Master Computer Science

LLM-Based Data Generation techniques for end-to-end models of grammatical error correction applied to Dutch Care Text

Name:	Floris Tomassen
Student ID:	S3108643
Date:	28/01/2024
Specialisation:	Data Science
1st supervisor:	Marco Spruit
2nd supervisor:	Gijs Wijnholds

Master's Thesis in Computer Science

Leiden Institute of Advanced Computer Science (LIACS)
Leiden University
Niels Bohrweg 1
2333 CA Leiden
The Netherlands

1 Abstract

The aim of this research, conducted on behalf of PrimeVision, is to develop an effective methodology for correcting colloquial Dutch care text. In the absence of Dutch grammatical error correction datasets, synthetic data creation through error introduction methods and text generation is explored. Experiments are performed using different seq2seq transformer models, error introduction methods and corpora of text. We evaluated 11 models trained using different error introduction methods and GPT-generated base texts through automated metrics and human evaluation. The 11 models were selected based on in-training GLEU+ scores, suspected to be corrupted. Despite this suspicion, identical ranking among peer models between post and in-training GLEU+ scores suggests a potentially reliable model selection process. Human assessment, yielding a Cohen's kappa of 0.54 (moderate agreement), and post-training GLEU+ scores identified the UL2 model in combination with interpunction & replace bigram errors as the most effective model-error combination for correcting Dutch colloquial texts. BERTScore and Sentence-BERT semantic similarity identified this model-error combination as a top performer for preserving semantic information. The final experiment compared the effectiveness of GPT-generated base texts and off-topic mC4 texts for synthetic training data using error introduction. Both human evaluation (Cohen's kappa of 0.67, substantial agreement) and GLEU+ scores consistently favored the GPT-generated text model over the mC4 model, highlighting the superior performance of generated on-topic texts. Post-training GLEU+ scores ranked model performance similiarly to human evaluation, indicating it's use for further model enhancement without time-consuming human evaluation. All code and non-sensitive data are available on Github[1].

[1] `https://github.com/skylerf1/paraphraser_code/tree/main`

Contents

2 Introduction

In the field of elderly care, care professionals interact with vast amounts of different clients every day. Care is provided by many different employees that differ significantly in level of training. All these employees have to be up to date with the situation of their clients. To achieve this, caregivers are obliged to write a report after each interaction with a client about their status. Subsequently, these reports can be read by caregivers before interacting with a client. While specific charts exist for vital information such as prescribed medication, these reports are important to remain informed about a client. Additionally, the reports are used to inform the family of clients about their situation. The reports are also used to assess the level of care required for the client and plan future care.

Writing reports in Dutch poses a challenge for care professionals. Without explicit formatting guidelines, the records lack structure. Additionally, the demanding workload leads to hastily written, sloppy and ungrammatical notes, especially for those with lower level of proficiency in Dutch.

On behalf of PrimeVision, we aim to create a tool which can improve the clarity of care professional's reports while maintaining the meaning. This would contribute to making reports more legible and understandable. This tool aims to elevate the experience of reading reports for both employees and family members, ultimately enhancing the overall service quality. This tool should enhance care professionals' reports by generating improved and more readable versions. It would then prompt caregivers to confirm whether the revised text maintains the original message or if further adaptation is required. Figure 1 shows an example report along with possible corrections.

Original Sentence: Kennisgemaakt met mw. en 3 zoons mw. komt uit Pentagon 17 oost. Altijd gezonde dame. Collumfractuur waarvoor revalidatie in Pentagon. Dementie gediagnostiseerd. Doofheid waarvoor 2 hoortoestellen. Galblaas wordt niet geopereerd.

Improved Sentence: Ik heb kennisgemaakt met mevrouw en haar 3 zoons. Mevrouw woont in Pentagon 17 oost. Ze is altijd een gezonde dame geweest. Ze heeft een collumfractuur waarvoor ze revalidatie ontvangt in Pentagon. Er is dementie gediagnostiseerd. Ze heeft last van doofheid waarvoor ze 2 ge hoortoestellen heeft. Haar galblaas wordt niet geopereerd.

Figure 1: Anonimized report and a possible improved report.

For this tool, a system is required which can generate a corrected text, based on incorrect input text. The field of research closest associated to this is grammatical error correction (GEC). Despite it's name this field focuses on the correction of all types of errors in text, not just grammatical errors. Currently, neural machine translation (NMT) models are the best at GEC (Qorib et al., 2022; Sun and Wang, 2022; Omelianchuk et al., 2020; Rothe et al., 2021). To correct input text, they usually utilize sequence-to-sequence (seq2seq) and sequence-tagging deep learning models. An additional challenge for this research is the fact that most GEC research is focused on English text, with some research in Arabic (Mohit et al., 2014), Chinese (Lee et al., 2018), Czech (Náplava and Straka, 2019), Russian (Rozovskaya and Roth, 2019), Ukranian (Syvokon and Nahorna, 2021), German (Boyd, 2018) and Japanese (Koyama et al., 2020). There is currently no Dutch language GEC dataset or research conducted using NMT. Therefore, we first need to create synthetic training data. To create this tool, this research will delve into deep learning models, GEC for low-resource languages and methods to generate synthetic training data. This research aims to address the following questions:

Main Research Question: What is the effectiveness of different error introduction methods on LLM-based generated Dutch care texts for enhancing the performance of end-to-end deep learning models for correcting colloquial Dutch care text while preserving the semantics of the original text?

- **Subquestion 1:** How should we generate our Dutch care texts?
- **Subquestion 2:** How can we assess the performance of the models, including the generated Dutch care texts and error introduction methods?
- **Subquestion 3:** How can we determine the most suitable error introduction methods and models for effectively correcting colloquial Dutch care text?

To address Subquestion 1, methods for creating synthetic training data for seq2seq models are discussed. This is followed by a discussion of experiments with Generative Pre-trained Transformers (GPT) as a means to generate annotations, errorful text, and on-topic texts.

To answer Subquestion 2, different evaluation methods to assess corrections created by a model are explored. This will also include metrics to measure semantic preservation between original sentence and candidate sentence. Both human and automatic metrics will be used.

To answer Subquestion 3, we will discuss different error introduction methods and models of prior research. We will also go into an experimental design using the evaluation metrics from Subquestion 2 to select the most suitable models and methods. Finally we will discuss if it is possible to rely solely on automated metrics as human evaluation is highly time consuming.

Firstly, we will delve into the theoretical framework (3) of this field. Subsequently, we will elaborate on our methodology (4), followed by a description of the experimental setup. Next, we will present the results (5), which will be thoroughly discussed in the following section (6). Finally, we will conclude our study (7).

3 Theoretical Framework

The goal of this research is to correct Dutch care reports. Therefore we will discuss the way grammatical error correction (GEC) is approached in prior research since this is the most similar task at hand. In order to utilize end-to-end neural models effectively, a significant amount of data is required. While we have access to a large corpus of care reports, these lack annotations which are required to train models. Furthermore, we do not have the capacity to label large amounts of care reports manually due to the expensive nature of the task. Consequently, we will go into the process of synthetic data generation and explore relevant data augmentation techniques. This is particularly crucial due to the non-existance of Dutch GEC data.

Next, methods of evaluation for GEC systems will be discussed, since quality control of text correction and text generation is not straightforward. Furthermore, automated evaluation metrics as well as human evaluation are looked into. Finally we look into semantic preservation and if we can incorporate semantic similarity metrics in our solution.

3.1 Grammatical Error Correction

Grammatical Error Correction (GEC) is vital in correcting errorful text, therefore it is a key focus of this research. GEC considers the task of detecting and correcting errors in text automatically. These errors can be all types of mistakes including spelling errors and semantic errors, not just grammatical errors, as the name suggests. In practice, the goal of GEC is to create error-free text from errorful text while maintaining the same intended meaning, using minimal modifications. This goal is less ambitious than the example in Figure 1 which aims to include contextual information, even if this is not present in the text itself. Furthermore, GEC usually operates at the sentence level, so cross-sentence contextual information is lost. GEC has a long history of rule-based methods, statistical classifiers and statistical machine translation. However the current state-of-the-art systems are based on NMT models (Qorib et al., 2022; Sun and Wang, 2022; Omelianchuk et al., 2020; Rothe et al., 2021), which will be the focus of this paper.

The foundation of modern GEC models is data. Generally GEC data consists of text which is badly written and annotated by an annotator with corrections. Due to the nature of the GEC task acquiring annotated data is an expensive and slow process since erroneous text has to be carefully read and corrected. Furthermore the text can be corrected in numerous ways and the chosen correction is subjective to the annotator (Bryant and Ng, 2015; Choshen and Abend, 2018a). To normalize the annotation method as much as possible, annotation guidelines are made. The most popular methods are minimal corrections and fluent corrections (Table 1). The idea of minimal correction is to change as little as possible to create a grammatically sound text. Most GEC datasets are annotated according to the minimal corrections method. However Sakaguchi et al. (2016) argue that this leads to unnatural corrections and they come up with the idea of fluent correction. Fluent correction aims to correct a sentence by optimizing any additional unnatural aspects of text. This usually means changing a word to a different one. Instead of building a sentence around a misused word, as minimal correction would do, fluent correction improves the word choice. Minimal correction guidelines are generally easier to use for annotators than fluent corrections although fluent corrections are more desirable. Both methods have their shortcomings but aim to give an annotator some clearly defined guidelines. These guidelines do not take away the labour intensive nature of manually annotating erroneous text in a consistent manner. This results in there being less available data in the field of GEC compared to similar tasks like machine translation, especially in low-resource languages. There is only one Dutch GEC dataset as far as we can tell which is the Dutch section of the lang-8 dataset (Tajiri et al., 2012). The problem with this dataset is the fact that it is corrected by users instead of professionals, and therefore contains many errors. It also consists of at most 588 sentences, since many German texts are mislabeled as Dutch as well.

Original	She always hand me good advices when I need it.
Minimal corrections	She always hands me good advice when I need it.
Fluent corrections	She always offers me valuable advice when I need it.

Table 1: Minimal and Fluent grammatical error corrections examples

3.2 Levenhstein distance

The Levenshtein distance (Levenshtein et al., 1966), or edit distance, is the minimum number of single-character edits needed to transform one string into another. These character edits include insertions, deletions, and substitutions. Calculated through dynamic programming, it reflects the minimal cost of transformations. This measure is used in many evaluation metrics and error-introduction methods.

3.3 Synthetic Data

Given the data sparsity inherent in GEC tasks, extensive research has been conducted to generate synthetic data. This synthetic data is intended to address the demand for substantial datasets required by neural machine translation models. Given the scalability of neural models with increased data, the acquisition of additional data that enhances the performance of the model holds significant value. This is especially crucial for low-resource languages where alternative data sources are limited, therefore the generation of data becomes even more valuable.

The original idea of creating synthetic data for GEC is to introduce errors to correct text. Usually, correct text is found from websites, books, articles or other sources of clean text which are similar to the type of text that needs to be corrected down the line. Research by Felice and Yuan (2014) underscores the significance of considering the context of the text, as it plays a significant role in influencing the quality of the dataset used for GEC. By corrupting a clean sentence a 'learner' sentence is created which is paired along with the original sentence which serves as a reference. There are numerous ways to generate this noise but the dominant methods fall under noise injection and back-translation (Kiyono et al., 2019). We will discuss these methods in the next sections.

3.4 Noise injection

Noise injection is the idea of injecting errors into clean text corpora. The injected noise is either a pattern of errors which is common in the downstream GEC task or rule-based error.

3.4.1 Rule-based

Rule-based error is created by defining rules which apply changes to the clean text based on linguistic knowledge. Each rule has a probability associated with it which controls when it will apply. These probabilities can be based on the available data distribution, empirical experimentation or can be chosen arbitrarily (Bryant et al., 2022). The introduced error can be focused on the character or word level. Lichtarge et al. (2019) introduce spelling errors to Wikipedia edit history by replacing, inserting, deleting and swapping characters. In their study, Zhao et al. (2019) employ comparable techniques focusing on the word level. These techniques encompass the deletion, insertion, shuffling, and replacement of words within a sentence. Combinations of word and character noising were used by the succesful approach of Grundkiewicz et al. (2019). Rule-based error introduction can be used both offline to generate more training data or during training to increase the error rate in a parallel corpus instead of creating additional data (Zhao and Wang, 2020).

3.4.2 Error patterns

Another method to introduce errors to clean text is by identifying and injecting error patterns which frequently occur in relevant text. The advantage of this method is that the introduced errors are relevant and more similar to human error. Rozovskaya and Roth (2010) introduced three distinct approaches for incorporating article errors, drawing from the error distribution observed in English as a Second Language (ESL) data. These approaches involve adding article errors based on the distribution of articles in the original text, the distribution of articles in the corrected text, and the distribution of article corrections themselves. Yuan and Felice (2013) explored an alternative approach to mimic human errors by extracting correction patterns from GEC datasets. They applied the inverse of those corrections on grammatically accurate sentences, taking inspiration from the corrections found in the NUCLE corpus (Dahlmeier et al., 2013). These methodologies require detailed GEC datasets which is a problem for low-resource languages.

3.5 Back-translation

A more automated way of generating human error is by using a noisy channel model which is essentially an inverted GEC model. Instead of using a learner sentence as the source and a reference sentence as the target as training data, the learner sentence becomes the target with the reference as the source. The method was initially introduced as a means to generate supplementary data in the domain of machine translation (Sennrich et al., 2016). However, its applicability extends seamlessly to GEC. This approach was applied to GEC first by Xie et al. (2018). While this is a valuable research direction it will further be excluded from this research since there are no Dutch GEC models nor datasets which can be used for this approach.

3.5.1 Round-trip translation

An alternative approach to back-translation, which is less commonly used, is round-trip translation. This method generates synthetic sentence pairs by utilizing a bridge language, such as English-Japanese-English (Madnani et al., 2012). The key idea here is that the translation model will not be perfect and introduce translation errors to the text. This will create new learner sentences along with the correct original, expanding training data. Zhou et al. (2019) introduce the idea of using two translation models, a low and a high quality model, and using these to generate a learner and reference sentence respectively. This can be of value if there is a relevant dataset in a different language for additional data.

3.6 Low-resource approaches

Kiyono et al. (2019) perform extensive experimentation with the settings for generating synthetic data. They corrupt text using both back-translation (BackTrans) and noise injection (DirectNoise). BackTrans was introduced by Xie et al. (2018). It is a modification of back-translation that incorporates noise into the scoring process. The method involves adding a randomly sampled noise term, β_{random}, to the score of each hypothesis in the beam at every time step. In this case, the noise value r is uniformly sampled from the range [0, 1], while $\beta_{random} \in \mathbb{R} \geq 0$ is a hyper-parameter that determines the scale of the noise. If β_{random} is set to 0, BackTrans becomes equivalent to the standard back-translation method.The DirectNoise method entails inserting different types of noise. First of all using masking with the placeholder token '`<mask>`'. Secondly, deleting tokens. Thirdly, inserting a random token. Finally, keeping the original token. For each token, the choice is made based on the categorical distribution (μmask, μdeletion, μinsertion, μkeep). The best settings they found in their experimentation are: (μmask, μdeletion, μinsertion, μkeep) = (0.5, 0.15, 0.15, 0.2). These settings are used together with a large database of clean text to generate pseudo data, used for pre-training a model. The best settings this approach found lead to severely corrupted text.

Náplava and Straka (2019) research GEC in low-resource scenarios using synthetic data. Their method is based on Grundkiewicz et al. (2019). They draw a probability p_{error_word} from a normal distribution with a predefined mean and standard deviation. They multiply this probability with the amount of words in a sentence and get the number of words to introduce permutations to. These permutations are selected based on predefined probabilities. For the model to be robust they also apply the same noise to characters based on p_{error_char} sampled from a different distribution. They assigned a value of 0.15 to p_{error_word} and 0.02 to p_{error_char}. To estimate the error distributions of individual operations, they used the development sets of each language. They used the permutations described below.

- Substitution: substituting using spellchecker suggestions/random character with a probability p_{word_sub} 0.6-0.7 and p_{char_sub} 0.2-0.25
- Deletion: deleting with a probability p_{word_del} 0.05-0.1 and p_{char_del} 0.2-0.25
- Swap: swapping the word with it's right-hand neighbour with a probability p_{word_swap} 0.01-0.1 (only for words)
- Insertion: inserting a random word after the selected word with a probability p_{word_sub} 0.1-0.2 and p_{char_ins} 0.2-0.25
- Capitalization: removing or adding capitalization with a probability p_{word_cap} 0.2-0.25 and p_{char_cap} 0.2-0.25

White and Rozovskaya (2020) conducted a comparative analysis of the synthetic data generation techniques employed by the top-performing submissions in the restricted and low-resource tracks of the BEA-2019 Shared Task on Grammatical Error Correction (Bryant et al., 2019). UEDIN-MS (Grundkiewicz et al., 2019) and Kakao&Brain (Choe et al., 2019) performed best on these tracks. UEDIN-MS utilized an inverted spellchecker to introduce noise to clean text. The spellchecker suggests a list of words for a word. These words are then ordered based on the Levenshtein (Levenshtein et al., 1966) based weighted edit distance and phonetic distance. This results in a confusion set consisting of 20 words. In this approach, the number of words to modify in each sentence is determined by considering the word error rate observed in the real dataset. For each selected word, one of the following operations is applied:

- With a probability of 0.7, the word is replaced with a randomly chosen word from the confusion set.
- With a probability of 0.1, the word is deleted.
- With a probability of 0.1, a random word is inserted.
- With a probability of 0.1, the position of the word is swapped with an adjacent word.

These operations are also applied at the character level to 10% of the words to generate spelling errors. This method was created to introduce general noisy data to train a general purpose GEC model.

Kakao&Brain use two noising methods. First of all they use a small sample of real data with annotated error patterns to capture common errors. These common errors are then used on generated data in reverse to generate errors. Secondly, if a word does not have an associated common error associated to it, they apply a type based noising scheme based on parts-of-speech (POS). Only prepositions, nouns and verbs are selected with a probability of 0.15. During the transformation process, propositions are substituted with alternative propositions, nouns undergo either pluralization or singularization, and verbs are morphologically modified. According to experiments from White and Rozovskaya (2020), the method of Kakao&Brain performs better than UEDIN-MS, especially in conjunction with an off the shelf spellchecker during preprocessing.

The sequence-tagging approach presented by Omelianchuk et al. (2020) for GEC differs from traditional seq2seq GEC models. Their system uses a pre-trained language model (BERT) and 9 million synthetic sentence pairs to train a modified BERT model (Devlin et al., 2018). While seq2seq models generate the corrected text from an input text, Omelianchuk et al.'s approach generates a sequence of edits which have to be applied to the input text, resulting in a ten-fold increase in inference time. To address the challenge posed by token-based edit operations, which may sometimes fall short in capturing more complex, multi-token fluency edits (Lai et al., 2022), Omelianchuk et al. employ multiple iterations of correction. Omelianchuk et al.'s method reaches near state-of-the-art performance with a significant inference speed reduction.

Rothe et al. (2021) introduce a simple recipe for multilingual grammatical error correction. They apply a language agnostic method to generate a large amount of synthetic pre-training data. Their method consists of using the mC4 corpus and splitting all the paragraphs into sentences. 98% of the sentences are corrupted using a combination of the following operations: dropping spans of tokens, swapping tokens, dropping spans of characters, swapping characters, inserting characters, lower-casing or upper-casing the first character of a word. 2% of the sentences are left unaltered to teach the model that input can be correct. They do not share the exact probabilities of these operations. Since this approach is focused on multilingual applications they refrain from using language specific text corruption methods because those methods would be difficult to apply to all languages. Additionally, Rothe et al. (2021) tried out three options for pre-training their seq2seq models: pre-training with a mix of synthetic GEC data and hand-annotated gold standard data, with and without different prefixes to signal what type of data it is to the model. Finally they tried using synthetic GEC data pre-training until convergence and fine-tuning on gold standard data afterwards. While the latter is the most computationally expensive, it performed the best.

3.7 Evaluation

To measure the performance of GEC systems, evaluation metrics are used. Due to the complicated nature of comparing pieces of text, this is an active area of research and there are many different evaluation metrics. We will discuss the most used and relevant evaluation metrics for our research. There is a distinction between reference-based evaluation metrics and reference-less metrics. Reference-based metrics are generally more powerful, while reference-less metrics are easier to apply, but tend to be less powerful.

3.7.1 Maxmatch

Maxmatch (M^2) scorer (Dahlmeier and Ng, 2012) is a reference-based evaluation metric. It compares hypothesis corrections and human annotated corrections, where a correction consists of a set of edits. An edit consists of a word or span of words from an original sentence along with a correction, for example for the original sentence 'The car has driving to the store by Joe.' an edit can be [has driving $\rightarrow$ was driven]. When an hypothesis edit matches the human correction it is considered a true positive (TP). If a hypothesized edit is not present in the reference it is considered a false positive (FP) and if a reference edit is not present in the hypothesis it is considered a false negative (FN). These TP, FP and FN can be used to calculate precision and recall. Precision and recall represent the amount of hypothesis edits that were correct and the amount of reference edits that were present in the hypothesis respectively. Precision and recall can be used to calculate the F_β score. In GEC research it is standard to use $\beta = 0.5$ which weights precision twice as much as recall since it is generally considered more important to be accurate than to capture all errors. A problem of edit based measures is that there is often numerous ways to define the same edit. e.g an edit can be considered [has driving $\rightarrow$ was driven] as well as [has $\rightarrow$ was] and [driving $\rightarrow$ driven]. Traditionally these were not considered TP however M^2 employs Levenshtein alignment (Levenshtein et al., 1966) between original text and the hypothesis which makes it possible to consider all ways to combine edits so the hypothesis edits maximally match the reference edits. M^2 does require specific annotation methods to acquire the edits.

$$P = \frac{TP}{TP + FP}$$

$$R = \frac{TP}{TP + FN}$$

$$F_\beta = \frac{(1 + \beta^2) \times P \times R}{(\beta^2 \times P) + R}$$

3.7.2 ERRANT

ERRANT is similar to M^2 as it is also a reference- and edit-based evaluation metric. ERRANT expands on M^2 by going into detail and judging systems based on their performance on specific error types. It uses a Damerau-Levehnstein alignment algorithm to filter edits from a hypothesis text (Felice et al., 2016). ERRANT then uses rule based framework to classify the changes to error types. This makes it possible to calculate F_β scores for each error type individually. It has three levels of error distinction listed below. ERRANT was initially developed for the English language and has been extended to multiple languages. However, Dutch is not currently supported. This is currently the most detailed error type specific error metric.

- Edit Operation (3 labels): Replacement, Missing, Unnecessary
- Main type (25 labels): e.g., Verb, Noun, Verb Tense, Spelling
- Full type (55 labels): e.g., Replacement Verb, Missing Noun, Unnecessary Verb

3.7.3 Generalized Language Evaluation Understanding

Generalized Language Evaluation Understanding (GLEU) (Napoles et al., 2016a) is another reference based metric, inspired by BLEU (Papineni et al., 2002). Shortly after initial publication, GLEU+ was introduced in Napoles et al. (2016b), improving their methodology. It differs from M^2 and ERRANT since it only requires a corrected reference text and not explicit edit annotations. The core idea behind GLEU+ is to recognize and reward hypothesis n-grams that align with the reference but differ from the original text, while also imposing penalties on hypothesis n-grams that match the original text but differ from the reference. In equation 1 the calculation of GLEU+ precision p_n is shown. Assume a set of original sentences $O = \{o_1, \ldots, o_k\}$ and their respective hypothesis sentences $H = \{h_1, \ldots, h_k\}$ and reference sentences $R = \{r_1, \ldots, r_k\}$. $o_i, h_i,$ and r_i represent the sequences of n-grams of length $n = \{1, 2, \ldots, N\}$ (where $N = 4$ is the default in GLEU+) present in the sentences instead of the sentences itself. These n-gram sequences $o_i, h_i,$ and r_i are then utilized to compute a precision term p_n which considers the intuitive idea of rewarding and punishing n-gram overlap. If more than one reference sentence is available, a random reference is selected and the score is averaged over 500 iterations of this random selection.

$$p_n = \frac{\sum_{i=1}^{|H|} \left(\sum_{g \in \{h_i \cap r_i\}} \text{count}_{h_i,r_i}(g) - \sum_{g \in \{h_i \cap o_i\}} \max[0, \text{count}_{h_i,o_i}(g) - \text{count}_{h_i,r_i}(g)] \right)}{\sum_{i=1}^{|H|} \sum_{g \in \{h_i\}} \text{count}_{h_i}(g)}$$
$$\text{count}_a(g) = \text{number of occurrences of n-gram } g \text{ in } a$$
$$\text{count}_{a,b}(g) = \min(\text{number of occurrences of n-gram } g \text{ in } a, \text{number of occurrences of n-gram } g \text{ in } b) \quad (1)$$

3.7.4 Human evaluation

Various approaches have been suggested for the human assessment of system outputs. For GEC, this was initially done by ranking corrections of various systems against each other using the Appraisal framework (Federmann, 2012). This framework involves ranking randomly chosen samples of n system hypotheses to get pairwise judgments and calculate an overall system ranking. This methodology was later deemed unreliable due to varying correlation coefficients of research while they essentially performed the same experiment (Bryant et al., 2022). Choshen and Abend (2018b) found that these methods were unreliable partially due to low inter-annotator agreement (IAA). In their exploration of reliable methods for collecting human ratings of competing systems, Sakaguchi and Van Durme (2018) and Novikova et al. (2018) found that partial ranking with scalers (PRWS) works best. The method involves presenting participants with the original sentence and n corrected versions, along with a continuous scale to score the sentence. they both found that evaluators implicitly adjust their scores for each hypothesis to be relative to the other hypotheses, making the numeric scores more discriminative.

This method was adopted by Napoles et al. (2019) for their human evaluation. They selected hypothesis sentences from five models with a respective continuous scale from 0 (Completely ungrammatical) to 100 (Perfect). They introduced 3 'systems' as a baseline: source sentence, randomly chosen reference sentence and the source sentence with 1-2 errors introduced. They ensured to perform human and automatic evaluation on their entire dataset to adress the criticism of inconsistant sampling from Choshen and Abend (2018a). These judgments on the continuous scale were then used to calculate human scores for each sentence per system by averaging over the 8 evaluators.

To determine the effectiveness of automated metrics, extensive research has been conducted with the premise that the best metric is the one that exhibits the highest correlation with ground-truth human judgments. Initial studies by Napoles et al. (2015) and Grundkiewicz et al. (2015) used the Appraise evaluation framework (Federmann, 2012). Subsequent research by Napoles et al. (2019) acknowledged the subjectivity issue brought forward by Choshen and Abend (2018b) and introduced a continuous scale and collected evaluations on all sentence-pairs to combat sampling bias. They discovered that the dataset does impact metric performance, likely due to inconsistent human judgments across different error type distributions. Bryant et al. (2022) note that using ground-truth human judgments as a benchmark for metric performance seems like an intuitive approach, however it's highly subjective and should be taken with caution. The case of the I-measure illustrate this well. It was initially thought to have a weak correlation with human judgments (Grundkiewicz et al., 2015), then turned out to have a good correlation at the sentence level (Napoles et al., 2016b), and finally was considered the best metric across different domains (Náplava and Straka, 2019). This inconsistency highlights the ongoing challenge of figuring out reliable methods for evaluating automatic metrics.

3.7.5 Semantic similarity

There are two types of semantic similarity evaluation; human and automatic evaluation (Zhou and Bhat, 2021). We will first go into automatic evaluation metrics to assess semantic similarity. First of all BLEU was used to judge the performance of GEC models. BLEU was originally made to evaluate machine translation models. BLEU focuses on precision of the n-gram overlap between generated and reference (Papineni et al., 2002). Secondly Rouge is used to capture the recall and other aspects such as skip-bigrams and the longest common subsequence from the original sentence within the paraphrase (Lin, 2004). However, both BLEU and ROUGE are unable to assess semantic similarity between words since they are based on n-grams and just look for identical text. Metric for Evaluation of Translation with Explicit ORdering (METEOR) (Banerjee and Lavie, 2005) attempts to capture the semantic equivalents between sentences. This is done based on unigram matching, including stemming and dictionary-based synonymn search. However, for the Dutch language no synonymn dictionary is present for this metric.

Recently contextualized text embeddings have been used to compare semantic similarity between texts. Contextualized text embeddings consist of different high-dimensional representations for the same word in different texts, depending on the the adjacent words. BERTScore (Zhang et al., 2019a) is an evaluation method which uses the contextualized text embeddings directly from a BERT model (Devlin et al., 2018). BERTScore computes pairwise cosine similarity between each token in the hypothesis and reference text (Figure 2). Next each hypothesis token is matched to their most similar reference token, to capture both precision and recall between the texts. Optionally, the inverse document frequency (IDF) can be used to increase the weight of rarely used words, assuming they are highly indicative of meaning. Finally, BERTScore is calculated by averaging the cosine similarity between all best found token pairs, with or without IDF weighting. This results in a score between 0 and 1 of how similar a text is. Which BERT model layer performs best for the contextualized sentence embeddings is based on tuning using the WMT16 metric evaluation dataset (Bojar et al., 2016).

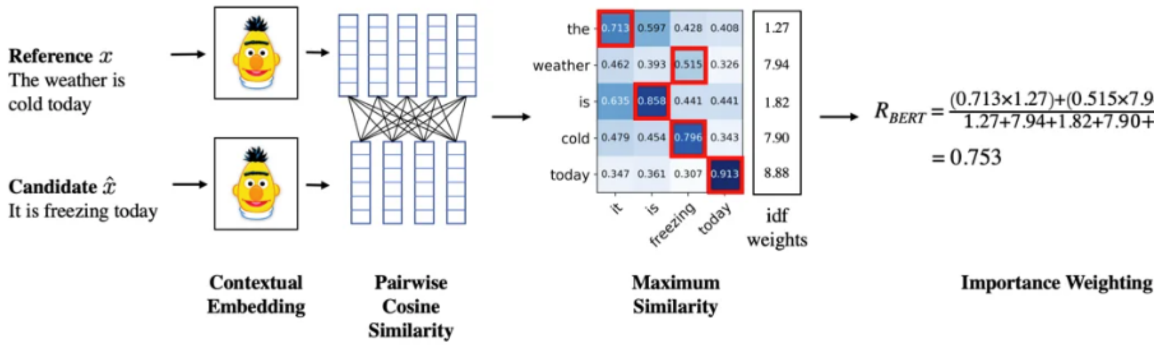


Figure 2: How BERTScore works (taken from Zhang et al. (2019a))

Contextualized embeddings have been shown to perform well at paraphrase detection as well as being able to capture distant dependencies (Devlin et al., 2018). BERTScore is also more robust to adversarial data from the PAWS high lexical overlap dataset (Zhang et al., 2019b). BERTScore is available in multiple languages, including Dutch. However, only a multilingual model has been properly tested for the application in the Dutch language.

Another method for generating contextualized sentence embeddings is Sentence-BERT (SBERT) (Reimers and Gurevych, 2019). SBERT is a modified version of BERT designed to efficiently generate semantically meaningful sentence embeddings. Unlike BERT, SBERT focuses on enabling tasks like semantic similarity search and clustering with reduced computational time. It achieves this by using siamese and triplet network structures, which allows sentence embeddings to be compared using cosine similarity efficiently. In comparison to BERTScore, which evaluates the quality of machine-generated text, SBERT is tailored for generating embeddings and assessing semantic similarity, offering a more efficient approach for tasks like similarity search.

Given the focus of automatic evaluation metrics on n-gram overlap rather than meaning, human evaluation is used for a more accurate and qualitative assessment of the generated output. In human evaluation, annotators are tasked with scoring sentences on similarity. Despite the manual efforts involved, human evaluation, although costlier than its automatic counterpart, emerges as a more representative measure of the generated output's quality.

3.8 Models

The focus of this research is on pre-trained seq2seq transformer models that have been pre-trained on Dutch text. This pre-training process involves training the model on raw text data without any human annotations or labels. This approach allows the model to leverage a vast amount of publicly available text data. The models use an automatic process to derive inputs and corresponding outputs from this text corpus, making it a valuable tool for a wide range of NLP tasks.

3.8.1 Unifying Language Learning model

The 'Unifying Language Learning' (UL2) model has a classic T5 model architecture (Raffel et al., 2020). The T5 (Text-to-Text Transfer Transformer) model, operates as an encoder-decoder architecture, and it approaches all natural language processing (NLP) problems in a text-to-text format. The model closely resembles the original Transformer architecture proposed by Vaswani et al. (2017), with a few changes. Specifically, they eliminate the layer norm bias and relocate layer normalization outside the residual path. They also adopt an alternative position embedding scheme. It's trained using the UL2 Mixture-of-Denoisers (MOD) objective (Tay et al., 2022).

The UL2 MOD objective consists of R-denoising, X-denoising and S-denoising (Figure 3). R-denoising (or regular span corruption) is the original T5 span corruption objective. It masks a range of 2 to 5 tokens, which results in around 15% of the input tokens being masked. The goal of this short span denoising method is to acquire knowledge about language. X-denoising is an extreme form of denoising since the model must reproduce much of the input, while it only has a small amount of it. This scenario simulates a situation where the model is tasked with generating an extensive output from a memory source that contains a comparatively limited amount of information. To do this 50% of the input tokens are masked. This is done by increasing the corruption rate per token or increasing the span length of masking. A pre-training task is considered extreme when it has a long span (>11 tokens) or a high corruption rate (>30%). It is to used interpolate between a downstream language model objective and regular span corruption. S-denoising is a specific case of denoising. Here a sequential order is maintained between the inputs to target. The input text is divided into two sub sequences and the first part is always the context and the final part is the target. This means that the target tokens never rely on future information. This denoising method is different because a target token can never appear before the context. The model is provided with a specific prefix for each type of denoising, to inform what type of denoising it should consider for an input text. During inference the UL2 model also expects a prefix. For general language understanding fine-tuning tasks, usually the R-denoising prefix is used.

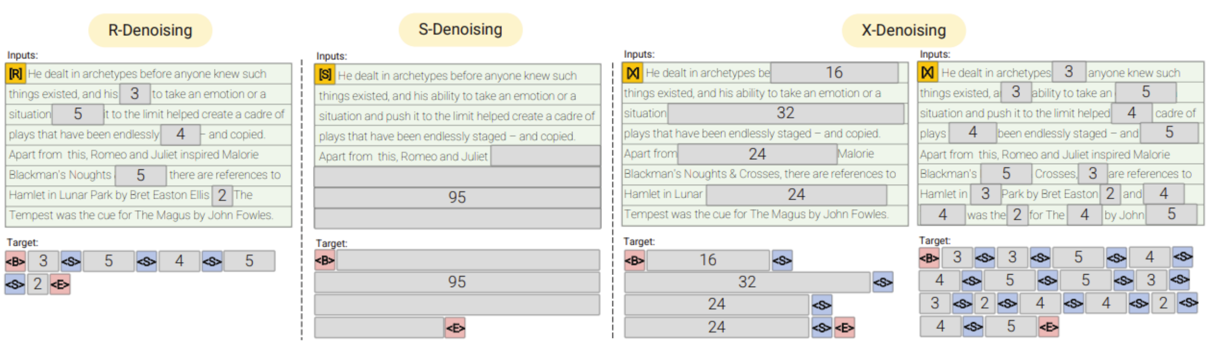


Figure 3: Visualisation of R,S and X-denoising (taken from Raffel et al. (2020))

3.8.2 Multilingual Bidirectional and Auto-Regressive Transformers model

Multilingual Bidirectional and Auto-Regressive Transformers (mBART) (Liu et al., 2020) is a denoising auto-encoder with a seq2seq architecture, pre-trained on monolingual datasets across multiple languages, utilizing the BART pre-training methodology (Lewis et al., 2020). This pre-training objective uses text corrupted by using masking, sentence permutation, document rotation, token deletion and span prediction. Sentence permutation swaps sentences around, span prediction selects 35% of the tokens based on a Poisson distribution and masks these. Document rotation entails that the sentences are shuffled randomly. The model uses the standard seq2seq Transformer setup (Vaswani et al., 2017). This method is employed for many languages in the same model; including Dutch.

3.9 Large Language Models

Large Language Models (LLMs) have emerged as valuable tools in natural language processing. They demonstrate significant advancements in understanding and generating human-like text (Brown et al., 2020). These models utilize deep learning and are often pre-trained on massive datasets to acquire a comprehensive understanding of language patterns, structures, and semantics.

3.9.1 Generative Pre-trained Transformers

The most well-known version of LLMs is the family of Generative Pre-trained Transformers (GPT), developed by OpenAI. GPT relies on the transformer architecture for capturing intricate linguistic features supported by unsupervised learning, where the model is pre-trained on diverse and extensive textual corpora, enabling it to learn the intricacies of language without task-specific annotations.

GPT operates autoregressively, generating text one token at a time based on the context of input and the generated text so far. This is facilitated by self-attention mechanisms in the transformer architecture (Vaswani et al., 2017), which allows GPT to consider an entire input sequence when predicting the next token. The attention mechanism enables GPT to capture long-range dependencies and nuanced context. This results in coherent text generation, exemplified in the popular ChatGPT model.

For this research we will discuss GPT-3 over GPT-2 and GPT-4 since GPT-2 is outdated and GPT-4 was unavailable at the beginning of this research. GPT-3 is a 175-billion parameter transformer model (Vaswani et al., 2017), including modified model initialization, pre-normalization and reversible tokenization (Brown et al., 2020). GPT-3 is pre-trained on an extensive dataset consisting of a filtered version of CommonCrawl, the English-language Wikipedia combined with high-quality reference corpora. According to Brown et al. (2020), GPT can understand language at different levels, recognizing grammar, meaning, and practical usage. This versatility makes GPT a valuable asset for numerous natural language processing tasks, including text generation and language understanding.

GPT-3 can perform few-shot and zero-shot learning. Zero-shot learning consists of instructing a model on a task using only words, without any examples. Few-shot learning involves providing examples of the desired task to the model. Based on these instructions GPT-3 can generalize the information and is capable of performing various tasks described in this manner.

In the context of this research, GPT-3 is leveraged to generate synthetic Dutch care text, contributing to the creation of a contextually relevant dataset. Further details on the specific implementation and application of GPT models in this study will be discussed in subsequent sections.

4 Method

In the method section, we delve into a detailed examination of our subquestions.

Firstly, we attempt to answer Subquestion 1: 'How should we generate our Dutch care texts?' To answer this we discuss different approaches to generate Dutch care texts and their merits and drawbacks.

Secondly we address Subquestion 2: "How can we assess the performance of the models, including the generated Dutch care texts and error introduction methods?" This involves a detailed examination of relevant human and automatic evaluation methods, including an examination conducted on the evaluation of semantic preservation.

Third, to answer Subquestion 3: "How can we determine the most suitable error introduction methods and models for effectively correcting colloquial Dutch care text?" There will be a detailed description of the chosen error introduction methods and models. Finally we will discuss our experimental setup.

Before delving into the subquestions, we will first offer a detailed overview of the available dataset, outlining the data preprocessing steps, and describing the methods we employ to generate a synthetic dataset. All our generated synthetic data is available on Github[2]

4.1 Data

This research has access to a dataset comprising textual reports written by employees of an elderly care organisation, detailing both in-house and external care patients. This data belongs to an anonymous private healthcare provider with facilities in multiple cities in the Netherlands. These records encompass a diverse range of texts, extending beyond employee-patient interactions to include a spectrum of updates, observations, and notes related to patients. The primary purpose of these entries is to facilitate communication among co-workers, ensuring that all relevant team members remain informed about a patient's status. Moreover, these records are also used to offer updates to family members. They are accessible online through the care organization's website, if given access by the patient. In the absence of any formatting guidelines, the records are unstructured. Furthermore, due to the employees' numerous responsibilities, the quantity of notes that need to be written and their often limited proficiency in Dutch, the notes are often sloppy and ungrammatical. There are up to 32 possible fields available per care report entry, however most fields are empty and unused. The fields that are consistently used are the time and date of creation, client ID and employee ID. While there is a specific field dedicated to 'pulse pressure' it is empty even when this information is mentioned in the care report. This indicates how most fields are unused in general. Due to privacy regulations, it is not allowed the share samples from the data, however, to give an impression of the data we will provide a few anonymized examples (Table 2). Note that due to the extreme diversity and lack of guidelines, many different types of reports are also present in the corpus.

[2] https://github.com/skylerf1/paraphraser_code/tree/main/data/synthetic_error_data

Entry #	Anonymized care entry
1	Meneer vanmorgen gedoucht. Alles schoon aangedaan kon geen schonen ondergoed vinden. Kleding die er hing heb ik in de wasmand gedaan. Bed was schoon dit opgemaakt.
2	Meneer op de gang tegen gekomen. Meneer vroeg aan mij of ik zijn trui aan kan trekken omdat hij koffie wilt gaan drinken. Aangegeven dat ik zijn trui een halfuur geleden uit heb getrokken.
3	Middag: O:Mevr heeft vanmiddag 50 cc urine in haar opvangzak. P:Mevr extra drinken aanbieden. Avond: O:Mevr heeft blauwe plekken op de heupen.
4	Meneer liep hart over de gangen heen en telde regelmatig Meneer zag er moe uit Meneer is rond 20.30u naar bed gebracht is echter nog wel wakker om 22.20u.
5	zus was op bezoek vanmiddag even kort gesprek gehad met zus; zus geeft aan dat meneer in principe opgenomen zou worden op 03/04 in Richterburg/afd Dronte (tussentijdseopvang + wenning) zus heeft het idee dat dit nu niet meer doorgaat aangegeven dat meneer hier voor crisis is en niet voor tussentijdse.

Table 2: Example anonimized care entries

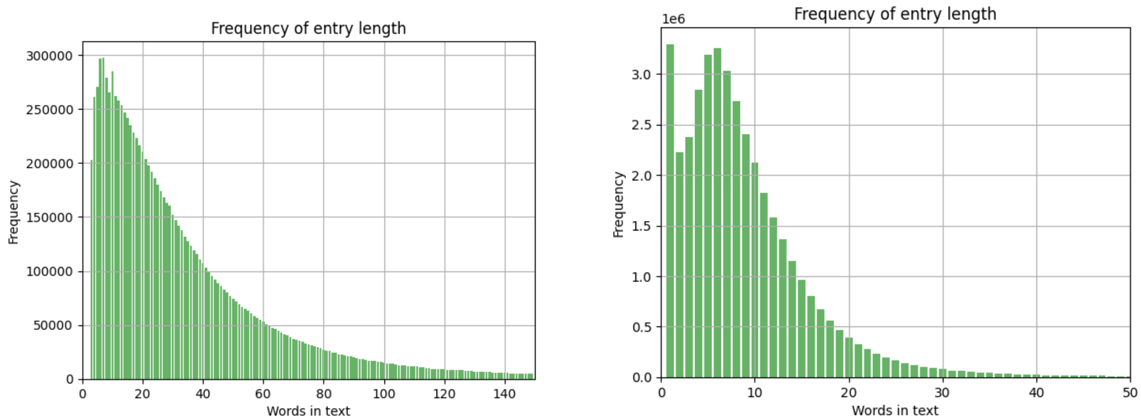
4.2 Data Preprocessing

The initial dataset was provided in a comma-separated values (CSV) file. Due to corruption, certain columns were concatenated with others. To address this issue, we parsed the data, identifying complete entries by checking if the first value in a row was an integer matching the size of an employee ID and the last two values were datetime. This parsing process resulted in 12.865.246 care entries. Subsequently, we applied an abbreviation substitutor to all text, replacing medical abbreviations with their fully written counterpart to enhance comprehension. The abbreviation substitutor was developed based on discussions with a care professional who works with these texts. It included only basic substitutions, lacking advanced and uncommon abbreviations.

Next, all care entries which had less than 3 or more than 150 words were removed. This was done to filter out brief entries, which typically lack substantial information. We excluded entries with more than 150 words since 95% of reports were shorter than 150 words. Longer entries were infrequent and often resulted from data corruption, involving concatenations of different data fields. This left us with 11.458.241 entries. The distribution of word length for these texts is found in Figure 4a. It clearly follows a long-tail distribution with most entries being relatively short.

In response to the client's request for a model operating at the sentence-level, we proceeded to convert the entries into individual sentences, resulting in 45.429.596 sentences. Figure 4b shows the distribution of amount of words per sentence. Some sentences were 150 words long, the entire entry. Texts often are multiple sentences without any interpunction applied to them (Entry 4 & 5 in Table 2). However, the majority of sentences are under ten words, with a median of 7 words per sentence.

A word frequency analysis was conducted on the processed dataset. The occurrences of each word and their frequency were recorded. These word-frequency pairs were stored for subsequent use in a spellchecker. To determine the legitimacy of a token as a word, we established a threshold based on its frequency within the dataset. A token was considered a legitimate word if it appeared at least once in every two and a half million words. Consequently, tokens occurring more than 170 times were deemed real words. This threshold was chosen after a thorough exploration of the data, inspecting the lower levels of the word frequency list, to identify when the majority of tokens stopped being legitimate words. The objective was to incorporate care-related words unique to our topic into the spellchecker wordlist, as they might be overlooked when relying solely on generic Dutch word lists.



(a) Distribution of words per entry (b) Distribution of words per sentence

Figure 4

4.3 Data generation

To train a neural machine translation (NMT) system for correcting Dutch medical text, a substantial amount of training data is required. However, Dutch GEC data is non-existent, posing a challenge. Although we possess a substantial corpus of relevant sentences in need of correction, these texts lack annotations. Regrettably, large-scale manual annotation is not feasible due to its expensive nature. Consequently, to answer Subquestion 1 we explored alternative methods, including Large Language Model (LLM) annotation, LLM-generated incorrect text, and LLM-generated correct text, as potential alternatives.

During our exploration of these models, we encountered challenges with their performance on Dutch text. Stanford's LLM, Alpaca (Taori et al., 2023), and Meta AI's LLM, LLama (Touvron et al., 2023), did not meet the desired level of Dutch language proficiency, as they were unable to correct basic Dutch erroneous sentences and would respond with English text. This limitation prompted us to shift our focus to GPT-3 models Brown et al. (2020), which demonstrated significantly higher capabilities as they were able to correctly enhance diverse Dutch erroneous sentences. Therefore they were subsequently selected for further exploration in our study. Due to budget constraints, we selected GPT-3.5 Turbo, OpenAI's most cost-effective model. Despite its affordability, the model demonstrated high capability, as evidenced by its use in the popular Chat-GPT application.

4.3.1 LLM annotation

Initially, we considered leveraging state-of-the-art LLMs to annotate our textual care reports, due to their impressive linguistic capabilities. The aim was to generate a dataset with accurate annotations for a fraction of the cost. Zero-shot instructions were given to the GPT-3.5 model in Dutch: Verbeter nederlandse medische tekst. Als de input verbeterd kan worden, geef de verbeterde versie van de tekst en niets anders. Als de input niet verbeterd kan worden, reageer met de input tekst. Als de input onbegrijpelijk is reageer dan met de input tekst. Geef geen extra informatie.' Translated to English this means: 'Improve Dutch medical text. If the input can be improved, provide the improved version of the text and nothing else. If the input cannot be improved, respond with the input text. If the input is unclear, respond with the input text. Do not provide additional information.' Refer to Appendix B for a complete list of all further used prompts. The instructions were in Dutch because, if given in English, the model would occasionally correct text in English. During experimentation we used the Dutch section of the lang-8 dataset (Tajiri et al., 2012). Despite occasional misunderstandings, this prompt and model seemed to perform effectively in labeling erroneous texts and replicating correct ones during experimentation (See Table 3). However, given that OpenAI retains records of input data, which is a clear violation of privacy laws surrounding personal medical information, we ultimately decided that automatic annotation of the corpus was no longer a viable option for this research.

4.3.2 LLM-generated incorrect text

Since we were unable to directly label our erroneous text, we opted to generate erroneous medical data using GPT models, thereby overcoming the privacy issues. These generated erroneous texts would then be annotated using GPT to create our training dataset. Recognizing the need for more context from the previous approach, we refined our approach to provide the model with a clearer understanding of the type of text we aimed to generate. This was attained by performing few-shot learning, including two examples of realistic entries in the prompt. We created 150 similar anonymized texts based on the real data. We revised the prompt by including two randomly sampled distinct texts as examples each time the model generated new texts. This resulted in texts which were more similar in context to the real data. However, the generated errors were not as diverse as in our actual corpus. Most errors were spelling errors and problematic sentence structure (See Table 3 for examples) Furthermore, the model would sometimes refuse to generate incorrect texts. Therefore we decided to stop directly generating the errors using the GPT model, as the model excelled at generating correct text.

GPT-generated incorrect care entry	GPT-generated annotation
De clint heft las van zijn bene. Hij klacht dat hij niks meer kan. Ik heef hem medicijne gegeven en hem geholpen met het aandoen van zijn schoene. Hij had ze verkeert om gedaan en het was moeilijk om ze goed te krijgen. Hij heef gezegd dat hij pijn heeft in zijn hofd, dus ik heb hem paracetamol gegeven. We hebben saamen kofie gedronken en gepraat over het weer. De clint is erg dankbaar dat ik er ben om hem te helpen.	De cliënt heeft last van zijn benen. Hij klaagt dat hij niets meer kan. Ik heb hem medicijnen gegeven en geholpen met het aandoen van zijn schoenen. Hij had ze verkeerd om gedaan en het was moeilijk om ze goed te krijgen. Hij heeft gezegd dat hij hoofdpijn heeft, dus ik heb hem paracetamol gegeven. We hebben samen koffie gedronken en gepraat over het weer. De cliënt is erg dankbaar dat ik er ben om hem te helpen.
Meervrouw voeld zich soms heel erg duizelig. Zij klaagd ook van hartkloppingen. Ik heb daarvoor een elektrocardiogram gemaakt. Het lijk normaal te zijn. Meervrouw wilt graag medicijnen. Ik heb uitgelegd dat ze de medicijnen regelmatig moet innemen. Maar zij zegd dat ze de medicijnen altijd op hetzelfde tijdstip inneemt. Ik heb haar uitgelegd dat dit niet goed is. Ze was heel geirriteerd en wilt dat ik de medicijnen voorschrijf zonder te kijken naar haar medische conditie.	De vrouw voelt zich soms duizelig en klaagt over hartkloppingen. Ik heb een elektrocardiogram gemaakt, maar deze lijkt normaal te zijn. De vrouw wil graag medicijnen, maar ik heb uitgelegd dat ze deze regelmatig moet innemen. Ze zegt dat ze de medicijnen altijd op hetzelfde tijdstip inneemt, maar ik heb haar uitgelegd dat dit niet goed is. Ze was geïrriteerd en wilde dat ik medicijnen voorschrijf zonder naar haar medische conditie te kijken.
Gisteren is meneer opstaan geholpen door de verpleger. Hij heb niet zo lekker geslapen en heeft geklaagd van een zore rug. We hebben de dokter gebeld om de medicijnen te wisselen, en de afspraak zal plaatsvinden in 3 weken. De toestemmingsverklaring heb ik onderteken en in de documenten geplaast. Hopelijk voelt meneer zich beter na de medicijn wisseling.	Gisteren is meneer geholpen met opstaan door de verpleger. Hij heeft niet zo lekker geslapen en heeft geklaagd over een zere rug. We hebben de dokter gebeld om de medicijnen te veranderen en de afspraak zal over 3 weken plaatsvinden. Ik heb de toestemmingsverklaring ondertekend en in de documenten geplaatst. Hopelijk voelt meneer zich beter na de medicijnwisseling.

Table 3: Example of LLM-generated incorrect texts & LLM-generated annotations

4.3.3 LLM-generated correct text

To create synthetic erroneous data, people typically use a large corpus of grammatically sound texts and introduce errors. However, aligning the content of this corpus with the specific type of text that needs correction can be challenging. In our case, focusing on Dutch medical text correction, an ideal scenario would involve a large corpus of accurately written Dutch medical texts in our writing style. To address the lack of access to such data, we turned to GPT for a solution. We opted to generate accurate medical texts, similar to our corpus, that we would later intentionally corrupt to create our erroneous and target texts. The corruption process is discussed further in the error introduction section. This data generation process aims to produce a wide variety of accurately written texts, capturing the diversity found in our medical corpus. This approach is rooted in the idea that a model trained on contextually similar data will outperform a model trained on off topic data, as Felice's research (Felice and Yuan, 2014) suggests.

Therefore, we shifted our focus to generating correct care entries. These texts could be used as our relevant corpus. We instructed the GPT model to emulate a Dutch medical care professional who communicates in plain language and provides daily or partial-day updates on the care situation of elderly individuals. This led to a varied array of descriptions depicting relevant situations. However, we observed that the generated data did not describe similar events as our actual dataset. To address this, we applied few-shot learning. We used the same 150 examples of medical entries described in the previous section. These were based on random samples from the real corpus, however no sensitive information was present in them. We then prompted the GPT model with the same instructions, along with two distinct randomly selected examples. We chose for two examples over one because different combinations of examples could result in different outcomes. Additionally, more than two examples would likely result in even better text. However, the more examples used, the more is paid per generated sentence since more tokens are used. It was our reasoning that the model would be able to generate more diverse texts with more examples. This was supported by the decrease in the amount of duplicate sentences. While zero-shot learning resulted in 16% of the sentences being duplicates, few-shot learning resulted in 6% duplicated sentences. Based on a qualitative assessment, it appears that the model tends to generate texts with a higher similarity to our real corpus compared to the other approaches.

Later, we incorporated a Named Entity Recognition (NER) model, provided by a colleague at PrimeVision working on the same dataset (Clemente, 2023). This NER model, tailored for this dataset, was specifically trained to identify entities such as persons, locations, diseases, medications, statuses, body parts and dosages. Applying the NER model to the care reports corpus produced word-NER tag combinations. Utilizing the identified words with NER tags (excluding person and location for confidentiality reasons) as keywords, we prompted the GPT model. These keywords, combined with hand-written examples, guided the GPT model to generate text mimicking a care professional (Full prompt in Appendix B). Each prompt involved three keywords and two example entries, contributing to the realistic nature of the generated data. This strategy should enhance the diversity of the generated text, and as a result, the scalability of the generation process was improved. This was supported by a further decrease in the occurrences of duplicate sentences to 1%.

To compare error introduction methods and models, we needed a clean care entries corpus. Therefore, we used the above method to generate enough care entries for 100.000 sentences (See Table 4 for some example sentences). We decided to generate five care entries per prompt, we did this because we wanted to experiment whether these texts would be significantly different and it was very cost effective. Subsequently, we split these texts into sentences. The generated entries consisted of approximately 4.7 sentences per entry on average. Finally, we checked for duplicate which we removed. We deemed 1% of duplicate sentences low enough to select this approach as our answer to Subquestion 1: 'How should we generate our Dutch care texts?'

4.4 Error Introduction

With our newly generated corpus of Dutch care sentences in hand, the next step was to corrupt the correct sentences by introducing errors, aiming to align it more closely with our actual corpus of care entries. Examples of the corruption methods used are found in Figure 5 and 6.

We base our methods on error introduction methods for low resource scenarios from Náplava and Straka (2019), Grundkiewicz et al. (2019), Choe et al. (2019) and Rothe et al. (2021). We have devised four distinct pipelines for introducing errors to our corpus of originally correct care entries. All of these versions involve word replacement through a spell-checker, and we will begin by elaborating on this process. Afterwards we will discuss the different error introduction versions and how we apply them.

4.4.1 Spell-checker replacement

We intend to apply the error introduction method proposed by Náplava and Straka (2019) to our generated texts. This method involves applying replacement, deletion, swapping, insertion, and recasing operations to words and characters. These operations are relatively straightforward; however, the replacement of words necessitates having words for substitution. In the original approach by Náplava and Straka (2019), words selected for replacement were substituted with English spell-checker suggestions from ASpell.

To enhance this method, we leverage the extensive corpus of errorful care entries at our disposal (see section 4.1). This was inspired by Choe et al. (2019) to combine spell-checker replacement with error pattern introduction. This decision arises from the observation that the Dutch ASpell vocabulary is less advanced than its English counterpart, and the basic Dutch ASpell vocabulary lacks Dutch medical terms found in our corpus. Consequently, we replace words with errors identified by a Levenshtein distance spell-checker within the care entries for those words. The spell-checker is supported with a Dutch wordlist from Opentaal (Simon Brouwer, 2023), enriched with words which occurred over 3 times in the Dutch NOS news dataset (Scheijen, 2022). Additionally, we include words occurring at least once every 2.5 million words in our corpus of care entries. We implemented this criterion because we noticed that, beyond this frequency threshold, words exhibited a higher tendency to no longer be legitimate words. This combined Dutch word list aimed to capture correct Dutch words and medical terms not present in both Dutch news and OpenTaal words. Following a spell-checker analysis on the complete care corpus, 95% of the words generated by GPT-3 were discovered to have at least one associated corrected error, if a word was selected which did not have an associated error, it was left alone.

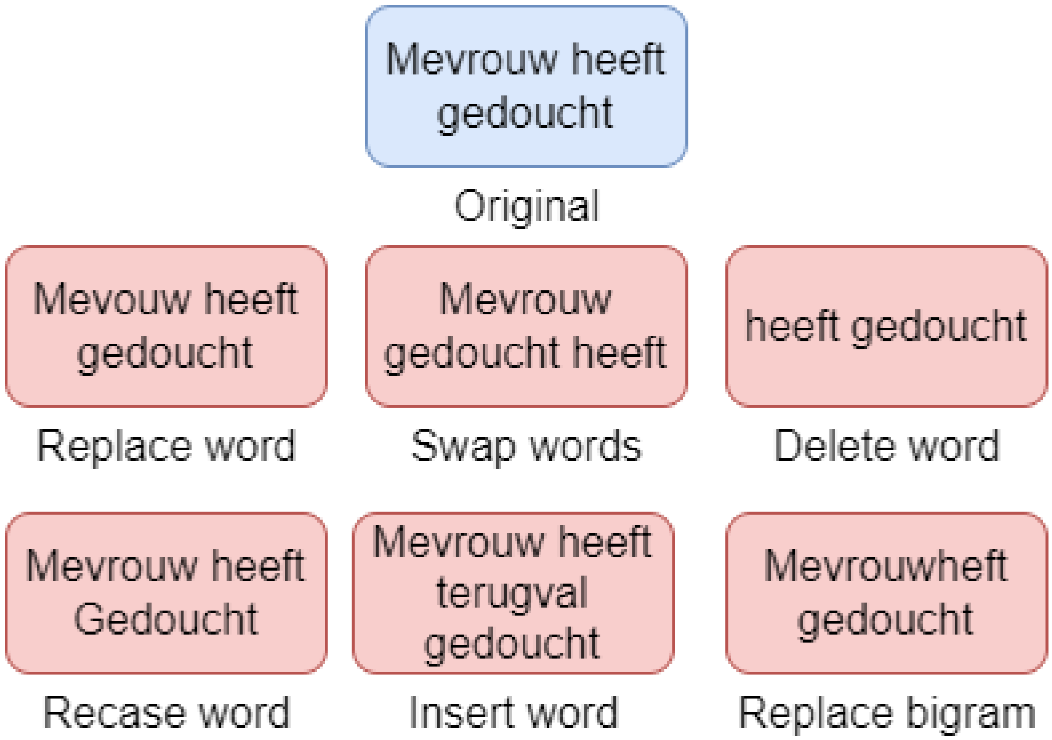


Figure 5: Examples of text corruption methods used in our error-introduction versions

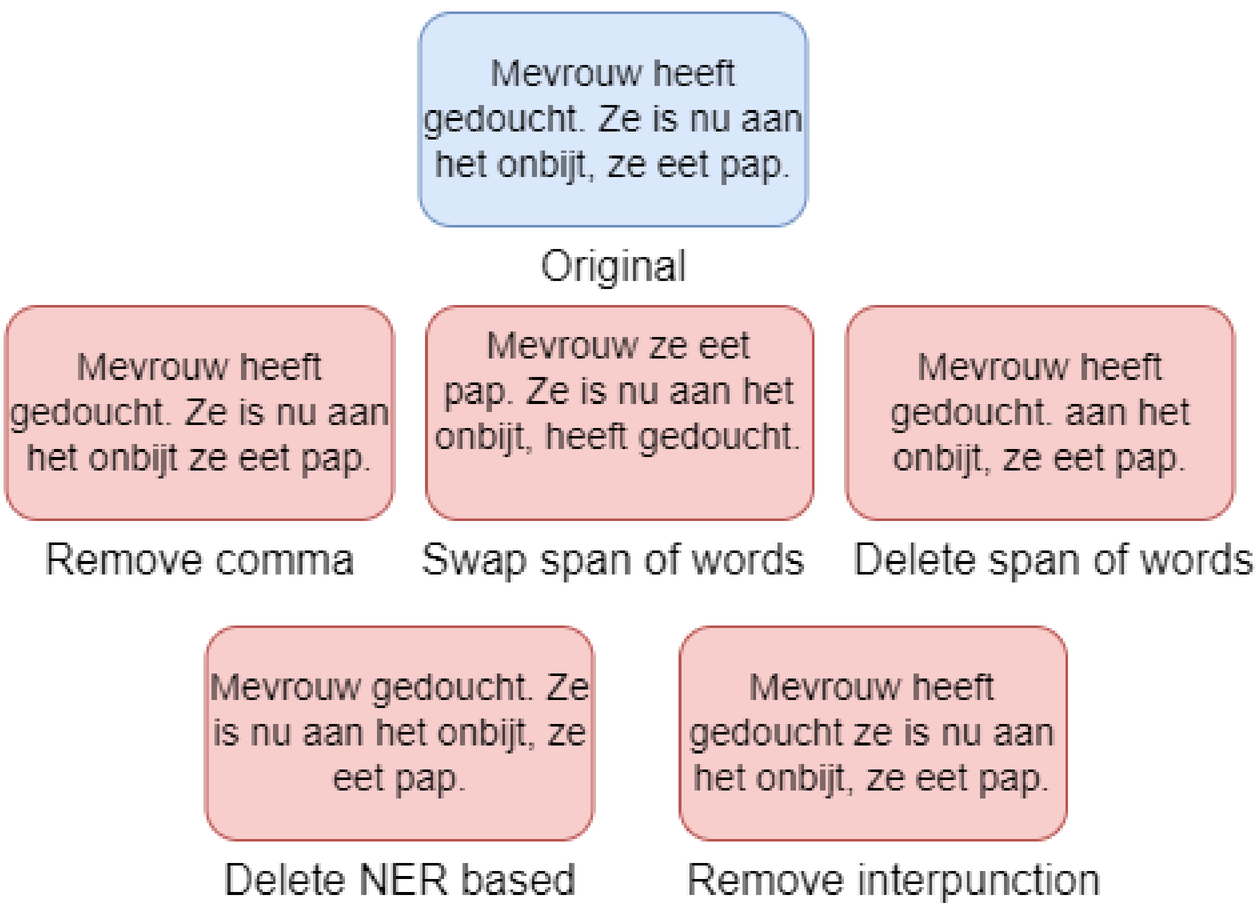


Figure 6: Examples of text corruption methods used in our error-introduction versions

4.4.2 Version 1 - Baseline

We implement the error introduction method proposed by Náplava and Straka (2019) in our generated texts. This method was originally designed for general low-resource synthetic data generation in the context of GEC based on Grundkiewicz et al. (2019). In the following error introduction versions we aim to extend on this method based on patterns found in our corpus of care entries and research. This will be a baseline with general error introduction for low-resource languages.

The method involves applying replacement, deletion, swapping, insertion, and recasing operations to words and characters. Each text undergoes an iterative corruption process. When a text is selected, it is split into words. While iteration over the words, each word has a 15% chance of being corrupted. If a word is chosen for corruption, there is a 70% chance for it to be replaced, 10% chance to be deleted, 10% chance to be swapped with its right neighbor, 5% chance for a random word from the vocabulary to be inserted on either side, and a 5% chance for the word to be recased. Replacement is executed following the method detailed earlier in the spellchecker paragraph. After the word corruption process is completed the corrupted text goes through further character corruption. Every character has a 0.5% chance to be corrupted. If a character is selected there is a 25% chance of either deletion of the character, replacement with a random other character, insertion of a random character to either side or swapping with right neighbor character (Figure 7).

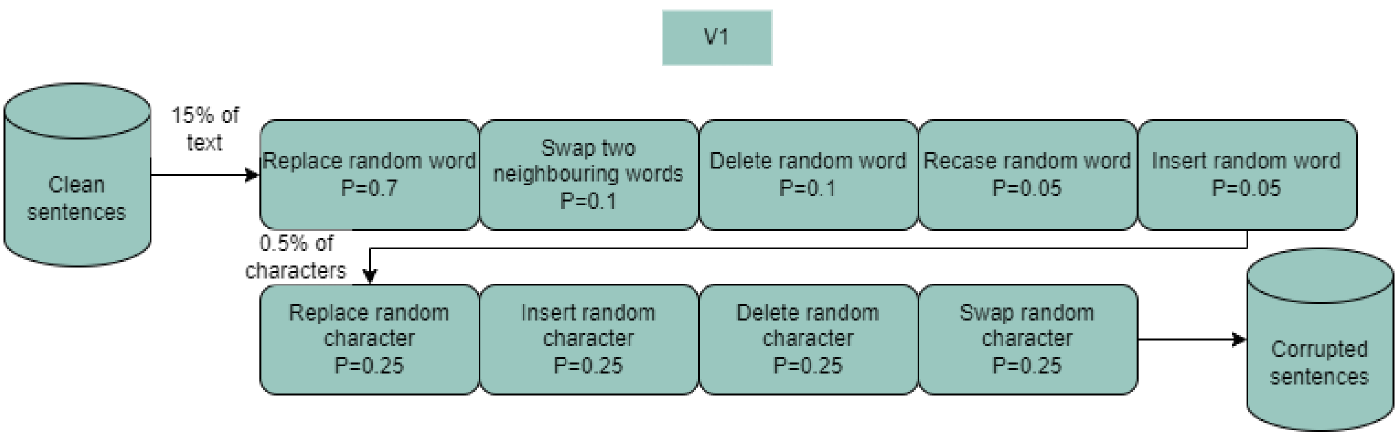


Figure 7: Version 1 - Baseline error introduction

4.4.3 Version 2 - Interpunction error & replace bigram

This error introduction method extends on the method of version 1. It involves introducing punctuation errors and replacing bigrams with identified errors (Figure 8).

Given the rushed nature of care entries, we noticed a lack of punctuation during data exploration. To address this issue, our aim is to train the model to incorporate punctuation into these texts. Our approach involves introducing a new base text, onto which we apply version 1 errors. This base text is constructed from two consecutive correct sentences extracted from our generated texts. Subsequently, we deliberately remove punctuation marks from the text. Additionally, we introduce a 50% probability of removing the capitalization of the word following the punctuation mark. This is done because entries often capitalize words to indicate the start of a new sentence but do not consistently employ punctuation. Furthermore, comma's are removed 30% of the time for all texts to teach the model to introduce them too.

We have also incorporated bigram replacement into our approach. After spell-checking care entries, we identified many errors where two words were erroneously combined and required separation to be corrected. Given the prevalence of these errors in our dataset, we sought to introduce them into the synthetic data through bigram replacement. In this process, we introduced the possibility of selecting a word for bigram replacement. If chosen, we examine the selected word and its right-side neighbor to check for a bigram error. If such an error is present in our error dictionary, both words are replaced with a random choice from the relevant errors. If not, we attempt to replace the unigram as in normal word replacement. To use this method, we adjusted the probabilities of text corruption, we redistributed the original 70% chance of unigram replacement to a 40% chance of attempting bigram replacement and a 30% chance of unigram replacement.

Furthermore, we incorporate a 2% probability for a sentence to remain unaltered. This addition is inspired by the work of Rothe et al. (2021), aiming to teach models that text can be correct without modification, which is true for the care entries.

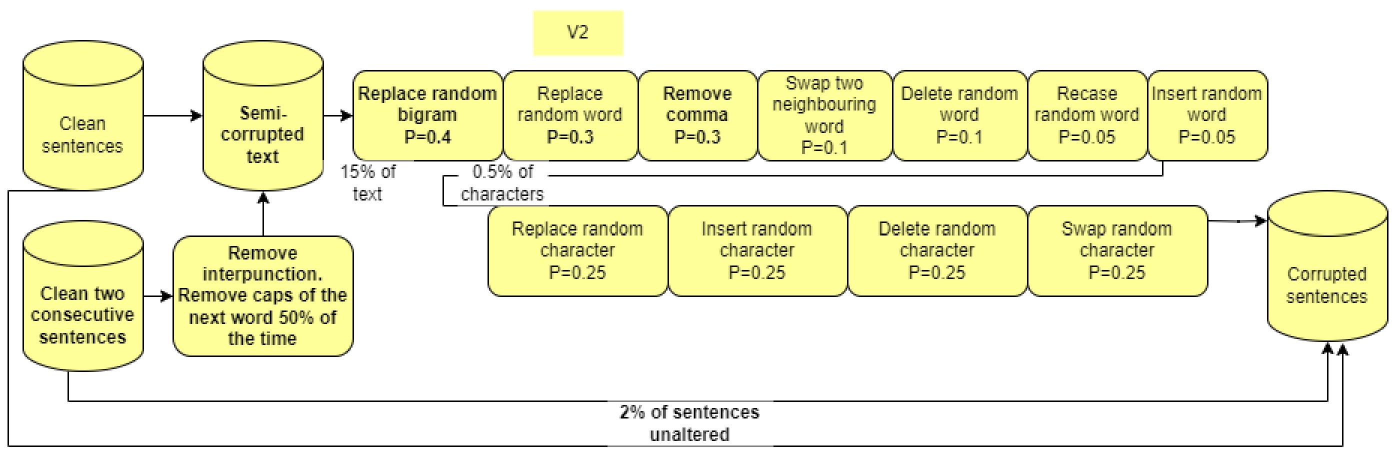


Figure 8: Version 2 - Interpunction error & replace bigram

4.4.4 Version 3 - Span corruption

The third error introduction method is an extension of the second method. It draws inspiration from the span corruption method applied by Rothe et al. (2021). Their method involves removing and swapping spans of tokens for general GEC, with good results. This was further supported by identified issues during data exploration, where many instances exhibited sequences of missing words or were arranged in the wrong order.

To reproduce these type of texts, we expand our error introduction by incorporating span corruption into a sentence. When a text is selected for span corruption, there is an equal likelihood of either span deletion or span swapping being applied. We made this choice to avoid sentences being affected by both, as it would excessively distort the sentence structure. Span deletion should simulate situations where spans of words are missing from care entries. Span swapping should simulate situations where text is hastily written and in the incorrect order. Both span operations have ranges for the length of sequences to be altered. In the case of deletion, a minimum of 2 to a maximum of 4 words were deleted. For span swapping, sequences ranging from 1 to 4 were exchanged with another span of 1 to 4 words. If the selected numbers did not align with the sentence length, they were chosen again until the numbers aligned.

Additionally, we adjust the probabilities for regular word deletion and swap deletion. The deletion probability is reduced from 10% to 5%, as we expect removing spans of words in combination with a lot of word deletion can result in substantial information loss. Simultaneously, the swapping probability is increased from 10% to 15%. This will allow the model to encounter swapped spans in conjunction with swapped words, hopefully facilitating the correction of jumbled texts.

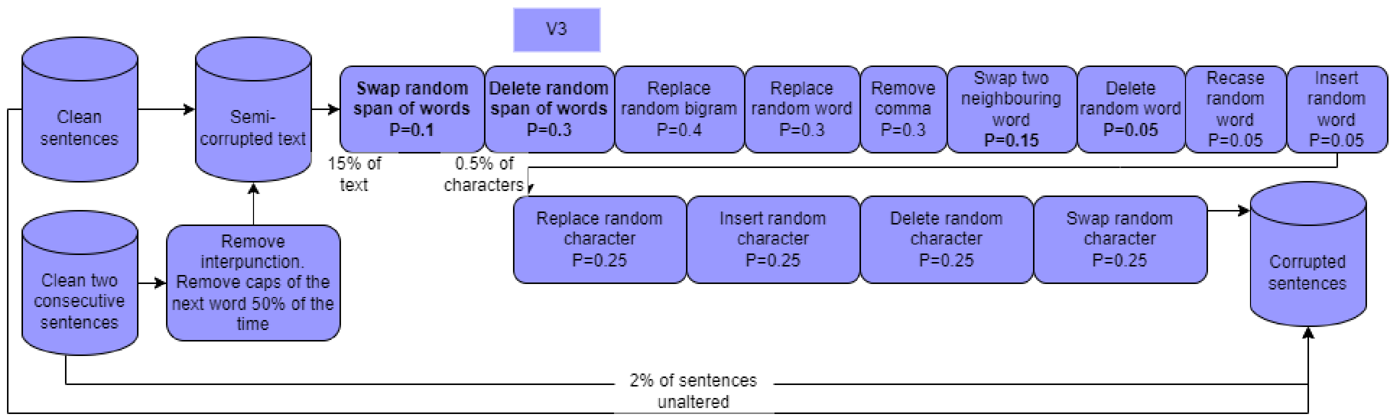


Figure 9: Version 3 - Span corruption

4.4.5 Version 4 - Syntactic dependency based deletion

Finally, we extend error introduction version three with syntactic parsing to delete words in a smart manner, replacing span deletion and word deletion entirely while keeping span swapping in place. Choe et al. (2019) utilized smart POS noising to improve performance, we aim for this method to remove words in a more realistic manner than random word deletion, thereby creating more realistic erroneous text.

When exploring the data, we applied spaCy's natural language processing module (Honnibal and Montani, 2017) to both hand-corrected and erroneous care entries. We checked the syntactic labels that were present in correct text and missing from incorrect text by comparing their respective counts. This gave us insight in what kind of words were often missing from erroneous sentences. Armed with this information, we delete crucial types of words from correct sentences to emulate the errors prevalent in the care entries corpus.

The syntactic dependency labels significantly more present in correct text, in descending order, were: auxiliary verbs, nominal subjects, root of the sentence, determiner, coordinating conjunction, nominal subject (passive), auxiliary verb (passive), copula (verb linking the subject to the subject complement) and case marker.

During text corruption, we apply spaCy's syntactic labelling module, selectively removing each of the above-mentioned word types with a probability of 0.375%. In previous error introduction versions, the odds of word deletion were around 1.5% per word, and we aimed to adhere to this by scaling the syntactic deletion probability with how often the chosen labels were present proportionally.

To maintain consistency with prior error introduction versions and accommodate the removal of the 10% chance of regular word deletion, we adjusted the scaling accordingly. This was achieved by increasing replace bigram, replace unigram, insert word, swap word and recase word by 10% in combination with lowering the original 15% probability to 13.5% to maintain a balanced distribution in the overall error introduction process. This adjustment ensures that the combined odds return to the required total of 1.

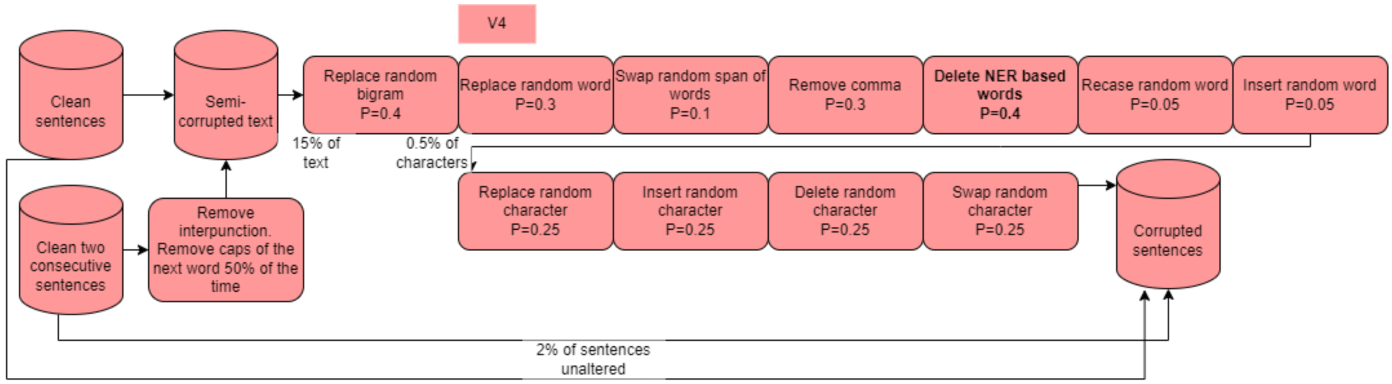


Figure 10: Version 4 - NER deletion

Version	Original	Corrupted
Version 1	Mevrouw heeft ook aangegeven dat ze meer haarverlies ervaart dan normaal.	Mevrouw ook heeft aangegeven dat zj meer haarverlies ervaart dan normaal.
Version 2	Ze geeft aan dat haar buik erg rommelt en dat ze meerdere malen naar het toilet is geweest.	Ze geeft aan dat haar buik werg rommelt koMt en dat ze meerdere naar het toilet is geweest.
Version 3	Incident: Vandaag vond er een incident plaats waarbij mevrouw tijdens de zorg voor haar ogen transpireerde. Het is belangrijk om hier extra aandacht aan te besteden en ervoor te zorgen dat haar ogen goed schoon en droog blijven.	Incident: Vandaag vond er een incidet plaats waarbij tijdens mevrouw zorg haar Voor ogen transpireerde Het is belangrijk om hier extra aandacht aan te besteden en ervoor te zorgen dathaar goed schoon en droog blijven.
Version 4	Daarnaast is haar po-gebruik verhoogd om haar stoelgang regelmatig te houden en obstipatie te voorkomen. We houden nauwlettend in de gaten of de klachten verminderen en zullen de kuur indien nodig aanpassen.	Daarnaast haar po-gebruik verhoogd om haar stoelgang regelmatig te houden obstipatie te Voorkomen we houden Nauwlettend in de gaten of de klachten verminderen en zullen de kuur indien nodig aanpsasen.

Table 4: Example of a generated text and a corrupted text for each error introduction version

4.5 Models

The primary models of interest for GEC in our study are the seq2seq "UL2" transformer model and the "Multilingual BART" (mBART) transformer model (Liu et al., 2020; Tay et al., 2022). State-of-the-art systems (Rothe et al., 2021; Sun and Wang, 2022) are built upon these type of models because of their pre-training with denoising objectives, a feature that closely aligns with the goals of GEC. By utilizing pre-trained models, we can avoid the need to teach language to a model and leverage its embedded linguistic knowledge. Additionally, they both have available pre-trained models on Dutch language. We steered clear of BERT models, despite the success of sequence-tagging models by Omelianchuk et al. (2020). While this approach is interesting due to the ten-fold decrease in inference speed, their BERT (Devlin et al., 2018), RoBERTa (Liu et al., 2019) and XLNet (Yang et al., 2019) ensamble approach is outperformed by the singular seq2seq models on the CoNNL-14 (Ng et al., 2014) and BEA-2019 (Bryant et al., 2019) benchmarks.

4.6 Evaluation

To evaluate models during training, 283 sentences from the care entry corpus were hand-annotated by the author, as references are required for reference-based evaluation metrics. The sentences were extracted from 100 randomly selected care entries. The references were hand-annotated using the minimal corrections guidelines by the author. While increasing the number of annotated sentences could enhance the evaluation's robustness, practical considerations led to the decision to limit the dataset to 100 care entries. This choice was motivated by the fact that scoring numerous model candidates for each annotated sentence during human evaluation would be a resource-intensive task.

A challenging aspect of GEC is the inherent variability in correcting incorrect text. Multiple corrections are possible, and determining the optimal correction is subjective, relying on individual perceptions. Nevertheless, as argued by Choshen and Abend (2018b), preparing reference sentences that encompass all conceivable corrections is an unrealistic task and we stick to a singular reference text. The aim is to identify an automated metric which aligns with human judgments, thereby eliminating the need for human judgments when improving error-introduction methods.

4.6.1 Evaluation metrics

For model evaluation during training, we opted for GLEU+ (Napoles et al., 2016a) as the metric of choice. This decision was driven by its simplicity, requiring only a source, candidate, and reference text for computation. Furthermore, GLEU+ was specifically designed with GEC in mind, utilizing the source, hypothesis and reference sentence while generic metrics like BLEU (Papineni et al., 2002) and ROUGE (Lin, 2004) do not. We considered the METEOR (Banerjee and Lavie, 2005) metric however the lack of a Dutch dictionary for synonymn checking made it less powerful than GLEU+. Several lesser known GEC specific evaluation metrics were excluded from further consideration for various reasons: incompatibility with Dutch (GoToScorer (Gotou et al., 2020), Scribendi Score (Islam and Magnani, 2021), PT-M^2 (Gong et al., 2022)), baseline GEC model requirement (GFM (Asano et al., 2017)), specific hypothesis annotating required to use (SOME, (Yoshimura et al., 2020), Usim (Choshen and Abend, 2018b)), extensive ensamble methods (GMEG (Napoles et al., 2019)). While widely used metrics like M^2 and ERRANT are effective, they require human-annotated edit spans, which can be unpredictable and time-consuming to collect. Moreover, extending these metrics to Dutch would be a resource-intensive task, outside the scope of this research. Finally, GLEU+ was selected as it aligned with human judgment in our initial experiments.

After completing the training phase, we assess the semantic preservation of information in the generated hypothesis sentences by employing BERTScore and a SBERT model along with cosine similarity in comparison to the target sentences produced by the models. These scoring methods measures help identify when a model alters the semantics of a text. This approach serves as a valuable metric for evaluating the efficacy of the models in maintaining semantic consistency.

For BERTScore (Zhang et al., 2019a), we employ the default BERT model for Dutch sentence similarity[3] as is recommended by the developers. Although it is possible to use a pre-trained Dutch monolingual BERT model, which would reasonably perform better than a multilingual model, we opted for the default multilingual version because the optimal model layer to use for this model was already found. To use a pre-trained Dutch monolingual BERT model would require additional tuning to determine the optimal layer for sentence similarity which the authors do based on the WMT16 metric evaluation dataset (Bojar et al., 2016). This is problematic since this dataset does not contain Dutch translations and can therefore not be used for a monolingual Dutch model.

For SBERT, we utilize a sentence-transformer model[4] (Reimers and Gurevych, 2019). This model is a fine-tuned version of a monolingual Dutch RoBERTa model[5] (Liu et al., 2019). The model is fine-tuned using Dutch frequently asked question-answer pairs from the CLIPS Multilingual corpus of Questions and Answers (MQA) (De Bruyn et al., 2021). This model provided the sentence embeddings which were used in combination with cosine similarity to calculate the sentence similarity.

4.6.2 Human evaluation

Robust evaluation is still an unsolved problem in GEC. The comparison of metrics in GEC poses challenges due to the subjectivity of human judgments. Human evaluations, while intuitive, are subjective and caution is advised. However, given that there is no consistently reliable automated metric and humans will ultimately utilize the model-generated sentences, we considered it crucial to depend on human judgments to assess the alignment between the chosen evaluation metric and human assessments.

Therefore, we decided to evaluate the performance of models by humans along with the automated GLEU+ metric. This involved scoring the performance of different models using the same set of 283 sentences that were previously used for evaluation with the GLEU+ metric during training. We instructed the evaluators with the instructions found in Appendix A. This included instructions about judging sentences on semantic consistency, fluency, grammatical correctness and spelling. We also instructed them to consider multiple interpretations of the available context when judging model corrections. Hypothesis sentences generated by the models were assessed by humans and assigned a score on a scale of 'significant deterioration, slight deterioratation, no deterioration/improvement, slight improvement, significant improvement. Two human evaluators were presented with the original sentence along with eleven model hypotheses, in randomized order to prevent bias. They then ranked these hypotheses within the provided range of options. The rankings will subsequently be rescaled to a numerical scale of 1 to 5 for computation. Using these scores the Cohen's Kappa between evaluators will be calculated. We will use dynamic weighting for Cohen's kappa to account for the fact that the scores are in an ordinal range.

[3]—bert-base-multilingual-cased model from Huggingface.

[4]jegorkitskerkin/robbert-v2-dutch-base-mqa-finetuned model from Huggingface.

[5]pdelobelle/robbert-v2-dutch-base model from Huggingface.

4.7 Experimental setup

The best method for evaluating synthetic data quality remains difficult to answer, with no solution for the most effective approach (Htut and Tetreault, 2019; White and Rozovskaya, 2020). Kiyono et al. (2019) attempt to compare noise injection and back-translation. However they find it is difficult to compare error introduction methods directly. Therefore most research indirectly evaluates the quality of synthetic data by examining its impact on the performance of experiments. Therefore we will experiment with different combinations of error introduction datasets and models (See Figure 11). We then compare the performance of these models using GLEU+, human evaluation and semantic similarity scores.

We create four training datasets using the four different error introduction methods, these will be used to compare their quality. To ensure a fair comparison between approaches, each method should introduce a similar number of errors based on the specified probabilities in their respective paragraphs. Additionally, we use a consistent set of around 100.000 GPT-generated sentences to be corrupted, resulting in four distinct training datasets of erroneous and reference sentences based on the same clean texts. However, for versions 2, 3, and 4, we randomly selected 25% of the sentences to conform to the format required for punctuation error introduction, as outlined in error introduction version 2. After introducing punctuation errors, these sentences were combined with the remaining 75%, and errors were applied to them following the same procedure as in version 1, with the incorporation of the features unique to each error introduction version. Additionally, 2% of all texts were intentionally left unaltered before corruption.

The UL2 model[6] used was pre-trained through self-supervised learning on the Dutch text from the mC4 cleaned corpus (Raffel et al., 2019). The data has been cleaned by removing sentences containing: Less than 3 words, a word longer than 250 characters, a symbol outside of the end-of-sentence punctuation or text associated to javascript code (e.g. {}), lorem ipsum, policy information in Dutch or English. After sentence filtering they also reject all documents with less than 5 sentences, less than 500 or more than 50.000 characters or documents not identified as mainly Dutch by the LangDetect package.

The mBART model[7] is pre-trained as described in Tang et al. (2020) using the BART objective for multiple languages at once using a source language token and a target language token. For GEC purposes we translate from Dutch to Dutch and use the Dutch language token for both the source as the target token.

We fine-tune our models on batches of 16 source and target texts for 10 epochs with a learning rate of 2e-5. Throughout the training process, the model performance was evaluated using our hand-annotated evaluation data of 283 sentences. The GLEU+ (Napoles et al., 2016a) score for each input-candidate-reference combination was calculated, and the average was returned to judge the quality of each model iteration.

The aim of this experiment is to identify effective error-introduction versions and models for Dutch care GEC. Additionally, the goal is to identify models that best preserve semantic information. Finally, the alignment of GLEU+ scores with human judgments would be valuable for future iterative enhancements of the error introduction versions. This would eliminate the need for time-consuming human evaluation.

[6]'yhavinga-UL2-large-Dutch' model from Huggingface.

[7]facebook/mbart-large-50 model from Huggingface.

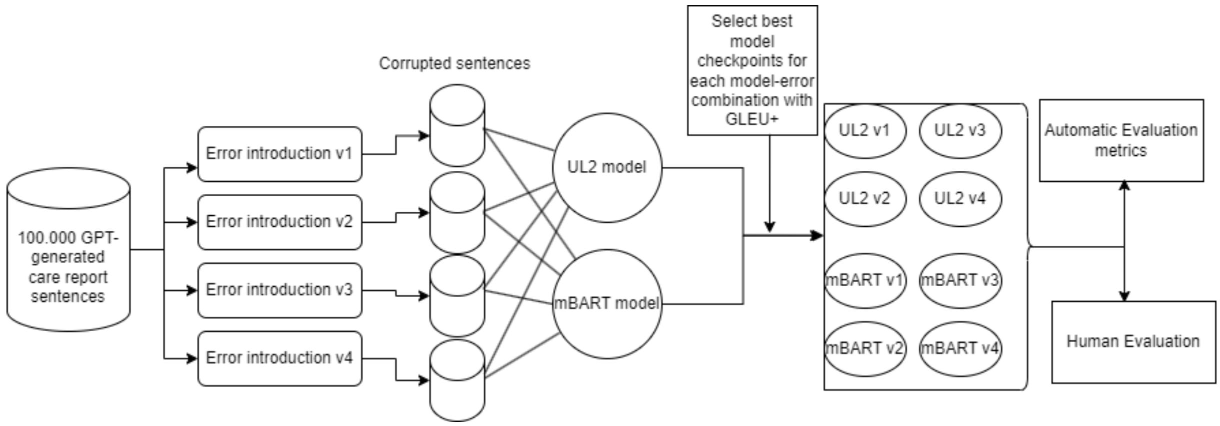


Figure 11: Experimental setup, visualising data generation, error introduction, model selection and evaluation

5 Results

In this section, we address subquestion 3: 'How can we determine the most suitable error introduction methods and models for effectively correcting colloquial Dutch care text?' To answer this question, we discuss the training process of the models, their GLEU+ scores on the evaluation set, and the corresponding human evaluation results, as outlined in the experimental setup section. We then present the outcomes of our experiments aimed at identifying the most effective model and error introduction method for correcting colloquial Dutch care sentences. To evaluate the preservation of semantic meaning, we delve into the semantic similarity scores of BERTScore and Sentence Similarity between hypothesis sentences generated by the models and the hand-written target sentences of the evaluation set.

5.1 Training

To pinpoint the optimal models, we assessed their performance throughout the training process, utilizing the evaluation set and the GLEU+ metric. Figure 12 displays the results of training the 8 models—four UL2 and four mBART models, each with four different error introduction versions. It's important to note that GLEU+ scores fall within the range of 0 to 1, and a higher score indicates better performance.

The GLEU+ scores indicate that the mBART model along with error introduction version four performed best. Furthermore all mBART models outperform their UL2 error-introduction version counterpart according to GLEU+. Both UL2 version 1 and mBART version 1 performed worst compared to the other UL2 and mBART version combinations respectively, indicating that error introduction version 1 performs worst for both model types, according to in-training GLEU+ scores.

Based on the obtained results on the evaluation set, we opted for the models with their respective highest GLEU+ score for human evaluation. Notably, some models demonstrated peak performance at a relatively low number of training steps (UL2 v3, mBART v3, mBART v4). To ensure a comprehensive assessment, we extended our evaluation to include models trained for a longer duration—referred to as UL2 v3 with extended training, mBART v3 with extended training, and mBART v4 with extended training. Despite their comparatively lower GLEU+ scores, we sought to evaluate models that underwent longer training periods. Furthermore, this provides an opportunity to compare the judgement of GLEU+ and humans.

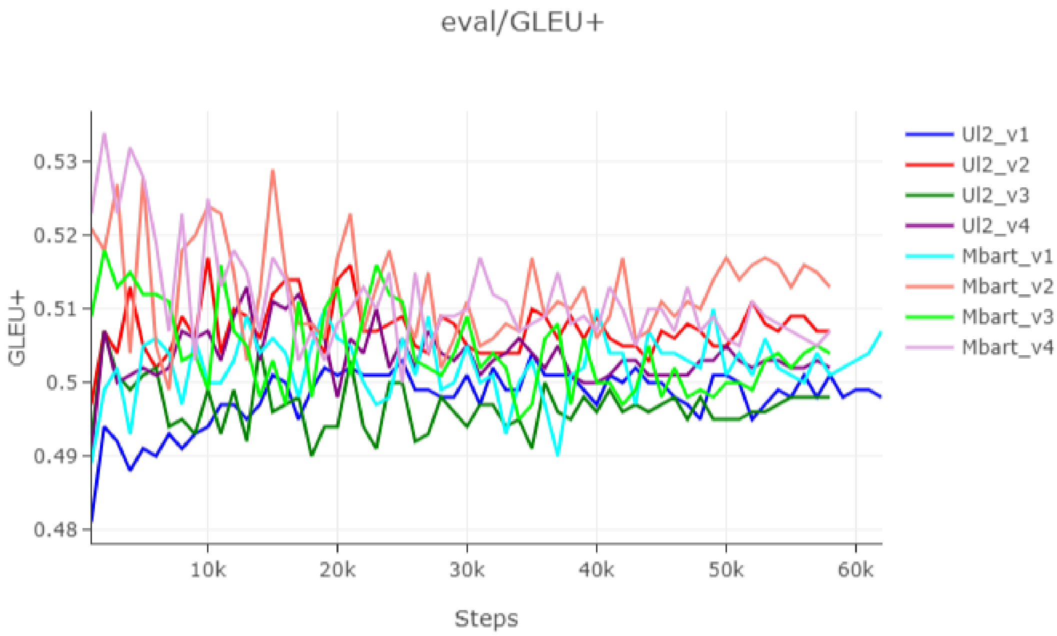


Figure 12: GLEU+ scores during training on the evaluation set measured every 1000 steps

5.2 Human Evaluation

In Figure 13 we visualize the outcomes of the human evaluation. We present a horizontal bar chart containing the average score per model for sentences in our evaluation set. The sentences that remained unchanged across all eleven models were kept out of this visualization. These sentences, denoted by a score of 3 in the evaluation metrics for all 11 models, were intentionally excluded. Given that these sentences exhibited no improvement or deterioration, their inclusion could introduce a bias, potentially misrepresenting the overall performance of the models. By excluding them from the visualization, the analysis focuses on instances where meaningful changes occurred, providing a more accurate reflection of the models' performance. The x-axis reflects the average scores on a scale ranging from one to five, originating from the predefined categories discussed in the method section: 'significant deterioration', 'slight deterioration', 'no deterioration/improvement', 'slight improvement', and 'significant improvement'. The y-axis is ordered by the average scores of the two evaluators with the best performing model on top. The chart clearly indicates the best performance is found by the UL2 model in combination with error introduction version 2.

From this chart (Figure 13) it is clear that, according to human judgment, UL2 models perform better than mBART models with the exclusion of UL2 v1. However, mBART v1 is the worst performing model, this is an indication that error introduction version 1 performs worst overall according to human evaluation. The best performing model according to human judgments is UL2 with error-introduction version 2. Furthermore, in the context of mBART model, we observe that humans align with the results from the GLEU+ scores, the mBART with extended training models perform worse than their best-found GLEU-based counterparts. However, this agreement is not observed for the UL2 with extended training models, as UL2 version 3 with extended training slightly outperforms the best-found GLEU-based UL2 version 3 model.

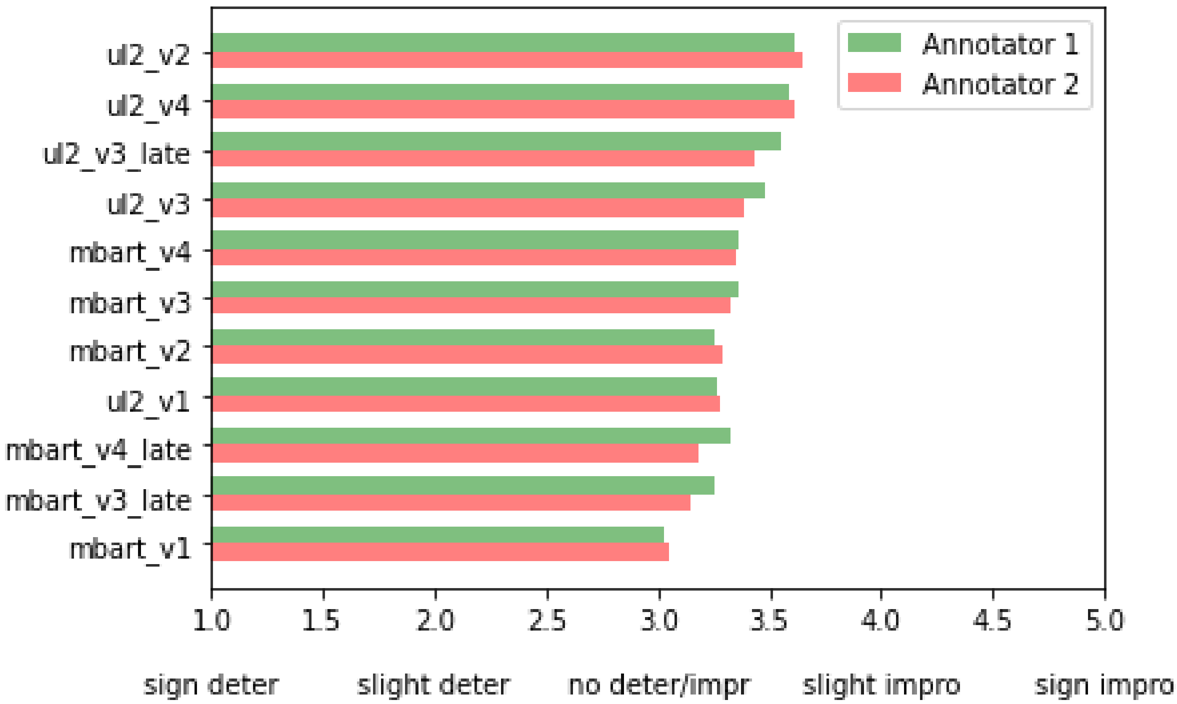


Figure 13: Average human rating on evaluation set

Human Evaluation	Cohen's Kappa	Agreement	Sentence pairs analyzed
11 Models	0.54	Moderate	2574
MC4 vs GPT	0.67	Substantial	136

Table 5: Cohen's Kappa of two human evaluators

Table 5 displays the Cohen's Kappa score between the two evaluators. For the evaluation of the 11 models the Cohen's kappa is 0.54, indicating a moderate level of agreement according to Cohen (1960). The Cohen's Kappa scale ranges from -1 to 1. This score is derived from 2574 sentence pairs, which were compared using dynamic weighting. The remaining sentence pairs from the evaluation set were excluded for the same reasons discussed earlier in the context of Figure 13.

5.3 Semantic preservation

We also present the results for Dutch SBERT sentence similarity and BERTScore in table 6. These metrics assess the similarity between two texts on a scale of 0 to 1. The calculations involve comparing the hand-written target sentence with the hypothesis sentence generated by the model based on a source sentence. These scores are the result of averaging the scores over the 283 sentences of the evaluation set. For BERTScore, UL2 with error introduction version 2 performed best, with both the GPT-generated sentences and MC4 sentences with a score of 0.942. Meanwhile according to cosine similarity based on SBERT, UL2 combined with version 2 and version 4 both performed best with a score of 0.945. These scores between BERTScore and Sentence Similarity cannot be directly compared.

Model	BERTScore	SBERT based similarity
UL2 v1	0.933	0.933
UL2 v2	**0.942**	**0.945**
UL2 v3	0.938	0.944
UL2 v4	0.940	**0.945**
mBART v1	0.933	0.930
mBART v2	0.934	0.935
mBART v3	0.936	0.934
mBART v4	0.937	0.942
UL2 mc4	**0.942**	0.930

Table 6: Semantic similarity measures on the evaluation set

6 Discussion

The primary aim of this research is to investigate the effectiveness of different error introduction methods and end-to-end deep learning models in conjunction with LLM-based generated data. Specifically, the study seeks to assess their impact on enhancing the performance for correcting colloquial Dutch care text while preserving the semantic meaning of the original text. To address the research question, the study is guided by three subquestions.

Subquestion 1, 'How should we generate our Dutch care texts?' aims to find the most appropriate method for generating synthetic data.

To answer the main question we need to identify methods of gauging the quality of the data generation, error-introduction versions and models. Without a comprehensive approach to evaluate the performance of these factors, it would be impossible to draw valid comparisons between different error-introduction versions and models. In essence, Subquestion 2, 'How can we assess the performance of the models, including the generated Dutch care texts and error introduction methods?' acts is the cornerstone that provides the necessary metrics and insights to evaluate the success of each element involved in the correction process. Facilitating a robust analysis of the main research question.

Subquestion 3, 'How can we determine the most suitable error introduction methods and models for effectively correcting colloquial Dutch care text?' is essential to develop practical solutions for improving the quality of Dutch care text. The methods brought forward by answering Subquestion 2 should provide the methods to compare different methodologies and set up an experiment to compare different methods. Additionally, finding an automated metric which aligns with human judgment consistently would be valuable to eliminate the need for human judgments when enhancing error-introduction methods and GPT-generation further.

To answer Subquestion 1, the choice was made to generate synthetic data using GPT since there were no available public Dutch care reports. We experimented with three different approaches, LLM annotation, LLM-generated incorrect text and LLM-generated correct text. LLM-generated correct text was selected as the best for this research since the other approaches suffered from privacy issues, as the available corpus of colloquial care reports was not allowed to be annotated using GPT. Finally the fact that GPT does not excel at generating diverse incorrect text made us step away from direct incorrect text generation.

In the pursuit to answer Subquestion 2, we discovered that there is no effective direct approach to evaluate if two corpuses are similar. Instead, the common protocol involves applying methods and then utilizing automated evaluation metrics and human judgments to compare performance between combinations of approaches, leading to meaningful conclusions.

To answer Subquestion 3, we set up an experiment where we compared four different error-introduction pipelines and two different models using human and automatic evaluation. We applied the error-introduction to 100.000 GPT-3-generated sentences and trained 8 different models on the error-introduction-model combinations. We now present an overview of the results from our investigation.

In terms of human evaluation, the two evaluators achieved a Cohen's kappa of 0.54 on 2574 sentence pairs, indicating a moderate level of agreement. Error introduction version 1 showed the lowest performance in the human evaluation. The combination of the UL2 model with error introduction version 2 performed the best according to human judgments. Notably, human evaluation consistently favored all UL2 models over their mBART counterparts. While human evaluation agreed with GLEU+ on the subpar mBART with extended training models, there was a disagreement about the UL2 with extended training models. Human evaluators preferred the UL2 with extended training models over their counterparts, which contradicted the GLEU+ rankings.

According to GLEU+ results, mBART combined with error introduction version 4 received the highest score. Additionally, GLEU+ suggests that all mBART models outperform their UL2 error-introduction version counterparts. Error introduction version 1 shows the lowest GLEU+ performance for both UL2 and mBART models.

In conclusion, error introduction version 1 is found to have the lowest performance according to both GLEU+ and human evaluation for both model types. Finally, while human evaluators rank all UL2 models higher than their mBART counterparts, GLEU+ rates all mBART models over the UL2 models. We now discuss the results and a final experiment in more detail in the following sections.

6.1 MC4 VS GPT

To investigate further, we aimed to evaluate whether the GPT-generated texts with a relevant topic contribute to the performance of models.

Therefore, we trained a new model using the error introduction method and model that received the highest scores in human evaluation; the UL2 version 2 model. This new model was trained on a distinct corpus of text, specifically a subset of the mC4 Dutch cleaned corpus. Texts were randomly selected from this corpus, cleaned of emojis, and removed if they included characters not present in our GPT-generated dataset. We selected a similar amount of text as used for the other models and applied error introduction version 2 to it. We then trained a UL2 model with this data. Next, we used human evaluation to compare the new model with the top-performing model based on human judgment. This process mirrored the evaluation of the 11 models but involved only two models. To ensure impartiality, the evaluators were unaware of which model generated each text, as we randomized the order of candidate texts. Furthermore, we used the same evaluation set, however we excluded all sentences where both models generated the same output. This choice was made because no distinction in performance can be made among identical texts.

The results are found in Figure 14, additional information is found in Table 5. The figure shows the average score per evaluator for each model based on 136 sentences of the evaluation set. This chart clearly shows the superiority of the model trained on GPT-generated data. Both annotators reached a significantly higher average score for the UL2 GPT model. On average the GPT model reached an average score of 3.65 and the mC4 model an average score of

3.29. These results indicate that the GPT-generated relevant texts do improve the model performance of a GEC model.

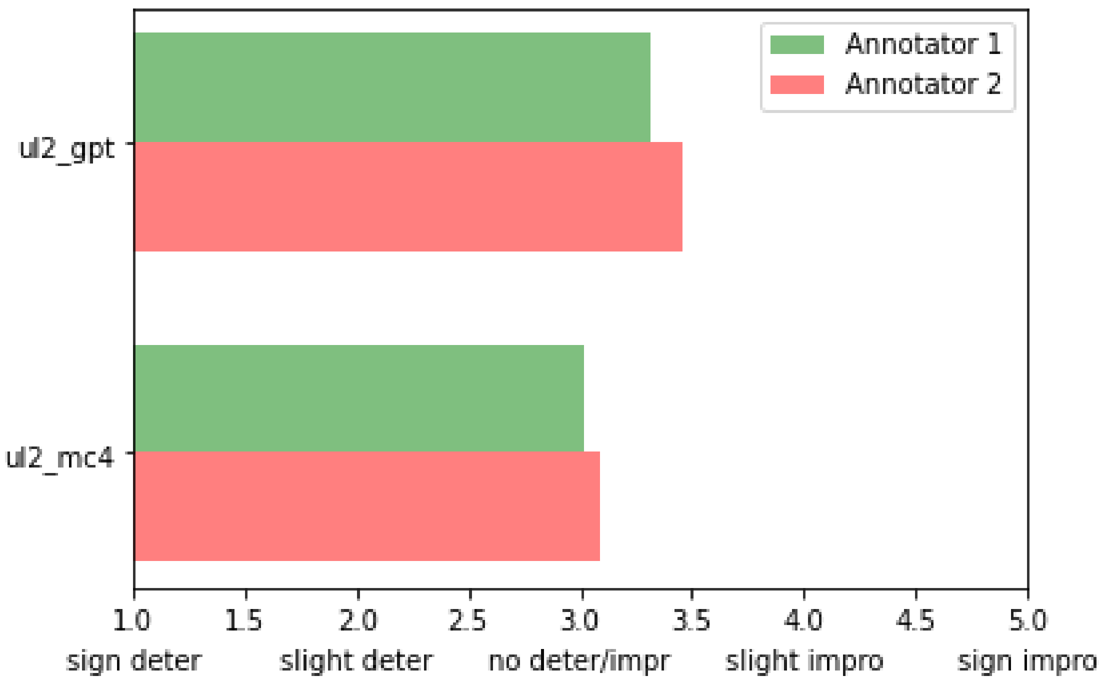


Figure 14: Average human rating on evaluation set MC4 and GPT

6.2 Semantic similarity

According to BERTScore scores, the UL2 model along with error-introduction version 2 performs best, for both the GPT-generated sentences and randomly selected MC4 sentences with a score of 0.942 (Table 6). Furthermore, according to SBERT embeddings and cosine similarity, UL2 version 2 and 4 with GPT-generated sentences were the top performing models with a score of 0.945.

Since these methods were originally designed to identify paraphrases and perform semantic search, these methods are tailored to assess text on a scale of 0 to 1, spanning from complete semantic dissimilarity to identical meaning. However, since we use these scores for GEC evaluation, where generally most of the text should remain unchanged, the scores we see are in the upper bounds of the score range. It is interesting to note that in the GEC context, where we look at small differences between hypothesis and target sentences, the metrics seem to hold up as they provide similar top models to human evaluation. These metrics agree on the superiority of the UL2 models, and the worst performance of error-introduction version 1. What is interesting is the distinction between the metrics for UL2 in combination with MC4. While BERTScore selects the MC4 model as one of the best, SBERT based embeddings identify MC4 as tied worst performer. This suggests that SBERT based cosine similarity is more in line with human judgments.

6.3 Contrasting Metrics: GLEU+ vs. Human Evaluation

When we compare error introduction methods per metric and model type we see differing trends. Examining GLEU+ scores (Figure 12), mBART achieves the highest performance with versions 4, 2, 3, and 1, respectively. In contrast, UL2 model variations exhibit superior performance in the order of 2, 4, 3, and 1. Turning our attention to human scores (Figure 13), for mBART, versions 4, 3, 2, and 1 emerge as the top performers. Conversely, error introduction versions 2, 4, 3, and 1 exhibit the best performance for UL2.

From this we can conclude that error introduction version 1 performed worst for all models according to the metrics. Furthermore we see that for the UL2 model, error introduction versions 2 and 4 perform best according to both metrics. While mBART performs best with version 4 according to all metrics. However, what the metrics do not agree on is which model type performs best.

During model training (Figure 12) a noteworthy observation arises from the comparison of mBART and UL2 models within each error introduction method. Specifically, mBART versions (v1, v2, v3, and v4) consistently demonstrate superior performance over their UL2 counterparts (v1, v2, v3, and v4) according to the GLEU+ scores. This is a strong indication that the mBART models generally outclass UL2 models according to the GLEU+ scores. Yet, according to the human evaluation in Figure 13 and Table 7 UL2 models systematically outperform their counterpart mBART model.

This is opposite to what we expected as in initial experiments UL2 models performed better than mBART models. This led to further investigation into the GLEU+ scores in an post-training evaluation.

6.4 Comparing GLEU+ scores: disparities between post-training and in-training evaluation results

We conducted a post-training analysis of the generated hypothesis sentences from the best-selected models, utilizing various evaluation metrics, including GLEU+. Surprisingly, we observed significant deviations in GLEU+ values for these texts compared to the results obtained during training, despite using the exact same evaluation set (Table 7). Although the stochastic nature of the models can lead to slightly varying results, some of the observed disparities significantly surpass the expected range, especially for the UL2 models. mBART versions 1, 2, 3 and 4 score within the expected range of each other for both in-training and post-training evaluations. However UL2 versions 1, 2, 3 and 4 all score significantly better during post-training GLEU+ evaluation. After further investigation we excluded that the disparity originated from the GLEU+ calculation itself. We expect that the problem lies in the encoding-decoding process of hypothesis texts and source texts during training, specifically due to the UL2 prefix. We expect this is part of the problem since this is the only discernable difference between the use of mBART and UL2. However, after thorough investigation, we were unable to pinpoint the exact source of this discrepency.

This implies that the in-training GLEU+ scores may be corrupted. We exclusively relied on these in-training GLEU+ scores to choose the best models, which we subsequently employed to generate texts for human evaluation. In the worst-case scenario, our selection of models for human evaluation could be based on a flawed metric, resulting in a random rather than merit-based selection. Nevertheless, we observe that the top-performing models, identified through in-training GLEU+ scores, exhibit consistent rankings for error introduction models when compared to the post-training evaluation, despite yielding different absolute values. This observation suggests that, in spite of the corruption, the metric is still informative when selecting the best relative model, despite the absolute value being wrong.

While encountering this issue is undesirable for any researcher, it also introduced a new set of GLEU+ scores which were more in line with our expected results (7). Both the post-training GLEU+ scores and human evaluations rank UL2 version 2 and UL2 version 4 as the top two model-error introduction combinations. Notably, the post-training evaluation did not include the 'with extended training' versions of the models. Despite this limitation, UL2 version 3 is ranked as the third-best model by the post-training evaluation metric. In contrast, human scoring places UL2 version 3 with extended training as the third-best and UL2 version 3 as the fourth best, indicating a close alignment in rankings despite the absence of 'with extended training' versions in the post-training evaluation. The top three models display a significantly higher post-training GLEU+ score than the other models.

In light of the post-training GLEU+ evaluation results, which demonstrated a close alignment with human evaluation results, it implies that the GLEU+ metric correlates effectively with human judgments. This finding suggests the potential utility of the GLEU+ metric for refining the error introduction method, dependant upon successful resolution of the unidentified bug.

Model	**Human Score**	**i-t GLEU+**	**p-t GLEU+**	**Human Rank**	**i-t GLEU+ Rank**	**p-t GLEU+ Rank**
mBART v1	3.04	0.510	0.508	11	5	8
mBART v2	3.27	0.529	0.525	7	2	4
mBART v3	3.34	0.518	0.512	6	3	7
mBART v4	3.36	**0.534**	0.522	5	1	5
UL2 v1	3.27	0.504	0.522	8	9	6
UL2 v2	**3.63**	0.517	**0.549**	1	4	1
UL2 v3	3.43	0.507	0.537	4	7	3
UL2 v4	3.60	0.513	0.541	2	6	2
mBART v3 ext train	3.20	0.504	n.d	10	10	n.d
mBART v4 ext train	3.26	0.507	n.d	9	8	n.d
UL2 v3 ext train	3.49	0.498	n.d	3	11	n.d

Table 7: Model evaluation by humans, in-training GLEU+, post-training GLEU+, and their respective rankings amongst themselves

6.5 Human evaluation analysis

Human evaluators demonstrated a moderate and substantial level of agreement for their evaluation, as indicated by the Cohen's Kappa score of 0.54 and 0.67, during the assessment of 11 distinct models and MC4 vs GPT. Choshen and Abend (2018b) argue that the reason inter-rater-agreement (IRA) for human evaluation scores is generally low because people often disagree when rating how grammatical sentences are, since it is a highly subjective task. While the levels of moderate and substantial agreement are satisfactory, our aim is to identify variations among evaluators. To achieve this, evaluators engaged in a discussion about the task, revealing specific challenges.

In our evaluation process, the assessors were the author of the paper and an individual not directly involved in the research, both possessing similar educational backgrounds. The first thing they noticed when discussing the differing scores was that one evaluator was more extreme in judging mistakes and positives while the other was more reserved. For example, this led to one evaluator scoring a sentence with a 1 or a 5, whereas the other evaluator would sooner use a 2 or a 4. Our interpretation of these observations suggests that the author, being intimately involved in the development of the models, adopted a more lenient stance, striving to understand and appreciate their performance. In contrast, the other evaluator, not directly connected to the model creation, adopted a more critical perspective.

Secondly, the familiarity of one evaluator with the texts granted them a deeper understanding of the original context of the sentences, compared to the other evaluator. This impacted the understanding of sentences between evaluators, which could be found in the scoring. Furthermore, this resulted in one evaluator penalizing the introduction of contextual information not explicitly present in the text, while the other evaluator occasionally rewarded such additions if it agreed with his knowledge of the text. In future research this should be discussed in even more detail in the instructions for human evaluation.

6.6 Identified patterns during human evaluation

The human evaluators examined numerous corrections produced by the correction models. On average, they observed an improvement in input text quality but identified recurring issues. The main issue was the product of a lack of contextual information for the models to use when correcting text. This led to models fabricating information to create a grammatical text, while the semantics changed from human interpretation (Example 3 in Table 8). This is problematic for care texts as it is required that they are trustworthy. However, it is often impossible for even humans to agree on the meaning of these errorful texts. To address this issue, future model designs should prioritize providing access to extensive contextual information. This may include surrounding sentences or a knowledge graph derived from previous care reports, aiming to maximize the available information.

Furthermore, in some care entries, vital information present in the errorful text is omitted to facilitate the creation of grammatically correct text (Example 4 and 5 in Table 8). This is problematic, as it may result in the loss of potentially crucial meaning in these corrections. The primary challenge in this context arises from the poor formulation of these texts, leading to uncertainty regarding their intended meaning. Consequently, creating a grammatically accurate sentence with certainty of conveying the identical intended meaning of the unintelligible input text becomes impossible. This suggests that there is a trade-off between ensuring grammaticality in the generated text and capturing all possible available information from the input text. For the medical field, capturing all available information is off the essence but the current models seem to prioritize creating grammatical sentences. This means the model is not ready to be used without confirmation from the caregivers if the correction it presents conveys their intended message.

Finally, both evaluators noted a significant enhancement in model performance stemming from the introduction of punctuation errors to address the care entries that lacked punctuation (Example 2 in Table 8). This indicates that this is an effective method to combat interpunction errors.

Input text	**UL2-v2 correction**
Toen mijn dienst gisteren was er die middag niemand van personeel behalve de gastvrouw in de woonkamer.	Toen mijn dienst gisteren begon , was er die middag niemand van personeel aanwezig behalve de gastvrouw in de woonkamer.
Mevrouw haar stuit is nog heel ziet er slecht uit n ieuwe vlinderplak op gedaan .	Mevrouw haar stuit is nog heel , maar ziet er slecht uit . Ik heb er een vlinderplak op gedaan.
O mw vanmorgen douche aangeboden maar mw wilde het niet.	Mevrouw had vanmorgen een douche aangeboden , maar mevrouw wilde het niet.
O. mw kan moeilijk staan met collega mw in de rolstoel geholpen mw eet weinig wil geen bouwsteentje mw heeft halve boterham gegeten en kopje koffie glaasje sap	Mevrouw kan moeilijk staan met haar collega. Mevrouw eet weinig en wil geen bouwsteentje . Mevrouw heeft een halve boterham gegeten en een glaasje sap gedronken.
Graag apotheek bellen klopt niet wat is binnen gekomen voor de catheter.	Graag de apotheek bellen, wat is binnen gekomen voor de catheter.

Table 8: Example model corrections. Blue indicates a missing piece of information in the correction, green indicates a good correction, yellow indicates a grammatical but uncertain correction and red indicates a wrong interpretation

6.7 Limitations

While this study has yielded valuable insights into Dutch care GEC, it is essential to acknowledge and address certain limitations inherent in the research design, methodology, and execution.

First of all, the implementation of the GLEU+ metric during training is likely corrupted. The selection of models to be evaluated by humans was affected by this.

Secondly, the implementation of the word-list for the spellchecker enables very common errors to be present in the word-list, thereby disabling a spellchecker to correct them. Consequently, errors more common than 170 instances were not identified and were not introduced into our synthetic data through spell-checker replacement. However, it is still possible for them to be introduced through character error-introduction, but ideally they should be introduced with replacement too.

Thirdly, the evaluation set which we utilized in this research was annotated by the author of the paper instead of a linguistic professional, which possibly led to faulty target sentences.

Next, context supplies essential information that is critical for fixing various grammatical errors and addressing inconsistencies. Our approach performs sentence-based GEC, without additional contextual information. This means there is a lack of contextual information fed to the system while it does exist in surrounding sentences. Consequentially, these systems fail to correct verb tense, pronoun and numerous types of errors, since they lack the information of surrounding text. Additionally, this leads to corrections which convey a different message than intended in the input text.

Additionally, the GPT generated texts were only evaluated in combination with many different factors. There is room for much more elaborate evaluation of the generated texts themselves to decide the best alignment with our corpus.

Furthermore, we performed human evaluation on 3113 pairs of sentences by two human evaluators. One of the evaluators was the author which could have possible introduced a bias. The inclusion of more evaluators would increase the reliability of this research and the conclusions about the automated metrics. It's essential to emphasize that, during human evaluation, sentences were generally evaluated on a scale of 1-5. In future research, it would be beneficial to incorporate separate scoring dimensions for assessment, such as fluency, grammaticality, and semantic similarity, to provide a more nuanced and detailed analysis.

Finally as mentioned in method section, addressing the variability in correcting incorrect text poses a challenge in GEC. With multiple possible corrections, determining the optimal one becomes subjective, relying on individual perceptions. Therefore, there is room for an extended amount of reference sentences, since only one is used per text. This means that currently models might accurately correct text differently from the reference text and face penalties for it, which is problematic.

6.8 Implications

This research provides insight in grammatical error correction for Dutch care texts using seq2seq models and synthetic data. We identify the UL2 model as a better model than the mBART model according to human judgments, post-training GLEU+ scores and semantic similarity metrics. Furthermore we find that error-introduction which introduces interpunction errors and bigram replacement in addition to the baseline errors based on Náplava and Straka (2019) performs best according to human judgments and semantic consistency analysis. According to human evaluation, our method to introduce interpunction errors worked well for this corpus. Granted that the in-training GLEU+ evaluation is fixed, we find that GLEU+ is able to identify the best performing error-introduction-model-corpus combination compared to human evaluation. Which can significantly decrease evaluation time when further enhancing the correction models.

6.9 Future research

This research opens up avenues for various potential further investigations.

Building upon this research, additional exploration can be conducted to transform the corrected sentences into formal, summary-style texts using paraphrasing and summarizing models. This would complete the request of care facilities to utilize their care entries to inform relatives in a more formal manner.

Furthermore, increased performance could be achieved by exploring methodologies to provide the models with more context, such as providing models with surrounding sentences, a client's care plan or all their care entries. This could improve performance significantly cross-sentence.

Moreover, it would be intriguing to further investigate direct annotation using LLMs on non-private data, or with an open-source LLM to evaluate the effectiveness of creating synthetic training data using this approach.

In depth research can be performed in different training regimes for training the seq2seq transformer models, as currently the default model settings were used. Potential improvements in performance could be explored through experimentation with different learning rates, optimizers, and the creation of additional hand-annotated data for fine-tuning after pretraining on synthetic data, which is an effective way to improve performance according to Rothe et al. (2021).

The error introduction methods can be expended upon significantly. Currently, the same error introduction rates were used as in the baseline approach for substitution, deletion, insertion and swapping. These could be experimented with further since our corpus arguably has a different distribution of errors than general GEC. Additionally, the metaphone algorithm of Philips (2000) could be integrated to expand on the Levenshtein distance with phonetic equivalents to find erroneous words based on phonetic mistakes for the spellchecker word-list, as Grundkiewicz et al. (2019) have done. Finally, there are limitless possible additions to be made to error-introduction pipelines. To list a few: introduce Dutch 'dt' errors, synonymn replacement, determiner replacement, smart word deletion, error-type extraction from care corpus and generating more text with medical abbreviations. It would be interesting to delve deeper into an error introduction method based solely on linguistic errors. This approach moves away from the conventional techniques of general deletion, substitution, insertion, and swapping, opting instead to embrace strategies involving POS, phonemes, NER and more.

It would be valuable to work on more advanced evaluation metrics for Dutch. Such as with tuning a Dutch pre-trained BERT model for BERTScore to improve the performance of BERTScore. Additionally, extending the ERRANT or M^2 metric to Dutch and applying it to this problem would be a highly valuable asset for Dutch GEC as this would create more insight in the distribution of different error-types.

7 Conclusion

In conclusion, there exists a high amount of information within care reports however they contain errors. In response, on behalf of PrimeVision, our objective is to develop a system capable of correcting these errorful care reports. This research aimed to address the main research question: "What is the effectiveness of different error introduction methods in combination with LLM-based generated data in enhancing the performance of end-to-end deep learning models for correcting colloquial Dutch care text while preserving the semantic meaning of the original text?" To structure the investigation, three subquestions were formulated.

- **Subquestion 1:** How should we generate our Dutch care texts?
- **Subquestion 2:** How can we assess the performance of the models, including the generated Dutch care texts and error introduction methods?
- **Subquestion 3:** How can we determine the most suitable error introduction methods and models for effectively correcting colloquial Dutch care text?

For Subquestion 1, GPT was chosen to generate synthetic Dutch care texts due to the absence of public datasets. Three approaches were tested; LLM annotation, LLM-generated incorrect text, and LLM-generated correct text. Opting for LLM-generated correct text was based on its suitability for the research, as privacy issues were associated with LLM annotation. GPT's limited ability to generate incorrect text led to the decision to avoid incorrect text generation.

To answer Subquestion 2, literature research identified effective methodologies for evaluating text quality, concluding that no strong direct approach is available. Instead, the conventional approach involves setting up an experiment and subsequently using evaluation metrics to compare the performance of different combinations of models, generated text and error introduction methods. To evaluate performance, the automated evaluation metrics GLEU+, BERTScore and SBERT sentence similarity were used. Furthermore human judgments in combination with Cohen's kappa were employed.

To address Subquestion 3, an experiment compared four error-introduction pipelines with two models, evaluated through both human and automatic methods. This resulted in the assessment of eight models. Additionally, three more models underwent extended training, bringing the total number of evaluated correction models to 11. Human evaluation achieved a Cohen's kappa of 0.54, indicating a moderate level agreement over 2574 sentence pairs. Both human and automatic evaluation scored the general low-resource error-introduction method (version 1) as the worst for both model types. Furthermore, humans scored the UL2 model in combination with interpunction error & replace bigram errors (version 2) as the best model-error combination. However, in-training GLEU+ scores indicated mBART and syntactic dependency based deletion (version 4) as the best model-error combination. Next, human evaluation consistently scored the mBART with extended training models as worse than their regular training counterparts while they preferred the UL2 with extended training model slightly over it's regular training counterpart.

The mBART error introduction-model combinations consistently exhibit superior performance over their UL2 counterparts, as evidenced by consistently higher in-training GLEU+ scores. This suggests a general proficiency of mBART models compared to UL2 models based on GLEU+ scores. However, Figure 13 and Table 7 reveal a contrasting outcome in human evaluation, where UL2 models consistently outperform their mBART counterparts. Given the unexpected reversal from our initial experiments, we conducted a post-training experiment to further investigate. Despite using the same evaluation set, we observed significant deviations in GLEU+ values compared to the training results, particularly for UL2 models. While mBART versions demonstrated consistency in both in-training and post-training evaluations, UL2 versions, exhibited a significant improvement in GLEU+ scores during post-training evaluation. Despite being unable to pinpoint the exact source of the discrepancy, our suspicion centers on the encoding-decoding process during training, particularly linked to the UL2 prefix. Although our reliance on in-training GLEU+ scores for model selection introduces the possibility of flawed model selection, the top-performing model-error introduction combinations were consistent in both in-training and post-training evaluations. This resulted in a new set of post-training GLEU+ scores aligning more closely with our initial expectations. Post-training GLEU+ scores exhibited a notable correlation with human evaluation results, suggesting the metric's effectiveness in reflecting human judgments. This implies the potential utility of the GLEU+ metric for refining the error introduction method, contingent upon resolving the unidentified bug that caused the observed discrepancies in the in-training GLEU+ scores.

Semantic similarity evaluation metrics BERTScore and SBERT based similarity further confirmed the superiority of UL2 paired with error-introduction version 2 compared to model-error version combinations. A notable observation is the disparity in metric evaluations for UL2 in combination with MC4. While BERTScore designates UL2 MC4 and UL2 v2 as the best models, SBERT-based embeddings consider MC4 a tied worst performer. This discrepancy suggests that SBERT-based similarity metric aligns better with human judgments.

Finally, to answer the main research question: "What is the effectiveness of different error introduction methods in combination with LLM-based generated data in enhancing the performance of end-to-end deep learning models for correcting colloquial Dutch care text while preserving the semantic meaning of the original text?" We conclude that the UL2 model in combination with error introduction interpunction error & replace bigram errors performs best in correcting colloquial Dutch care text according to human evaluation, post-training GLEU+ scores and semantic preservation metrics. Furthermore the mC4 vs GPT experiment concluded that the generated care reports are beneficial compared to off-topic Dutch text grammatical error correction in the low-resource language Dutch. Finally, the post-training GLEU+ metric aligns with human judgments and could replace the need for time-intensive human evaluation in further enhancement of models.

This research opens the door to various possibilities for further exploration and enhancement of low-resource GEC models. The findings underscore the effectiveness of GPT-generated texts for grammatical error correction in the low-resource language Dutch. Experiments highlight the effectiveness of the UL2 model when paired with the error introduction version incorporating interpunction error and replacing bigrams. Furthermore, the results emphasize the effectiveness of BERTScore and SBERT to evaluate semantics preservation, as they align with human evaluations.

References

Hiroki Asano, Tomoya Mizumoto, and Kentaro Inui. Reference-based metrics can be replaced with reference-less metrics in evaluating grammatical error correction systems. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 343–348, 2017.

Satanjeev Banerjee and Alon Lavie. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, pages 65–72, 2005.

Ondřej Bojar, Yvette Graham, Amir Kamran, and Miloš Stanojević. Results of the wmt16 metrics shared task. In *Proceedings of the First Conference on Machine Translation: Volume 2, Shared Task Papers*, pages 199–231, 2016.

Adriane Boyd. Using wikipedia edits in low resource grammatical error correction. In *Proceedings of the 2018 EMNLP Workshop W-NUT: The 4th Workshop on Noisy User-generated Text*, pages 79–84, 2018.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Christopher Bryant and Hwee Tou Ng. How far are we from fully automatic high quality grammatical error correction? In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 697–707, 2015.

Christopher Bryant, Mariano Felice, Øistein E. Andersen, and Ted 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*, pages 52–75, Florence, Italy, August 2019. Association for Computational Linguistics. doi: 10.18653/v1/W19-4406. URL https://aclanthology.org/W19-4406.

Christopher Bryant, Zheng Yuan, Muhammad Reza Qorib, Hannan Cao, Hwee Tou Ng, and Ted Briscoe. Grammatical error correction: A survey of the state of the art. *Computational Linguistics*, pages 1–59, 2022.

Yo Joong Choe, Jiyeon Ham, Kyubyong Park, and Yeoil Yoon. A neural grammatical error correction system built on better pre-training and sequential transfer learning. *arXiv preprint arXiv:1907.01256*, 2019.

Leshem Choshen and Omri Abend. Inherent biases in reference based evaluation for grammatical error correction and text simplification. *arXiv preprint arXiv:1804.11254*, 2018a.

Leshem Choshen and Omri Abend. Reference-less measure of faithfulness for grammatical error correction. *arXiv preprint arXiv:1804.03824*, 2018b.

Diego Barreiro Clemente. Predicting care plan goals using a knowledge graph-based recommender system. *Journal Name*, 2023.

Jacob Cohen. A coefficient of agreement for nominal scales. *Educational and psychological measurement*, 20(1):37–46, 1960.

Daniel Dahlmeier and Hwee Tou 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*, pages 568–572, 2012.

Daniel Dahlmeier, Hwee Tou Ng, and Siew Mei 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*, pages 22–31, 2013.

Maxime De Bruyn, Ehsan Lotfi, Jeska Buhmann, and Walter Daelemans. MFAQ: a multilingual FAQ dataset. In *Proceedings of the 3rd Workshop on Machine Reading for Question Answering*, pages 1–13, Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. URL https://aclanthology.org/2021.mrqa-1.1.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Christian Federmann. Appraise: an open-source toolkit for manual evaluation of mt output. *Prague Bull. Math. Linguistics*, 98:25–36, 2012.

Mariano Felice and Zheng Yuan. Generating artificial errors for grammatical error correction. In *Proceedings of the Student Research Workshop at the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 116–126, 2014.

Mariano Felice, Christopher Bryant, and Ted Briscoe. Automatic extraction of learner errors in esl sentences using linguistically enhanced alignments. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 825–835, 2016.

Peiyuan Gong, Xuebo Liu, Heyan Huang, and Min Zhang. Revisiting grammatical error correction evaluation and beyond. *arXiv preprint arXiv:2211.01635*, 2022.

Takumi Gotou, Ryo Nagata, Masato Mita, and Kazuaki Hanawa. Taking the correction difficulty into account in grammatical error correction evaluation. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 2085–2095, Barcelona, Spain (Online), December 2020. International Committee on Computational Linguistics. doi: 10.18653/v1/2020.coling-main.188. URL `https://aclanthology.org/2020.coling-main.188`.

Roman Grundkiewicz, Marcin Junczys-Dowmunt, and Edward Gillian. Human evaluation of grammatical error correction systems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 461–470, 2015.

Roman Grundkiewicz, Marcin Junczys-Dowmunt, and Kenneth Heafield. Neural grammatical error correction systems with unsupervised pre-training on synthetic data. In *Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 252–263, 2019.

Matthew Honnibal and Ines Montani. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. To appear, 2017.

Phu Mon Htut and Joel Tetreault. The unbearable weight of generating artificial errors for grammatical error correction. *arXiv preprint arXiv:1907.08889*, 2019.

Md Asadul Islam and Enrico 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*, pages 3009–3015, 2021.

Shun Kiyono, Jun Suzuki, Masato Mita, Tomoya Mizumoto, and Kentaro Inui. An empirical study of incorporating pseudo data into grammatical error correction. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1236–1242, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1119. URL `https://aclanthology.org/D19-1119`.

Aomi Koyama, Tomoshige Kiyuna, Kenji Kobayashi, Mio Arai, and Mamoru Komachi. Construction of an evaluation corpus for grammatical error correction for learners of japanese as a second language. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 204–211, 2020.

Shaopeng Lai, Qingyu Zhou, Jiali Zeng, Zhongli Li, Chao Li, Yunbo Cao, and Jinsong Su. Type-driven multi-turn corrections for grammatical error correction. *arXiv preprint arXiv:2203.09136*, 2022.

Lung-Hao Lee, Yuen-Hsien Tseng, and Li-Ping Chang. Building a tocfl learner corpus for chinese grammatical error diagnosis. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, 2018.

Vladimir I Levenshtein et al. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, pages 707–710. Soviet Union, 1966.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault, editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.703. URL `https://aclanthology.org/2020.acl-main.703`.

Jared Lichtarge, Chris Alberti, Shankar Kumar, Noam Shazeer, Niki Parmar, and Simon Tong. Corpora generation for grammatical error correction. *arXiv preprint arXiv:1904.05780*, 2019.

Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81, 2004.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

Yinhan Liu, Jiatao Gu, Naman Goyal, Xian Li, Sergey Edunov, Marjan Ghazvininejad, Mike Lewis, and Luke Zettlemoyer. Multilingual Denoising Pre-training for Neural Machine Translation. *Transactions of the Association for Computational Linguistics*, 8:726–742, 11 2020. ISSN 2307-387X. doi: 10.1162/tacl_a_00343. URL `https://doi.org/10.1162/tacl\_a\_00343`.

Nitin Madnani, Joel Tetreault, and Martin Chodorow. Exploring grammatical error correction with not-so-crummy machine translation. In *Proceedings of the Seventh Workshop on Building Educational Applications Using NLP*, pages 44–53, 2012.

Behrang Mohit, Alla Rozovskaya, Nizar Habash, Wajdi Zaghouani, and Ossama Obeid. The first qalb shared task on automatic text correction for arabic. In *Proceedings of the EMNLP 2014 Workshop on Arabic Natural Language Processing (ANLP)*, pages 39–47, 2014.

Jakub Náplava and Milan Straka. Grammatical error correction in low-resource scenarios. *arXiv preprint arXiv:1910.00353*, 2019.

Courtney Napoles, Keisuke Sakaguchi, Matt Post, and Joel 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)*, pages 588–593, 2015.

Courtney Napoles, Keisuke Sakaguchi, Matt Post, and Joel Tetreault. Gleu without tuning. *arXiv preprint arXiv:1605.02592*, 2016a.

Courtney Napoles, Keisuke Sakaguchi, and Joel Tetreault. There's no comparison: Reference-less evaluation metrics in grammatical error correction. *arXiv preprint arXiv:1610.02124*, 2016b.

Courtney Napoles, Maria Nădejde, and Joel Tetreault. Enabling robust grammatical error correction in new domains: Data sets, metrics, and analyses. *Transactions of the Association for Computational Linguistics*, 7:551–566, 2019.

Hwee Tou Ng, Siew Mei Wu, Ted Briscoe, Christian Hadiwinoto, Raymond Hendy Susanto, and Christopher Bryant. The conll-2014 shared task on grammatical error correction. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task*, pages 1–14, 2014.

Jekaterina Novikova, Ondřej Dušek, and Verena Rieser. Rankme: Reliable human ratings for natural language generation. *arXiv preprint arXiv:1803.05928*, 2018.

Kostiantyn Omelianchuk, Vitaliy Atrasevych, Artem Chernodub, and Oleksandr Skurzhanskyi. Gector–grammatical error correction: tag, not rewrite. *arXiv preprint arXiv:2005.12592*, 2020.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318, 2002.

Lawrence Philips. The double metaphone search algorithm. *C/C++ users journal*, 18(6): 38–43, 2000.

Muhammad Qorib, Seung-Hoon Na, and Hwee Tou Ng. Frustratingly easy system combination for grammatical error correction. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1964–1974, 2022.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv e-prints*, 2019.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551, 2020.

Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 11 2019. URL `http://arxiv.org/abs/1908.10084`.

Sascha Rothe, Jonathan Mallinson, Eric Malmi, Sebastian Krause, and Aliaksei Severyn. A simple recipe for multilingual grammatical error correction. *arXiv preprint arXiv:2106.03830*, 2021.

Alla Rozovskaya and Dan Roth. Generating confusion sets for context-sensitive error correction. In *Proceedings of the 2010 conference on empirical methods in natural language processing*, pages 961–970, 2010.

Alla Rozovskaya and Dan Roth. Grammar error correction in morphologically rich languages: The case of russian. *Transactions of the Association for Computational Linguistics*, 7:1–17, 2019.

Keisuke Sakaguchi and Benjamin Van Durme. Efficient online scalar annotation with bounded support. *arXiv preprint arXiv:1806.01170*, 2018.

Keisuke Sakaguchi, Courtney Napoles, Matt Post, and Joel Tetreault. Reassessing the goals of grammatical error correction: Fluency instead of grammaticality. *Transactions of the Association for Computational Linguistics*, 4:169–182, 2016.

Max Scheijen. Dutch news articles, 2022. URL `https://www.kaggle.com/ds/1013130`.

Rico Sennrich, Barry Haddow, and Alexandra Birch. Edinburgh neural machine translation systems for wmt 16. *arXiv preprint arXiv:1606.02891*, 2016.

Sander van Geloven Simon Brouwer. Opentaal wordlist. `https://github.com/OpenTaal/opentaal-wordlist`, 2023. Accessed: January 28, 2024.

Xin Sun and Houfeng Wang. Adjusting the precision-recall trade-off with align-and-predict decoding for grammatical error correction. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 686–693, 2022.

Oleksiy Syvokon and Olena Nahorna. Ua-gec: Grammatical error correction and fluency corpus for the ukrainian language. *arXiv preprint arXiv:2103.16997*, 2021.

Toshikazu Tajiri, Mamoru Komachi, and Yuji Matsumoto. Tense and aspect error correction for esl learners using global context. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 198–202, 2012.

Yuqing Tang, Chau Tran, Xian Li, Peng-Jen Chen, Naman Goyal, Vishrav Chaudhary, Jiatao Gu, and Angela Fan. Multilingual translation with extensible multilingual pretraining and finetuning. 2020.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B Hashimoto. Alpaca: A strong, replicable instruction-following model. *Stanford Center for Research on Foundation Models. https://crfm. stanford. edu/2023/03/13/alpaca. html*, 3(6):7, 2023.

Yi Tay, Mostafa Dehghani, Vinh Q Tran, Xavier Garcia, Jason Wei, Xuezhi Wang, Hyung Won Chung, Dara Bahri, Tal Schuster, Steven Zheng, et al. Ul2: Unifying language learning paradigms. In *The Eleventh International Conference on Learning Representations*, 2022.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Max White and Alla Rozovskaya. A comparative study of synthetic data generation methods for grammatical error correction. In *Proceedings of the Fifteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 198–208, 2020.

Ziang Xie, Guillaume Genthial, Stanley Xie, Andrew Y Ng, and Dan Jurafsky. Noising and denoising natural language: Diverse backtranslation for grammar correction. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 619–628, 2018.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems*, 32, 2019.

Ryoma Yoshimura, Masahiro Kaneko, Tomoyuki Kajiwara, and Mamoru Komachi. Some: Reference-less sub-metrics optimized for manual evaluations of grammatical error correction. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6516–6522, 2020.

Zheng Yuan and Mariano Felice. Constrained grammatical error correction using statistical machine translation. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning: Shared Task*, pages 52–61, 2013.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. Bertscore: Evaluating text generation with bert. *arXiv preprint arXiv:1904.09675*, 2019a.

Yuan Zhang, Jason Baldridge, and Luheng He. Paws: Paraphrase adversaries from word scrambling. *arXiv preprint arXiv:1904.01130*, 2019b.

Wei Zhao, Liang Wang, Kewei Shen, Ruoyu Jia, and Jingming Liu. Improving grammatical error correction via pre-training a copy-augmented architecture with unlabeled data. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 156–165, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1014. URL https://aclanthology.org/N19-1014.

Zewei Zhao and Houfeng Wang. Maskgec: Improving neural grammatical error correction via dynamic masking. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 1226–1233, 2020.

Jianing Zhou and Suma Bhat. Paraphrase generation: A survey of the state of the art. In *Proceedings of the 2021 conference on empirical methods in natural language processing*, pages 5075–5086, 2021.

Wangchunshu Zhou, Tao Ge, Chang Mu, Ke Xu, Furu Wei, and Ming Zhou. Improving grammatical error correction with machine translation pairs. *arXiv preprint arXiv:1911.02825*, 2019.

A Appendix A - Human evaluation guidelines

Deze enquête vergelijkt verschillende tekstverbeteringsmodellen. Je krijgt een originele zin te zien en 11 verbeteringen die door modellen zijn gemaakt op basis van de originele zin. Beoordeel de verbeteringen van de modellen. Hierbij heeft u de keuze uit: erg verslechterd, een beetje verslechterd, geen verbetering/verslechtering, een beetje verbeterd en erg verbeterd. Er zijn veel verschillende fouten die op kunnen treden bij het verbeteren van tekst. Let op: 1) Semantische betekenis: betekent de verbeterde zin nog hetzelfde als de originele zin? 2) Vloeiendheid: loopt de zin goed en wordt er goede interpunctie gebruikt. 3) Grammaticale correctheid: de zin is in correct Nederlands geschreven. 4) Spelling: woorden zijn correct geschreven. Vaak zijn er verschillende manieren om een tekst te verbeteren. Zo lang de verbetering correct Nederlands is en de semantische betekenis niet wordt veranderd is de verbetering goed. Soms is er te weinig context in de originele zin aanwezig om een correcte zin te maken voor de modellen. Als dit het geval is hou hier dan rekening mee door verschillende interpretaties van de originele zin te overwegen. Bijvoorbeeld: 'Mevrouw gewassen' → 'Mevrouw is gewassen' is een correcte verbetering aangezien de context mist. Maar: 'Mevrouw heeft zich gewassen' is ook een correcte verbetering. Wanneer een zin op een bepaald vlak is verbeterd (bijvoorbeeld interpunctie) maar ergens anders achteruit is gegaan (bijvoorbeeld semantische betekenis) kan je kiezen voor geen verbetering of verslechtering. De originele zinnen zijn fragmenten uit notities uit het zorgdomein die geschreven zijn door medische professionals. Hierin zijn veel standaard afkortingen aanwezig: (O: → Observatie, S: → Patient zei het volgende, P: → Procedure). Probeer bij de afkortingen na te denken wat ze kunnen betekenen en of ze relevant zijn om opgenomen te worden in een correcte zin.

B Appendix B - Prompts

LLM-annotation:

'role' : "system", "content":"Verbeter nederlandse medische tekst. Als de input verbeterd kan worden, geef de verbeterde versie van de tekst en niets anders. Als de input niet verbeterd kan worden, reageer met de input tekst. Als de input onbegrijpelijk is reageer dan met de input tekst. Geef geen extra informatie."

LLM-generated incorrect text:

'role' : "system", "content":"Genereer medische tekst met grammaticale en spellingsfouten. De tekst beschrijft dagelijkse gebeurtenissen van de client onder zorg. Maak zeer diverse tekst. Bijvoorbeeld: " + voorbeeld_zin + "Bijvoorbeeld: " + voorbeeld_zin2"

LLM-generated correct text:

'role' : "system", "content":"Genereer medische tekst. De tekst is kort en bondig geschreven en brengt de essentie over van relevante gebeurtenissen van de dag of dagdeel. Het gaat over een oudere client die zorg ontvangt. De tekst is geschreven door een zorgprofessional die geen moeilijke woorden gebruikt. Bijvoorbeeld: " + voorbeeld_zin + "Bijvoorbeeld: " + voorbeeld_zin2"

"role": "user", 'content' : 'genereer 1 tekst met de volgende woorden: '+ keywords