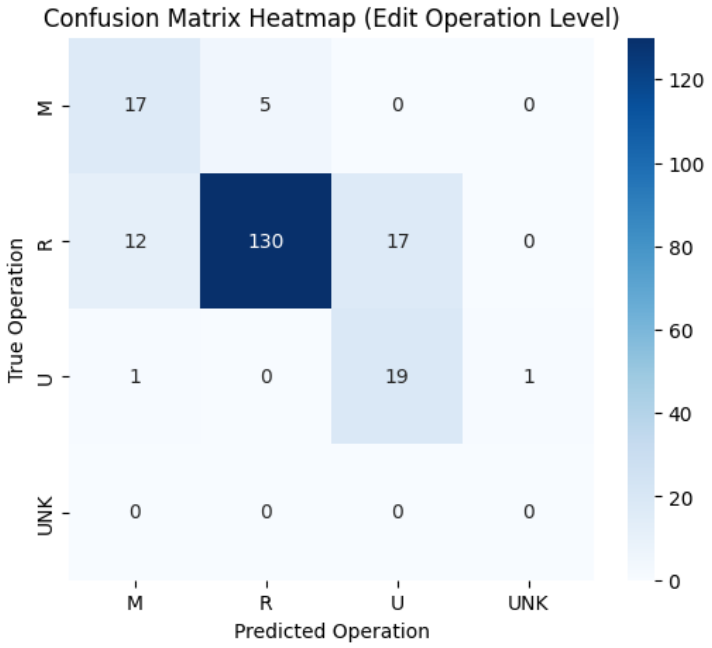

Figure 10.5: GUM zero-shot confusion matrix heatmap (edit operation level)

zero-shot baseline, though it also highlights areas where carefully developed prompts could yield considerable improvements.

10.5.4 Misclassification Patterns on UD English GUM

Notably, there were 7 cases where the true label was `R:NOUN:NUM` but the model predicted `M:NOUN:NUM`, and 5 cases where the true label was `R:PUNCT` but the model predicted `U:ORTH`, highlighting specific confusions between these error types.

For instance, in cases such as *"two thing"* (instead of *"two things"*), the correct ERRANT label is `R:NOUN:NUM` because the plural noun form *things* replaces the incorrect singular *thing*. However, the model labeled it as `M:NOUN:NUM`, suggesting that it interpreted the plural marker *s* as missing rather than incorrect. This reflects a conceptual mismatch between the ERRANT framework and the model's reasoning: ERRANT treats it as a replacement, whereas the language model treats the plural as a missing, implying that it thinks the plural suffix was simply omitted.

Similarly, in punctuation-related errors such as *"etc"* (instead of *"etc."*), the correct label is `R:PUNCT`, indicating a replacement to include the missing period. However, the model predicted `U:ORTH`, implying that it considered the period as an unnecessary or-

thographic error. A similar issue arises in cases like *"aka"* instead of *"a.k.a."*, where ERRANT treats the punctuation as a replacement, while the model again marks it as an unnecessary orthographic error. These examples demonstrate that the model struggles to align with ERRANT's treatment of punctuation corrections. These examples highlight the need for better alignment between model reasoning and annotation frameworks like ERRANT.

10.5.5 UD Finnish TDT

To further assess cross-lingual error classification, we evaluated the model's zero-shot classification capabilities on a 100-item subset of UD Finnish TDT. This dataset consists of 100 Replacement (R), with no instances of Missing (M) or Unnecessary (U) edits. Regarding the full label match, the model achieved a classification accuracy of **33.00%**. Among the top 10 classes by support, the model performed well on surface-level categories such as `R:SPELL`, which attained a high precision of 90.91% and an F1-score of 40.82%, and `R:ORTH`, which reached an F1-score of 61.90%. `R:PRON` and `R:VERB:FORM` also showed strong performance despite their lower support, with F1-scores of 57.14% and 50.00%, respectively. In contrast, categories like `R:NOUN:INFL`, `R:VERB:INFL`, and `R:PUNCT` yielded F1-scores of 0.00%, highlighting challenges in handling morphosyntactic and punctuation-related edits. Figure 10.6 displays a horizontal bar chart of the top 10 classes by support, with F1-score color coding to visualize relative performance across categories.

At the operation level, the model correctly identified the edit operation (`R`) in **88.00%** of the cases. Of the 100 edits labeled as `Replacement (R)`, 88 were correctly predicted, while 12 were misclassified as `Unnecessary (U)`. The confusion matrix in Figure 10.7 illustrates these results.

When evaluating the prediction of error types alone (ignoring the operation), the model achieved an accuracy of **36.00%**.

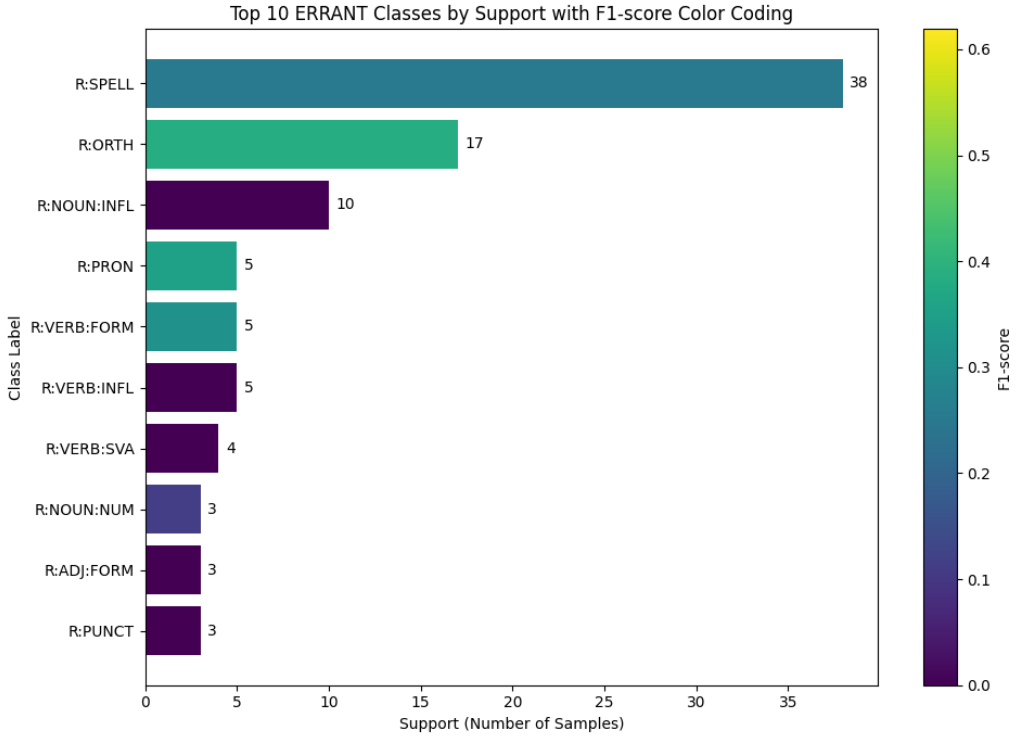

Figure 10.6: TDT zero-shot top 10 ERRANT classes by support with F1-score color coding

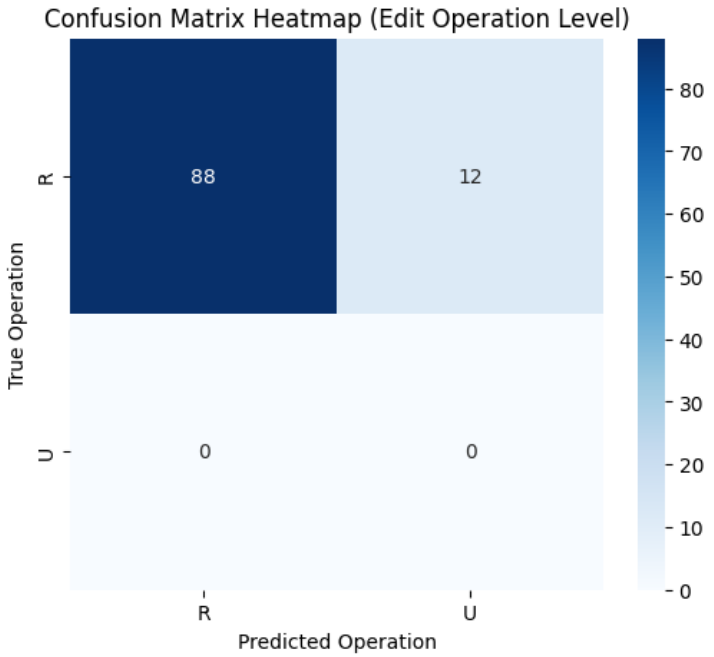

Figure 10.7: TDT zero-shot confusion matrix heatmap (edit operation level)

10.6 Few-Shot Prompt Design

To improve the model's ability, with a particular focus on distinguishing between edit operations, we created a few-shot prompt consisting of 10 hand-crafted examples. These were designed after analyzing zero-shot results on UD English EWT and UD English GUM, where we observed common misclassifications between `Replacement`, `Missing`, and `Unnecessary` operations. Since the author has only basic proficiency in Finnish, it was not feasible to design similar examples informed by insights from the Finnish TDT dataset.

The selected examples aim to guide the model toward better performance specifically on English UD corpora, particularly EWT and GUM. While they improve handling of surface-level errors—such as spacing, omitted function words, and simple agreement mismatches—they do not guarantee improved performance on other UD datasets with different domains or languages.

Importantly, these examples are constructed to help the model internalize how the ERRANT framework categorizes edits—not just in terms of specific error types, but also in distinguishing among the three fundamental operation classes: `R`, `U`, and `M`. By including a diverse mix of edits (e.g., `R:ORTH`, `R:NOUN:POSS`, `U:DET`, `M:VERB:FORM`), the few-shot prompt encourages a general understanding of how ERRANT defines and labels grammatical corrections. The complete few-shot design is provided in the Appendix.

10.7 Few-Shot Results

To evaluate the impact of few-shot prompting, the model's performance was measured on UD English EWT, UD English GUM, and UD Finnish TDT, utilizing the 10 manually designed examples (see Appendix). Table 10.1 summarizes results for three metrics: full label accuracy (edit operation + error type), edit operation accuracy, and error type accuracy.

The few-shot prompt clearly enhances model performance on the English datasets, with full label accuracy increasing by over 12 percentage points on EWT (from 47.5% to

Dataset	Full Label (%)	Op Only (%)	Type Only (%)
Zero-shot			
English EWT	47.50	87.50	50.83
English GUM	47.52	82.18	54.95
Finnish TDT	33.00	88.00	36.00
Few-shot			
English EWT	**60.00**	95.83	**60.42**
English GUM	51.49	87.62	54.95
Finnish TDT	32.00	**96.00**	34.00

Table 10.1: Accuracy comparison between zero-shot and few-shot prompting.

60.0%) and by nearly 4 percentage points on GUM (from 47.5% to 51.5%). Operation-only accuracy also improved significantly, reaching 95.83% on EWT and 87.62% on GUM.

Surprisingly, despite not including any Finnish examples in the few-shot prompt, the edit operation accuracy for Finnish TDT improved from 88.00% to 96.00%. This suggests that the model may generalize the concept of edit operations across languages, even when specific linguistic examples are absent.

Conversely, Finnish TDT shows no significant improvement in full label and error type accuracy, consistent with the lack of Finnish few-shot examples.

In general, the few-shot examples effectively guide the model in distinguishing between edit operations (R, U, M) and key ERRANT error types.

10.8 Limitations in the Few-Shot Experiment

Few-shot prompting led to noticeable improvements in model performance. However, the model continued to struggle with less common and syntactically complex errors, particularly in Finnish, a morphologically rich language with a distinct grammatical structure.

The 10 few-shot examples used in this study were manually crafted, based on the most frequent classification errors observed in the zero-shot experiments on the English EWT and GUM treebanks. While this approach helped address specific challenges, it introduced domain and language bias. These examples were tailored to English and may not generalize well to other languages or treebanks.

A key limitation was the absence of few-shot examples for the Finnish TDT treebank. Due to the author's limited proficiency in Finnish, no handcrafted examples could be provided for that language. This likely impacted the model's performance in Finnish. Nevertheless, the model showed some promising signs of cross-lingual generalization, particularly at the level of edit operations.

It is also important to consider the limitations of the ERRANT framework itself. Originally developed for English, ERRANT may not be directly applicable to languages like Finnish. Given Finnish's rich morphology and unique syntax, a dedicated error classification framework would likely be more appropriate.

Finally, the nature of this particular classification task adds another layer of complexity. This was not a typical binary or multi-label classification problem. Instead, it involved 75 labels, created by combining three types of edit operations with 25 main error types. This resulted in a highly complex classification task that required a deeper level of linguistic understanding. While the model's performance was not perfect, the results highlight the potential of prompt-based approaches in handling complex NLP tasks.

10.9 Concluding Findings on the Prompt Engineering Experiment

This experiment explored how prompt engineering can guide an LLM to perform grammatical error classification following the ERRANT framework. Through a series of zero-shot and few-shot evaluations on three UD treebanks (English EWT, English GUM, and Finnish TDT), the model demonstrated strong performance on frequent and surface-level error types, particularly in English datasets. Few-shot prompting led to notable improvements in classification accuracy, especially in distinguishing edit operations, and even enhanced operation-level performance in Finnish, despite the absence of language-specific examples due to the author's limited proficiency in Finnish.

However, the model continued to face challenges with more complex or less frequent grammatical categories, especially those requiring deeper syntactic or semantic under-

standing. These findings highlight both the potential and the current limitations of prompt-based approaches for detailed grammatical error classification. They also suggest that future work could benefit from more targeted examples, domain-specific prompts, and multilingual focus to further refine model performance.

11 Conclusion

11.1 Overview of the Thesis

This thesis investigates the topic of *Grammatical Error Correction Using Large Language Models: A Case Study on Universal Dependencies Treebanks*. The central goal was to explore how syntactic resources like UD treebanks, which were originally designed for linguistic annotation, can be adapted and leveraged to improve GEC systems.

The research was carried out in two main phases. The first phase focused on analyzing the `Typo=Yes` annotation in UD treebanks for English and Finnish, manually aligning these error tokens with detailed grammatical error types defined by the ERRANT framework. This effort demonstrated that although UD was not created with GEC in mind, its rich syntactic information and consistent multilingual framework can provide valuable insights and data for error correction tasks.

The second phase shifted attention to the use of LLMs for automatic error type classification. By designing structured prompts based on the ERRANT framework, the study evaluated the ability of an LLM to classify grammatical errors in both zero-shot and few-shot settings. The results showed that carefully developed few-shot prompts can improve model performance, especially in distinguishing between different edit operations and error types for English data, while also showing some positive transfer to Finnish despite limitations in language-specific examples, which were due to the author's limited proficiency in Finnish.

Together, these two phases offer a novel perspective on combining linguistic annotation frameworks with modern machine learning approaches. The findings support the

potential of UD treebanks as a multilingual and syntactically aware resource for GEC, and illustrate how prompt engineering can enhance LLM capabilities in detailed linguistic tasks, such as error classification based on the ERRANT framework, a complex classification task involving 75 different labels. This thesis thus contributes both new annotated resources and empirical insights that can inform future work on building more accurate and inclusive GEC systems.

11.2 Key Contributions

Several standard GEC datasets and error classification frameworks were explored, selecting ERRANT as the most suitable framework due to its detailed taxonomy of 75 error classes and its alignment with GEC evaluation needs.

The development of a reproducible pipeline for extracting and converting `Typo=Yes` annotations from UD treebanks into a structured, ERRANT-compatible format resulted in 542 manually annotated tokens across three UD treebanks: English EWT, English GUM, and Finnish TDT.

It was demonstrated that UD treebanks, originally developed for syntactic annotation, can be repurposed for GEC tasks through the `Typo=Yes` feature.

The broader potential of leveraging UD's multilingual consistency was highlighted to support GEC research in under-resourced and morphologically complex languages like Finnish.

A prompt engineering framework was designed and implemented to guide LLMs in grammatical error classification, using both zero-shot and few-shot prompts based on the ERRANT framework.

Ten hand-crafted few-shot examples were created for English that considerably improved LLM performance in identifying both error types and edit operations, particularly in English datasets (e.g., +12.5 percentage points full-label accuracy on EWT).

Model performance was quantitatively evaluated across zero-shot and few-shot settings, representing metrics at three evaluation levels: full label (edit operation + error

type), edit operation only, and error type only, and providing insight into LLM strengths and failure points in a complex, subtle classification task with 75 distinct labels.

A notable cross-lingual generalization effect was discovered, where edit operation classification for Finnish improved despite no Finnish examples in the few-shot prompt, suggesting inherent cross-lingual capabilities in modern LLMs.

11.3 Limitations

The `Typo=Yes` annotation in UD treebanks is sparse and inconsistently applied, which significantly limits the amount of usable data. For example, in the first phase of this thesis, during the review of previous research on the use of UD treebanks for GEC, it was noted that the Treebank of Learner English (TLE), the first publicly available syntactic treebank designed for English as a Second Language, did not contain any tokens annotated with `Typo=Yes`.

Furthermore, most UD treebanks do not contain fully corrected sentence pairs, restricting their direct applicability to standard GEC tasks. For example, in the UD Finnish TDT treebank explored during this thesis, tokens annotated with `Typo=Yes` did not include any corresponding correct form. This is not required by the UD guidelines and further limits the usefulness of such annotations.

Few-shot prompting led to noticeable improvements in model performance, yet challenges remain, especially with less common and syntactically complex errors, which are particularly difficult in a language like Finnish due to its rich morphology.

The few-shot examples used in this study were manually designed based on frequent classification issues found in zero-shot experiments on English datasets (EWT and GUM). While effective for those cases, their domain-specific nature limits their generalizability to other languages or treebanks.

Since no Finnish few-shot examples were created, mainly due to the author's limited proficiency, the model's performance in that language may have been affected. Still, some degree of cross-lingual generalization was observed, especially in edit operation

identification, which is an encouraging sign.

It is also worth noting that the ERRANT framework, central to this work to classify errors, was originally developed for English. Adapting it to Finnish is complex and likely requires a language-specific framework to account for the unique grammatical and morphological features.

Finally, this task went far beyond simple classification. With 75 detailed labels formed by combining edit operations and main error types, the model had to perform subtle linguistic distinctions. While not fully accurate, the results show clear potential for prompt-based methods in tackling complex NLP challenges.

11.4 Future Work

Future research could build on these findings by developing automated methods to align and expand `Typo=Yes` annotations across more UD treebanks. Moreover, investigating fine-tuning or domain adaptation strategies for LLMs is another important direction, particularly approaches that incorporate UD's syntactic relations directly into models or prompts. Furthermore, creating or adapting error classification frameworks tailored for morphologically complex and under-resourced languages such as Finnish is also a crucial area for future work. In addition, conducting detailed analyses of misclassifications could help improve prompt design and model performance. Finally, exploring larger-scale multilingual datasets and expanding few-shot or fine-tuning examples would likely enhance cross-lingual generalization.

11.5 Concluding Insights

This thesis demonstrates that Universal Dependencies, combined with detailed error classification frameworks like ERRANT, represent an under-explored yet promising resource for developing multilingual, linguistically aware grammatical error correction datasets.

The effective use of prompt engineering for grammatical error classification highlights new opportunities to employ large language models for subtle linguistic tasks. Although

challenges persist, especially with ambiguous or complex syntactic errors, as well as morphologically rich languages like Finnish, integrating UD's syntactic knowledge with LLM capabilities lays the groundwork for future research and the creation of more inclusive, robust GEC systems.

References

[1] 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*, pp. 1–59, Jul. 2023, ISSN: 1530-9312. DOI: 10.1162/coli_a_00478. [Online]. Available: http://dx.doi.org/10.1162/coli_a_00478.

[2] J. Nivre, D. Zeman, F. Ginter, and F. Tyers, "Universal Dependencies", in *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Tutorial Abstracts*, A. Klementiev and L. Specia, Eds., Valencia, Spain: Association for Computational Linguistics, Apr. 2017. [Online]. Available: https://aclanthology.org/E17-5001/.

[3] Universal Dependencies, *Typo — universal dependencies*, Accessed: 2025-05-07, 2024. [Online]. Available: https://universaldependencies.org/u/feat/Typo.html.

[4] C. Bryant, M. Felice, and T. Briscoe, "Automatic annotation and evaluation of error types for grammatical error correction", in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, R. Barzilay and M.-Y. Kan, Eds., Vancouver, Canada: Association for Computational Linguistics, Jul. 2017, pp. 793–805. DOI: 10.18653/v1/P17-1074. [Online]. Available: https://aclanthology.org/P17-1074/.

[5] T. B. Brown *et al.*, *Language models are few-shot learners*, 2020. arXiv: 2005.14165 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2005.14165.

[6] A. Chowdhery *et al.*, *Palm: Scaling language modeling with pathways*, 2022. arXiv: 2204.02311 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2204.02311.

[7] 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. DOI: 10.3115/v1/W14-1701. [Online]. Available: https://aclanthology.org/W14-1701/.

[8] 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*, H. Yannakoudakis, E. Kochmar, C. Leacock, N. Madnani, I. Pilán, and T. Zesch, Eds., Florence, Italy: Association for Computational Linguistics, Aug. 2019, pp. 52–75. DOI: 10.18653/v1/W19-4406. [Online]. Available: https://aclanthology.org/W19-4406/.

[9] C. Napoles, K. Sakaguchi, and J. Tetreault, "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*, M. Lapata, P. Blunsom, and A. Koller, Eds., Valencia, Spain: Association for Computational Linguistics, Apr. 2017, pp. 229–234. [Online]. Available: https://aclanthology.org/E17-2037/.

[10] D. M. W. Powers, *Evaluation: From precision, recall and f-measure to roc, informedness, markedness and correlation*, 2020. arXiv: 2010.16061 [cs.LG]. [Online]. Available: https://arxiv.org/abs/2010.16061.

[11] D. Dahlmeier and H. T. Ng, "Better evaluation for grammatical error correction", in *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, E. Fosler-Lussier, E. Riloff, and S. Bangalore, Eds., Montréal, Canada: Association for Computational Linguistics, Jun. 2012, pp. 568–572. [Online]. Available: https://aclanthology.org/N12-1067/.

[12] C. Napoles, K. Sakaguchi, M. Post, and J. Tetreault, "Ground truth for grammatical error correction metrics", in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, C. Zong and M. Strube, Eds., Beijing, China: Association for Computational Linguistics, Jul. 2015, pp. 588–593. DOI: 10.3115/v1/P15-2097. [Online]. Available: https://aclanthology.org/P15-2097/.

[13] F. Stahlberg and S. Kumar, "Synthetic data generation for grammatical error correction with tagged corruption models", in *Proceedings of the 16th Workshop on Innovative Use of NLP for Building Educational Applications*, J. Burstein, A. Horbach, E. Kochmar, *et al.*, Eds., Online: Association for Computational Linguistics, Apr. 2021, pp. 37–47. [Online]. Available: https://aclanthology.org/2021.bea-1.4/.

[14] D. Dahlmeier, H. T. Ng, and S. M. Wu, "Building a large annotated corpus of learner English: The NUS corpus of learner English", in *Proceedings of the Eighth Workshop on Innovative Use of NLP for Building Educational Applications*, J. Tetreault, J. Burstein, and C. Leacock, Eds., Atlanta, Georgia: Association for Computational Linguistics, Jun. 2013, pp. 22–31. [Online]. Available: https://aclanthology.org/W13-1703/.

[15] 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/.

[16] M. Hagiwara and M. Mita, "GitHub typo corpus: A large-scale multilingual dataset of misspellings and grammatical errors", eng, in *Proceedings of the Twelfth Language Resources and Evaluation Conference*, N. Calzolari, F. Béchet, P. Blache, *et al.*, Eds., Marseille, France: European Language Resources Association, May 2020,

pp. 6761–6768, ISBN: 979-10-95546-34-4. [Online]. Available: https://aclanthology.org/2020.lrec-1.835/.

[17] *Gnu aspell*, http://aspell.net/, Accessed: 2025-06-19.

[18] *Enchant spell-checking library*, https://github.com/rrthomas/enchant, Accessed: 2025-06-19.

[19] S. Rothe, J. Mallinson, E. Malmi, S. Krause, and A. Severyn, "A simple recipe for multilingual grammatical error correction", in *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, C. Zong, F. Xia, W. Li, and R. Navigli, Eds., Online: Association for Computational Linguistics, Aug. 2021, pp. 702–707. DOI: 10.18653/v1/2021.acl-short.89. [Online]. Available: https://aclanthology.org/2021.acl-short.89/.

[20] A. Koyama, T. Kiyuna, K. Kobayashi, M. Arai, and M. Komachi, "Construction of an evaluation corpus for grammatical error correction for learners of japanese as a second language", in *Proceedings of the 12th Language Resources and Evaluation Conference (LREC 2020)*, European Language Resources Association, 2020, pp. 204–211. [Online]. Available: https://aclanthology.org/2020.lrec-1.26/.

[21] A. Masciolini, A. Caines, O. De Clercq, *et al.*, "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/.

[22] M. A. Islam and E. Magnani, "Is this the end of the gold standard? a straightforward reference-less grammatical error correction metric", in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, M.-F. Moens, X. Huang, L. Specia, and S. W.-t. Yih, Eds., Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, Nov. 2021, pp. 3009–

3015. DOI: 10.18653/v1/2021.emnlp-main.239. [Online]. Available: https://aclanthology.org/2021.emnlp-main.239/.

[23] X. Yuan, D. Pham, S. Davidson, and Z. Yu, "ErAConD: Error annotated conversational dialog dataset for grammatical error correction", in *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, M. Carpuat, M.-C. de Marneffe, and I. V. Meza Ruiz, Eds., Seattle, United States: Association for Computational Linguistics, Jul. 2022, pp. 76–84. DOI: 10.18653/v1/2022.naacl-main.5. [Online]. Available: https://aclanthology.org/2022.naacl-main.5/.

[24] Universal Dependencies Consortium, *Universal dependencies: Introduction*, Accessed: 2025-06-12, 2023. [Online]. Available: https://universaldependencies.org/introduction.html.

[25] J. Nivre, M.-C. de Marneffe, F. Ginter, *et al.*, "Universal Dependencies v2: An evergrowing multilingual treebank collection", eng, in *Proceedings of the Twelfth Language Resources and Evaluation Conference*, N. Calzolari, F. Béchet, P. Blache, *et al.*, Eds., Marseille, France: European Language Resources Association, May 2020, pp. 4034–4043, ISBN: 979-10-95546-34-4. [Online]. Available: https://aclanthology.org/2020.lrec-1.497/.

[26] Universal Dependencies, *Typos and other errors in underlying text*, https://universaldependencies.org/u/overview/typos.html, Accessed: 2025-05-07, 2024.

[27] Universal Dependencies, *Conll-u format*, Accessed: 2025-05-07, 2024. [Online]. Available: https://universaldependencies.org/format.html.

[28] Y. Berzak, J. Kenney, C. Spadine, *et al.*, "Universal Dependencies for learner English", in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, K. Erk and N. A. Smith, Eds., Berlin, Germany: Association for Computational Linguistics, Aug. 2016, pp. 737–746. DOI:

10.18653/v1/P16-1070. [Online]. Available: https://aclanthology.org/P16-1070/.

[29] Y. Berzak, R. Reichart, and B. Katz, *Cambridge first certificate in english (fce) dataset*, TIB, Dec. 2024. DOI: 10.57702/7sqjl4ys. [Online]. Available: https://service.tib.eu/ldmservice/dataset/cambridge-first-certi-cate-in-english--fce--dataset.

[30] N. Silveira *et al.*, "A gold standard dependency corpus for English", in *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, Reykjavik, Iceland: European Language Resources Association (ELRA), May 2014, pp. 2897–2904. [Online]. Available: https://aclanthology.org/L14-1067/.

[31] Linguistic Data Consortium, *English Web Treebank*, https://catalog.ldc.upenn.edu/LDC2012T13, LDC2012T13, 2012.

[32] S. Pyysalo, J. Kanerva, A. Missilä, V. Laippala, and F. Ginter, "Universal Dependencies for Finnish", in *Proceedings of the 20th Nordic Conference of Computational Linguistics (NODALIDA 2015)*, B. Megyesi, Ed., Vilnius, Lithuania: Linköping University Electronic Press, Sweden, May 2015, pp. 163–172. [Online]. Available: https://aclanthology.org/W15-1821/.

[33] K. Haverinen, J. Nyblom, T. Viljanen, *et al.*, "Building the essential resources for finnish: The turku dependency treebank", *Language Resources and Evaluation*, vol. 48, no. 3, pp. 493–531, 2014. DOI: 10.1007/s10579-013-9244-1. [Online]. Available: https://doi.org/10.1007/s10579-013-9244-1.

[34] A. Zeldes, "The GUM corpus: Creating multilayer resources in the classroom", *Language Resources and Evaluation*, vol. 51, no. 3, pp. 581–612, 2017. DOI: http://dx.doi.org/10.1007/s10579-016-9343-x.

[35] F. Karlsson, *Finnish: An Essential Grammar* (Routledge Essential Grammars), 2nd ed. London and New York: Routledge, 1999.

[36] U. Dependencies, *Universal dependencies*, Accessed: 2025-05-11, 2025. [Online]. Available: https://universaldependencies.org/.

[37] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, *Attention is all you need*, 2023. arXiv: 1706.03762 [cs.CL]. [Online]. Available: https://arxiv.org/abs/1706.03762.

[38] C. Zhou, Q. Li, C. Li, *et al.*, *A comprehensive survey on pretrained foundation models: A history from bert to chatgpt*, 2023. arXiv: 2302.09419 [cs.AI]. [Online]. Available: https://arxiv.org/abs/2302.09419.

[39] A. Radford and K. Narasimhan, "Improving language understanding by generative pre-training", 2018. [Online]. Available: https://api.semanticscholar.org/CorpusID:49313245.

[40] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, *Bert: Pre-training of deep bidirectional transformers for language understanding*, 2019. arXiv: 1810.04805 [cs.CL]. [Online]. Available: https://arxiv.org/abs/1810.04805.

[41] C. Raffel, N. Shazeer, A. Roberts, *et al.*, "Exploring the limits of transfer learning with a unified text-to-text transformer", *Journal of Machine Learning Research*, vol. 21, no. 140, pp. 1–67, 2020. [Online]. Available: http://jmlr.org/papers/v21/20-074.html.

[42] W. X. Zhao, K. Zhou, J. Li, *et al.*, *A survey of large language models*, 2025. arXiv: 2303.18223 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2303.18223.

[43] D. Alikaniotis and V. Raheja, "The unreasonable effectiveness of transformer language models in grammatical error correction", in *Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications*, H. Yannakoudakis, E. Kochmar, C. Leacock, N. Madnani, I. Pilán, and T. Zesch, Eds., Florence, Italy: Association for Computational Linguistics, Aug. 2019, pp. 127–133. DOI: 10.18653/v1/W19-4412. [Online]. Available: https://aclanthology.org/W19-4412/.

[44] K. Omelianchuk, A. Liubonko, O. Skurzhanskyi, A. Chernodub, O. Korniienko, and I. Samokhin, "Pillars of grammatical error correction: Comprehensive inspection of contemporary approaches in the era of large language models", in *Proceedings of the 19th Workshop on Innovative Use of NLP for Building Educational Applications (BEA 2024)*, E. Kochmar, M. Bexte, J. Burstein, *et al.*, Eds., Mexico City, Mexico: Association for Computational Linguistics, Jun. 2024, pp. 17–33. [Online]. Available: https://aclanthology.org/2024.bea-1.3/.

[45] M. Yasunaga, J. Leskovec, and P. Liang, "LM-critic: Language models for unsupervised grammatical error correction", in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, M.-F. Moens, X. Huang, L. Specia, and S. W.-t. Yih, Eds., Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, Nov. 2021, pp. 7752–7763. DOI: 10.18653/v1/2021.emnlp-main.611. [Online]. Available: https://aclanthology.org/2021.emnlp-main.611/.

[46] Y. Song, K. Krishna, R. Bhatt, K. Gimpel, and M. Iyyer, "GEE! grammar error explanation with large language models", in *Findings of the Association for Computational Linguistics: NAACL 2024*, K. Duh, H. Gomez, and S. Bethard, Eds., Mexico City, Mexico: Association for Computational Linguistics, Jun. 2024, pp. 754–781. DOI: 10.18653/v1/2024.findings-naacl.49. [Online]. Available: https://aclanthology.org/2024.findings-naacl.49/.

[47] M. R. Qorib, A. F. Aji, and H. T. Ng, "Efficient and interpretable grammatical error correction with mixture of experts", in *Findings of the Association for Computational Linguistics: EMNLP 2024*, Y. Al-Onaizan, M. Bansal, and Y.-N. Chen, Eds., Miami, Florida, USA: Association for Computational Linguistics, Nov. 2024, pp. 17 127–17 138. DOI: 10.18653/v1/2024.findings-emnlp.997. [Online]. Available: https://aclanthology.org/2024.findings-emnlp.997/.

[48] X. Li and Y. Lan, "Large language models are good annotators for type-aware data augmentation in grammatical error correction", in *Proceedings of the 31st International Conference on Computational Linguistics*, O. Rambow, L. Wanner, M.

Apidianaki, H. Al-Khalifa, B. D. Eugenio, and S. Schockaert, Eds., Abu Dhabi, UAE: Association for Computational Linguistics, Jan. 2025, pp. 199–213. [Online]. Available: https://aclanthology.org/2025.coling-main.14/.

[49] M. Kobayashi, M. Mita, and M. Komachi, "Large language models are state-of-the-art evaluator for grammatical error correction", in *Proceedings of the 19th Workshop on Innovative Use of NLP for Building Educational Applications (BEA 2024)*, E. Kochmar, M. Bexte, J. Burstein, *et al.*, Eds., Mexico City, Mexico: Association for Computational Linguistics, Jun. 2024, pp. 68–77. [Online]. Available: https://aclanthology.org/2024.bea-1.6/.

[50] P. Sahoo, A. K. Singh, S. Saha, V. Jain, S. Mondal, and A. Chadha, *A systematic survey of prompt engineering in large language models: Techniques and applications*, 2025. arXiv: 2402.07927 [cs.AI]. [Online]. Available: https://arxiv.org/abs/2402.07927.

[51] C. Davis, A. Caines, O. Andersen, *et al.*, "Prompting open-source and commercial language models for grammatical error correction of english learner text", in *Findings of the Association for Computational Linguistics ACL 2024*, Association for Computational Linguistics, 2024, pp. 11 952–11 967. DOI: 10.18653/v1/2024.findings-acl.711. [Online]. Available: http://dx.doi.org/10.18653/v1/2024.findings-acl.711.

[52] S. Coyne, K. Sakaguchi, D. Galvan-Sosa, M. Zock, and K. Inui, *Analyzing the performance of gpt-3.5 and gpt-4 in grammatical error correction*, 2023. arXiv: 2303.14342 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2303.14342.

Appendix A Zero-Shot Prompt

```
You are a grammatical error annotation expert using the ERRANT (ERRor ANnotation Toolkit) classification system.

ERRANT is a framework used to categorize grammatical errors by analyzing edits between original and corrected versions of sentences. It classifies these edits into structured error types to support consistent and automated grammatical error correction.

ERRANT uses 25 main error types, each of which can be further subcategorized with one of the following edit operations:
- R: Replacement -- A token (word) in the original sentence is replaced with a different token (word) in the corrected sentence. This indicates the original word was incorrect and required substitution.
- U: Unnecessary -- A token (word) in the original sentence was removed because it was extraneous or grammatically incorrect.
- M: Missing -- A token (word) was missing in the original sentence and has been inserted in the corrected sentence to fix the error. When the correct form includes an extra word that is not present in the ungrammatical form, it clearly indicates a missing (M).

Here are the 25 ERRANT error types, their descriptions, and examples:

1. ADJ -- Adjective: big -> wide, long -> tall
2. ADJ:FORM -- Adjective Form: Comparative or superlative adjective errors. Examples: goodest -> best, bigger -> biggest, more easy -> easier
3. ADV -- Adverb: speedily -> quickly
4. CONJ -- Conjunction: and -> but
5. CONTR -- Contraction: n't -> not
6. DET -- Determiner: the -> a
7. MORPH -- Morphology: Tokens have the same lemma but nothing else in common. Example: rapid (adj) -> rapidly (adv)
8. NOUN -- Noun: person -> people
9. NOUN:INFL -- Noun Inflection: Count-mass noun errors. Example: informations -> information
```

```
10. NOUN:NUM -- Noun Number: dog -> dogs
11. NOUN:POSS -- Noun Possessive: my teachers book -> my teacher's book, my horses leg -> my horse's leg
12. ORTH -- Orthography: Case and/or whitespace errors. Example: Bestfriend -> best friend, arvin jalali@tut -> arvinjalali@tut, northeast -> Northeast
13. OTHER -- Other: Errors that do not fall into any other category (e.g., paraphrasing). Examples: at his best -> well, job -> professional
14. PART -- Particle: (look) in -> (look) at
15. PREP -- Preposition: of -> at
16. PRON -- Pronoun: ours -> ourselves
17. PUNCT -- Punctuation: ! -> .
18. SPELL -- Spelling: genectic -> genetic, color -> colour
19. UNK -- Unknown: The annotator detected an error but was unable to correct it.
20. VERB -- Verb: ambulate -> walk
21. VERB:FORM -- Verb Form: Infinitives (with or without "to"), gerunds (-ing), and participles. Examples: to walk -> walking, dancing -> danced
22. VERB:INFL -- Verb Inflection: Misapplication of verb inflection. Examples: getted -> got, fliped -> flipped
23. VERB:SVA -- Subject-Verb Agreement: (She) have -> (She) has
24. VERB:TENSE -- Verb Tense: Includes inflectional and periphrastic tense, modal verbs, and passivization. Examples: eats -> ate, eats -> has eaten, eats -> can eat, eats -> was eaten
25. WO -- Word Order: only can -> can only

---

When comparing the original token with its corrected form to determine the ERRANT error type, you must evaluate it in the context of the entire sentence. Do not rely on a surface-level comparison of the two forms alone, because context is essential for identifying the correct error category.

IMPORTANT: Your answer must strictly follow the ERRANT format:
- Start with one prefix: R: (Replacement), M: (Missing), or U: (Unnecessary)
- Then one of the 25 error categories above (e.g., SPELL, VERB:TENSE, NOUN, etc.)
- Example valid outputs: ERRANT=R:SPELL, ERRANT=M:VERB:SVA, ERRANT=U:DET
- Do NOT output just the error category without the prefix (e.g., SPELL alone is invalid)

---

Task:
You will now be shown a Sentence and a Token that contains a grammatical error as well as the Correct Form.
```

```
Your task is to classify the error by comparing the Token with the Correct Form using
    the ERRANT taxonomy provided above.

Sentence:
"{row['text']}"

Token needing correction:
- Token ID: {row['token_id']}
- Token: {row['token_form']}
- Lemma: {row['lemma']}
- Correct Form: {row['CorrectForm']}

Answer format:
ERRANT=...
```

Listing A.1: Zero-Shot Prompt for ERRANT Classification

Appendix B Few-Shot Prompt

```
You are a grammatical error annotation expert using the ERRANT (ERRor ANnotation Toolkit) classification system.

ERRANT is a framework used to categorize grammatical errors by analyzing edits between original and corrected versions of sentences. It classifies these edits into structured error types to support consistent and automated grammatical error correction.

ERRANT uses 25 main error types, each of which can be further subcategorized with one of the following edit operations:
- R: Replacement -- A token (word) in the original sentence is replaced with a different token (word) in the corrected sentence. This indicates the original word was incorrect and required substitution.
- U: Unnecessary -- A token (word) in the original sentence was removed because it was extraneous or grammatically incorrect.
- M: Missing -- A token (word) was missing in the original sentence and has been inserted in the corrected sentence to fix the error. When the correct form includes an extra word that is not present in the ungrammatical form, it clearly indicates a missing (M).

Here are the 25 ERRANT error types, their descriptions, and examples:

1. ADJ -- Adjective: big -> wide, long -> tall
2. ADJ:FORM -- Adjective Form: Comparative or superlative adjective errors. Examples: goodest -> best, bigger -> biggest, more easy -> easier
3. ADV -- Adverb: speedily -> quickly
4. CONJ -- Conjunction: and -> but
5. CONTR -- Contraction: n't -> not
6. DET -- Determiner: the -> a
7. MORPH -- Morphology: Tokens have the same lemma but nothing else in common. Example: rapid (adj) -> rapidly (adv)
8. NOUN -- Noun: person -> people
9. NOUN:INFL -- Noun Inflection: Count-mass noun errors. Example: informations -> information
```

```
10. NOUN:NUM -- Noun Number: dog -> dogs
11. NOUN:POSS -- Noun Possessive: my teachers book -> my teacher's book, my horses leg -> my horse's leg
12. ORTH -- Orthography: Case and/or whitespace errors. Example: Bestfriend -> best friend, arvin jalali@tut -> arvinjalali@tut, northeast -> Northeast
13. OTHER -- Other: Errors that do not fall into any other category (e.g., paraphrasing). Examples: at his best -> well, job -> professional
14. PART -- Particle: (look) in -> (look) at
15. PREP -- Preposition: of -> at
16. PRON -- Pronoun: ours -> ourselves
17. PUNCT -- Punctuation: ! -> .
18. SPELL -- Spelling: genectic -> genetic, color -> colour
19. UNK -- Unknown: The annotator detected an error but was unable to correct it.
20. VERB -- Verb: ambulate -> walk
21. VERB:FORM -- Verb Form: Infinitives (with or without "to"), gerunds (-ing), and participles. Examples: to walk -> walking, dancing -> danced
22. VERB:INFL -- Verb Inflection: Misapplication of verb inflection. Examples: getted -> got, fliped -> flipped
23. VERB:SVA -- Subject-Verb Agreement: (She) have -> (She) has
24. VERB:TENSE -- Verb Tense: Includes inflectional and periphrastic tense, modal verbs, and passivization. Examples: eats -> ate, eats -> has eaten, eats -> can eat, eats -> was eaten
25. WO -- Word Order: only can -> can only

---

IMPORTANT RULE ON R:ORTH vs U:ORTH
Use R:ORTH when a whitespace issue happens, especially in email addresses.
-> Example: "Ken Rice@ENRON"
Correct annotation: ERRANT=R:ORTH
Do NOT use U:ORTH for email spacing issues.

---

### Examples

**Example 1: Spacing error in email (R:ORTH)**
Sentence: "Michelle Blaine@ENRON_DEVELOPMENT"
Token ID: 1
Token: Michelle
Lemma: MichelleBlaine@ENRON_DEVELOPMENT
Correct Form: MichelleBlaine@ENRON_DEVELOPMENT

```

```
-> Explanation: "Michelle" is incorrectly separated from the rest of the email. It is replaced with the full email (no space), so this is a Replacement involving orthography (whitespace).
**ERRANT=R:ORTH**

---

**Example 2: Spacing error in another email (R:ORTH)**
Sentence: "Ken Rice@ENRON COMMUNICATIONS"
Token ID: 1
Token: Ken
Lemma: KenRice@ENRONCOMMUNICATIONS
Correct Form: KenRice@ENRONCOMMUNICATIONS

-> Explanation: "Ken" should not be separated from the rest of the email. It's replaced with the correct version of full email. This is again a Replacement and orthographic error.
**ERRANT=R:ORTH**

---

**Example 3: Apostrophe missing in noun possessive (R:NOUN:POSS)**
Sentence: "my horses foot"
Token ID: 3
Token: horses
Lemma: horse
Correct Form: horse's

-> Explanation: The intention is possession ("the foot of my horse"). "horses" should be replaced with "horse's". This is a noun possessive error.
**ERRANT=R:NOUN:POSS**

---

**Example 4: Unnecessary determiner (U:DET)**
Sentence: "It was like a workday hours."
Token ID: 4
Token: a
Lemma: a
Correct Form:

-> Explanation: "a" is an unnecessary determiner and should be removed.
**ERRANT=U:DET**

```

```
---

**Example 5: Unnecessary preposition (U:PREP)**
Sentence: "He jumped with into the water."
Token ID: 3
Token: with
Lemma: with
Correct Form:

-> Explanation: "with" is an unnecessary preposition and should be removed.
**ERRANT=U:PREP**

---

**Example 6: Noun number error (R:NOUN:NUM)**
Sentence: "The first pre-historic settlements here"
Token ID: 5
Token: settlements
Lemma: settlement
Correct Form: settlement

-> Explanation: "settlements" should be singular. This is a Replacement and a noun
    number error.
**ERRANT=R:NOUN:NUM**

---

**Example 7: Missing verb (M:VERB)**
Sentence: "Now, I would like to ask you, how these three things."
Token ID: 13
Token: things
Lemma: thing
Correct Form: things relate

-> Explanation: The correct form includes the verb "relate," which was missing. This is
    a missing verb.
**ERRANT=M:VERB**

---

**Example 8: Missing determiner (M:DET)**
Sentence: "And the first part of it, is like, well we have lecture, then we have lab."
Token ID: 14
Token: lecture
```

```
Lemma: lecture
Correct Form: a lecture

-> Explanation: The determiner "a" is missing before "lecture." This is a missing
    determiner.
**ERRANT=M:DET**

---

**Example 9: Missing auxiliary verb (M:VERB:FORM)**
Sentence: "Which way you gonna do?"
Token ID: 3
Token: you
Lemma: you
Correct Form: are you

-> Explanation: The auxiliary verb "are" is missing before "you." This is a missing verb
     form (auxiliary).
**ERRANT=M:VERB:FORM**

---

**Example 10: punctuation (R:PUNCT)**
Sentence: "It is often grouped depending on its uses that are distributed for heating
    and cooling, lighting, operating appliances etc [1]"
Token ID: 21
Token: etc
Lemma: etc
Correct Form: etc.

-> Explanation: A period is missing after "etc". This is a replacement involving
    punctuation, because "etc" will be replaced by "etc.", hence it is not missing.
**ERRANT=R:PUNCT**

---

When comparing the original token with its corrected form to determine the ERRANT error
    type, you must evaluate it in the context of the entire sentence. Do not rely on a
    surface-level comparison of the two forms alone, because context is essential for
    identifying the correct error category.

IMPORTANT: Your answer must strictly follow the ERRANT format:
- Start with one prefix: R: (Replacement), M: (Missing), or U: (Unnecessary)
- Then one of the 25 error categories above (e.g., SPELL, VERB:TENSE, NOUN, etc.)
```

```
- Example valid outputs: ERRANT=R:SPELL, ERRANT=M:VERB:SVA, ERRANT=U:DET
- Do NOT output just the error category without the prefix (e.g., SPELL alone is invalid)

---

### Task
You will now be shown a Sentence and a Token that contains a grammatical error as well as the Correct Form.
Your task is to classify the error by comparing the Token with the Correct Form using the ERRANT taxonomy provided above.

Sentence:
"{row['text']}"

Token needing correction:
- Token ID: {row['token_id']}
- Token: {row['token_form']}
- Lemma: {row['lemma']}
- Correct Form: {row['CorrectForm']}

Answer format:
ERRANT=...
```

Listing B.1: Few-Shot Prompt for ERRANT Classification

Appendix C Acknowledgement of AI Use

During the preparation of this master's thesis, I occasionally used OpenAI's ChatGPT, specifically the GPT-3.5 model available under the free plan. The main purpose of this use was for support in grammar and spell checking of sections, where I provided a portion of text and requested feedback on whether there were any major issues. This was done to polish the language, not to generate content.

Additionally, I used the tool to summarize academic papers purely for the purpose of learning and understanding their main ideas before reading them in depth. At no point was ChatGPT used as a source of academic truth or as a replacement for peer-reviewed literature. Furthermore, for cases where I encountered ambiguous or difficult concepts while reading papers, I used ChatGPT to help clarify those ideas and guide my learning outcomes.

In the technical (coding) part of this thesis, which was relatively straightforward, I also used ChatGPT occasionally, mostly to troubleshoot minor issues or clarify specific problems. However, the implementation work was generally simple and did not require heavy external assistance.

It is important to emphasize that I used AI tools critically and with awareness of their limitations. While legally permitted in this context, I made sure that AI was used in a responsible and limited way. Furthermore, the thesis goes beyond the typical scope of a master's project. I deliberately chose to explore more complex questions and conduct more detailed work than what was formally required, out of genuine academic interest

and commitment.

I would also like to note that this is my second completed master's thesis. My first was titled *Interactive Planning Tool For Global Software Projects*, completed in 2015 at the Tampere University of Technology, and is publicly available via the following link: `https://trepo.tuni.fi/handle/123456789/23927`. At that time, there were no AI tools like ChatGPT, and the academic writing in that thesis was highlighted by my supervisor as being of a high standard, especially for a non-native English speaker. I mention this to make clear that I have no difficulty in academic writing.

The narrative flow, analytical reasoning, and conclusions drawn in this current thesis are my own and reflect a unique logic and style of thinking that, to my understanding, is not replicable by current AI models. My use of AI tools was therefore supportive, not foundational.