# Challenging Error Correction in Recognised Byzantine Greek

**John Pavlopoulos**[1,2], **Vasiliki Kougia**[3], **Esteban Garces Arias**[4], **Paraskevi Platanou**[5]
**Stepan Shabalin**[6], **Konstantina Liagkou**[6], **Emmanouil Papadatos**[6]
**Holger Essler**[6], **Jean-Baptiste Camps**[6], **Franz Fischer**[6]
[1] Department of Informatics, Athens University of Economics and Business, Greece
[2] Archimedes/Athena RC, Greece (annis@aueb.gr)
[3] Faculty of Computer Science, University of Vienna, Vienna, Austria
[4] Department of Statistics, LMU Munich, Munich Center for Machine Learning, Germany
[5] University of Athens, Greece
[6] Ca'Foscari University of Venice, Italy

## Abstract

Automatic correction of errors in Handwritten Text Recognition (HTR) output poses persistent challenges yet to be fully resolved. In this study, we introduce a shared task aimed at addressing this challenge, which attracted 271 submissions, yielding only a handful of promising approaches. This paper presents the datasets, the most effective methods, and an experimental analysis in error-correcting HTRed manuscripts and papyri in Byzantine Greek, the language that followed Classical and preceded Modern Greek. By using recognised and transcribed data from seven centuries, the two best-performing methods are compared, one based on a neural encoder-decoder architecture and the other based on engineered linguistic rules. We show that the recognition error rate can be reduced by both, up to 2.5 points at the level of characters and up to 15 at the level of words, while also elucidating their respective strengths and weaknesses.

## 1 Introduction

The digitisation of ancient texts plays a crucial role in both analysing ancient corpora and preserving cultural heritage. However, transcribing ancient handwritten text using optical character and text recognition methods remains a challenging task. Handwritten text recognition (HTR) concerns the conversion of scanned images of handwritten text into machine-readable text. In contrast to recently printed materials, the analysis of images containing handwritten documents presents more intricate difficulties, particularly when dealing with historical and premodern manuscripts. These challenges may result in recognised text containing numerous errors or, at times, a complete inability to recognise the text. This is especially true when there is a low availability of suitable training data for specific scripts, such as medieval scripts.

### 1.1 Motivation

Natural language processing (NLP) can assist with the task of detecting and correcting erroneous text. When errors come from human learners of well-resourced languages, the task is undoubtedly challenging, yet notable advancements have been documented in recent research (Bryant et al., 2017, 2022). In the case of low-resource languages, however, the task can be more difficult and expensive, posing an additional hurdle not only to experts but also to systems. An example is the correction of recognition errors in historical newspapers, where recognition error rates of 10% (Chiron et al., 2017) have been reported. In this study, we escalate the difficulty by concentrating on the task of rectifying recognition errors in handwritten text. These errors tend to pose a greater challenge compared to those in printed text, primarily owing to the diversity in letter shapes and the distinct scripts employed by scribes. Error correction algorithms are applicable to HTRed material, benefiting macro-analytical applications, such as collation (Perdiki, 2022). They also concern transcribed text, e.g. by proposing corrections arising, for instance, due to distraction or fatigue during the annotation process.

### 1.2 Background

The written language of the Byzantine manuscripts and papyri,[1] such as the ones we shared with the challenge (see Section 3.2.3), reflects the language of the Byzantine times, following classical Greek and preceding the modern Greek language. Within these texts, morphological categories such as the optative, the pluperfect, and the perfect have disappeared, while others, such as the dative case have gradually decreased. Infinitives and participles are still there in the texts, serving as remnants of the

[1]We refer to Byzantine Greek, also known as Medieval or Middle Greek.

classical tradition, prompting one to regard the language as a distinct variant, separate from modern Greek. There are several spelling conventions that deviate from the older orthographic rules while the ancient punctuation signs are still in use, albeit not always with the same function. A more detailed description of this language is available in Papaioannou (2021).

### 1.3 Contributions

We study the benefits of error-correcting HTRed Byzantine text from the 10th to the 16th cent. CE. To conduct our research, we utilised a collection of transcribed images of Byzantine papyri and manuscripts documented by Platanou et al. (2022). For recognition, we used Transkribus (Kahle et al., 2017) to train an HTR model on seven images, one per century, and we used the trained model to recognise approx. one hundred pages. By using the recognised and transcribed images,[2] we introduced and successfully ran a shared task, challenging systems to correct errors in HTRed material (Fig. 1). Here:

- we present an overview of this challenge, which attracted 271 submissions, discussing the timeline, the evaluation, and the task difficulty that was introduced by a recognition error rate that varied across centuries;
- we introduce and publicly release a machine-actionable dataset for the correction of errors in HTRed Byzantine text.[3] Additionally, we offer three other resources: a synthetic dataset for evaluating error correction algorithms, and two corpora created specifically for this challenge, which we also make publicly available;
- by benchmarking the two best approaches—one based on engineered linguistic rules and the other on deep learning (the developers are co-authors)—we demonstrate that both effectively reduce the recognition-error rate, also outlining and analysing the merits of each approach.

## 2 Related work

Most studies approach the task of post-correction by focusing on printed text and by employing encoder-decoder architectures (Chiron et al., 2017; Rigaud et al., 2019; Schaefer and Neudecker, 2020; Lyu et al., 2021). The underlying idea is to encode the recognized erroneous text and then decode it into the corrected text, frequently employing methods from machine translation (Nguyen et al., 2020; Amrhein and Clematide, 2018).

### 2.1 Error correction

Error-correcting recognised text is a common approach when working with printed text (Schulz and Kuhn, 2017), where techniques such as spell checking, edit distance from lexicons, and the output of a statistical machine translation (SMT) model (Koehn et al., 2007) have been employed. A language model (i.e., the SMT decoder) decides the most probable correction, and to prevent the false alteration of a correct word, the authors introduce an additional input feature to the decision module. This feature indicates whether a word was found in a corpus alongside the preceding or following word. More generally, SMT is preferred in error correction while neural machine translation (NMT) has been reported advantageous in error detection (Amrhein and Clematide, 2018). More recently, an encoder-decoder model has been used to correct recognised printed text (on a character level) from historical books in German (Lyu et al., 2021). All the aforementioned studies pertain to printed text, where a recognition error rate of 10% is deemed challenging (Chiron et al., 2017). While we also experiment with statistical and neural error correction methods, our primary focus is on handwritten text, where the error rate is often higher (Figure 4).

### 2.2 Error detection for error correction

Error detection benefits error correction (Pavlopoulos et al., 2023). In 2017, ICDAR organised a competition focused on post-correcting recognised output (Chiron et al., 2017). The competition used a dataset comprising 12 million characters of printed text in English and French, and consisted of two subtasks. The first concerned error detection, aiming at the accurate identification of the position and the length of the errors. The second concerned error correction, where the errors were already provided to the participants (Chiron et al., 2017; Rigaud et al., 2019). The organisers noted 35 registrations, indicating a substantial interest from the community. However, it was also noted that only half of the submissions were deemed successful, underscoring the challenging nature of the task.

[2] To distinguish between the two, we will refer to 'transcribed' when the text is generated by a human expert and to 'recognised' when it is generated by a system.

[3] https://github.com/htrec-gr/challenge.

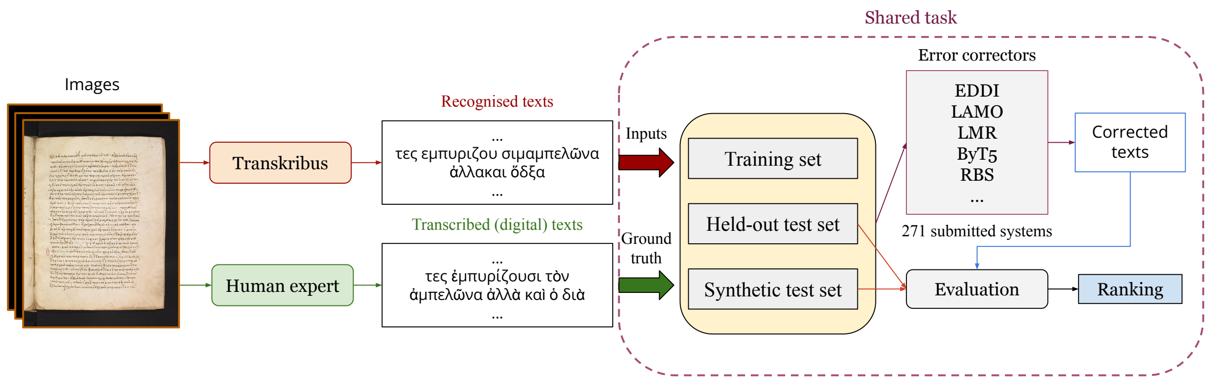

Figure 1: Overview of the organised shared task (details hidden to preserve anonymity)

In 2019, the competition was repeated, and the dataset's size was doubled, with the introduction of ten European languages (Rigaud et al., 2019). The texts used in both competitions were sourced from collections of national libraries or universities and encompassed a variety of formats, such as newspapers, historical books, and shopping receipts. In the 2017 edition, the most effective error correction method consisted of an ensemble combining statistical and neural machine translation models. In contrast, in the 2019 competition a character-level neural encoder-decoder took the top position, based on BiLSTM (Hochreiter and Schmidhuber, 1997) and BERT (Devlin et al., 2018).

BERT, fine-tuned on a named entity recognition task, was also used to perform error detection at the token level (Nguyen et al., 2020). After the subtoken tokenisation, the authors obtained GloVe or fastText word embeddings; combined with segment and positional embeddings, these were given as input to BERT. The hidden states were fed to a dense layer on top that classified each token as erroneous or not. Error correction, then, followed with a character-based NMT model. Error detection has been considered a reasonable first step to avoid the false alteration of already correctly recognised lines (Schaefer and Neudecker, 2020). The authors used a recurrent neural network (RNN) as a first step to detect erroneous characters in the recognised printed text. Then, a neural encoder-decoder translation model was fed only with sentences that comprised (detected) erroneous characters. Their two-step post-correction resulted in an 18.2% relative improvement in the recognition error rate.

## 3 The Shared Task

We used a dataset (§3.2) to set up a shared task on error-correcting the HTR output of Byzantine papyri and manuscripts. The challenge lasted from May 1st to July 1st, 2022, counting one hundred thirty-six registered participants from around the world,[4] and 271 submissions.

### 3.1 The language

We used images from Byzantine papyri and manuscripts from seven centuries (10th-16th c. CE). As was discussed already (§1.2), the written text reflects the language of the Byzantine times, a language during an intermediary phase of linguistic evolution between Classical and Modern Greek.

We employed the Handwritten Paleographic Greek Text Recognition (HPGTR) dataset (Platanou et al., 2022), comprising images from the digitised Barocci manuscript collection of the Bodleian Library that display text dating back from 10th to 17th c. CE. The scripts found in the respective manuscripts are the Greek minuscule script and the cursive style of the minuscule script, an example of which is shown in Figure 2(a). As shown in Figure 2(b), characters may join each other, disallowing empty space between words and leading to joined words that often characterise the cursive style. Also, joined characters can form ligatures, as shown in Figure 2(c), while the character position is not strict, as is shown with the character 'α' at the end of the word 'τάλαινα' in Figure 2(d).

Figure 2 also shows that lowercase and uppercase letters appear interchangeably in the text. Scriptura continua exists (not consistently) along

[4]India had the most participants (26), followed by the United States (7), Russia (6), Greece (6), and Japan (3).

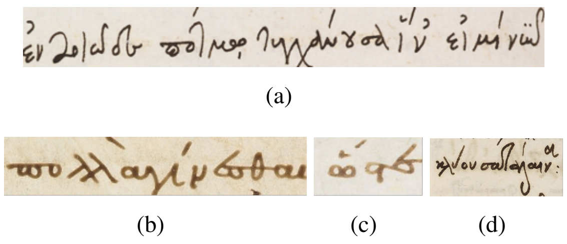

Figure 2: Visual examples of the language in the HPGTR dataset. An example of the cursive style of the minuscule script (a). The words ‘πολλά’ and ‘γί-νεσθαι’ are joined, leaving no empty space between them (b). In the word ‘ώστε’, the characters ‘σ’, ‘τ’ and ‘ε’ are combined to form the ligature ‘στε’ (c). The words ‘κλύουσα’ and ‘τάλαινα’ are shown in (d), with the final ‘α’ written above the latter.

with abbreviations. Furthermore, characters of various sizes may appear regardless of their neighbouring ones, such as in Figure 2(d) where the bigger letter ‘T’ is written between two small letters ‘α’.

### 3.2 The dataset

The dataset of the challenge comprises texts that are recognised (HTRed) and transcribed, with the latter serving as ground truth (hidden during evaluation).

#### 3.2.1 The HTR model

To recognise text from images of handwritten Byzantine papyri and manuscripts, we opted for Transkribus (Kahle et al., 2017).[5] This is an industrial platform that encompasses a wide range of functions (e.g., layout analysis, transcription, HTR training/prediction). To yield a rich material for our task and, hence, a diversity of recognition errors, we trained our model only on seven randomly-selected images (and transcriptions) from the HPGTR dataset, one per century. The centuries from 11th to 13th are better supported when counting words compared to the next three centuries, with the 16th being the least supported.

#### 3.2.2 Training data

We used the lines from ninety-eight HPGTR images. Each was transcribed by both a human expert, yielding the ground truth, and by our HTR network, yielding the input (see Fig. 1). To ensure a balanced representation across centuries, we randomly selected ten images per century (from the 10th to the 16th c. CE). However, the images from the 16th century contained fewer lines compared to other centuries, which we addressed by including additional images from that period. Overall, the training dataset comprises a (parallel) corpus of 1,800 lines (see also Table 5 in Appendix A).

[5]https://readcoop.eu/transkribus, Version 1.15.1.

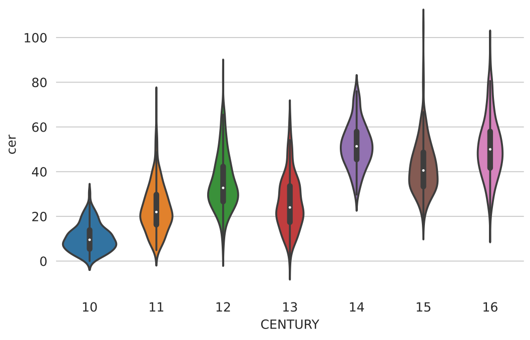

Figure 3: CER of recognised lines per century

**CER per century** When we group the character error rate (CER) by century, we notice that the rate tends to be higher for lines originating from the three most recent centuries (Fig. 3). This trend is consistent with findings from recognition systems trained on larger datasets (Platanou et al., 2022). Here, however, it is worth noting that lines with low CER present a more manageable correction task, whereas those with high CER pose greater challenges for parsing and correction.

**HTR error analysis** A common error in lines with a low CER is mistaken word division (i.e., space mistakenly added, e.g., by pushing away the final “s” of a word) and merging. Figure 4 shows that approx. 200 lines have a CER that is lower than 10% (fifty of which have less than 5%), while 500 have less than 20%. Further, approximately 400 lines have a CER of 50% or higher.

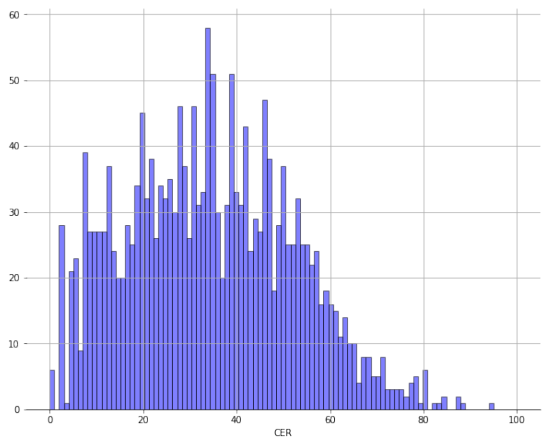

Figure 4: Number of recognised lines per CER

#### 3.2.3 Evaluation data

For evaluation purposes, a held-out test set was created, comprising 180 recognised lines but excluding their respective 180 transcriptions (hidden ground truth). These lines were taken from seven randomly selected HPGTR images, one per century, different from the ones used for creating the training set.

**Synthetic data** The test set comprised also synthetic recognised lines, designed by "attacking" human transcriptions of seven randomly selected images (153 lines), one per century (synthetic test set), as outlined in Table 5. Synthetic data had been shared also with the participants to serve validation purposes while avoiding overfitting the evaluation data. Our error-introducing attacks are based on five categories, shown in Table 1. We remove (I) or add (II) words in the text; add (III) or swap (IV) characters, and merge consecutive words (V).[6] Although different in nature when compared to data coming from an HTR system, we opt for this synthetic dataset to unlock a detailed error analysis.

| **Error type** | **Example** |
|---|---|
| I. *Remove randomly selected words* | this is a test > this is a __ |
| II. *Add random words at random positions* | this is a test > this is **word** a test |
| III. *Add random characters at random positions* | this is a test > this is a tes**k**t |
| IV. *Swap random characters* | this is a test > thi**i**s __s a test |
| V. *Merge random consecutive words* | this is a test > this is **atest** |

Table 1: Types of errors introduced (attacks) to yield the synthetic dataset. Instead of transcriptions, the same example sentence is shown to highlight the error types.

### 3.3 The evaluation metric

For evaluation we employ relative error reduction (ERr), which is applicable to CER and WER. We consider a (human) transcription $t_i^D$ for line $i$ in document $D$; the recognised text $r_i^D$ for the same line, and the corrected text $C(r_i^D)$, assuming the application of an error correction system. Then, assuming an error rate method $ER$ (e.g., CER), we define $ERr$ for line $i$ of $D$ as:

$$ERr(i, D) = ER(t_i^D, r_i^D) - ER(t_i^D, C(r_i^D)) \tag{1}$$

[6]The positions of the attacks are selected randomly.

A positive $ERr$ means that the error rate is reduced and that the applied correction (by $C$) yields a text that is closer to the human transcription. Negative values, on the other hand, mean that errors are introduced, increasing the edits needed to reach the transcribed text.

### 3.4 Leaderboard

A leaderboard was set up using the character error rate reduction (CERr) as the official evaluation metric but also reporting the word error rate reduction (WERr). The scores of the leaderboard were computed on the whole evaluation set, comprising system and synthetic transcriptions. The official ranking, however, ignores the synthetic transcriptions. We opted for adding instead of hiding data (i.e., using only a small part of the data for the leaderboard), for two reasons. First, synthetic errors provide valuable information regarding the generalisation ability of systems. Second, a small evaluation set is easier to overfit, which could yield a deceiving leaderboard.

## 4 Methods

For our error correction task, which aims to push the system transcription closer to the respective human transcription, we opted for three baselines (§4.1), which were shared with the participants of the challenge. Upon the evaluation of all the submissions, using the system and the synthetic transcriptions as input, we investigate further the two best-performing submitted approaches: one based on predefined rules and the other utilising a text-to-text Transformer.[7]

### 4.1 Baselines

We considered three baselines, which were based on edit distance (EDDI), a language model (LAMO), and linguistic rules (LMR).

**EDDI** replaces unknown words in the text by using the edit distance and a lexicon. Tokens that are not in the lexicon are replaced by the word in the lexicon with the lowest edit distance. As a lexicon, we use all the words of the training set.[8]

**LAMO** is similar to EDDI in that it uses a lexicon

[7]The developers of these two algorithms are co-authors of this paper. Other submissions were excluded due to their lower performance and a lack of accompanying system descriptions.

[8]The method returns the input text when the count of unknown words is larger than three, and only lexicon entries with low distance (lower than twenty-five) are considered for replacements. Thresholds are based on preliminary experiments.

to recognise unknown words in the text. However, word replacement is performed by a word-based statistical language model. We use a window of three words for the language model.
**LMR** is the third baseline, which is based on linguistic rules. Specifically, it focuses on the final "s" letter that is frequently the subject of wrong word division. Then, a character-based statistical language model decides whether it would be deleted (i.e., assuming it was mistakenly added) or merged with the previous word (a mistaken word division).

### 4.2 Deep learning with ByT5

ByT5 (Xue et al., 2022) is a byte-level pre-trained text-to-text Transformer (Raffel et al., 2020; Vaswani et al., 2017) that allows fine-tuning on various downstream tasks. For small model sizes, it outperforms MT5, which is the multilingual version of T5 (Xue et al., 2020) [9]. We fine-tuned the ByT5 "large" model variant by feeding it with recognised and transcribed texts, in order for it to learn to encode the former and decode the latter. We used a gradient accumulation of four steps, a standard cross-entropy loss, and the efficient Adafactor optimiser (Shazeer and Stern, 2018). At inference time, we used greedy decoding as it produced the best results. More details can be found in Appendix B.

### 4.3 Linguistic engineering with RBS

The rule-based correction system (RBS) is designed by making use of different rules, derived based on a qualitative analysis of what kind of errors typically occur in hand-written text recognition of Greek texts. These rules are described in more detail below (the algorithm is provided in Appendix C).
**Word subset (R1):** Any token comprising a word in a lexicon (formed by the transcriptions) is divided into two tokens with a white space.[10]
**Edit distance (R2):** Tokens that had an edit distance of one with (a) any possible valid alternation of the conjunction "και", and (b) a term in the lexicon (R1), are replaced with these two terms. For tokens of eight characters or more, not affected by this rule, we use an edit distance of two.
**Word bigrams (R3):** Recognition often produces white spaces at the wrong positions (e.g., "δικαι ονπερι" instead of "δικαιον περι"). To address such errors, any bigram in the text is merged (removing the white space) and passed to R1.
**Single-character tokens (R4):** Single-character tokens that weren't known articles are merged with the end of the previous token, if the merged token exists in the lexicon, and with the start of the next token otherwise.
**Duplicate characters (R5):** Tokens comprising two (or more) identical consecutive characters, and that are not present in the lexicon, are collapsed to a single character (e.g., "εεστιν" becomes "εστιν").
**Misspelled pronouns (R6):** Character order issues of pronouns are fixed by specific replacements. For example, "τνω" is replaced by "των".
**Joint pronouns (R7):** Pronouns merged with the next token (e.g., "τηνκαρδιαν") are searched and replaced by two words (e.g., the previous token would become "την καρδιαν").
**Main prepositions (R8):** Words beginning with specific prefixes (e.g., "ειντοις", "εντοις", "ηκτοις", "εκτης") can bypass the previous rules. Hence, a mapping is used to address such tokens.

## 5 Empirical analysis

### 5.1 Error rate reduction results

In Table 2, we present the ERr for characters (CERr) and words (WERr), achieved by error-correcting the HTR output or synthetic data. EDDI and LAMO display negative scores in both metrics on both input types. This means that such - rather simplistic - baselines introduce new errors instead of addressing existing ones. The third baseline, LMR, reduces slightly the CER and WER of the HTR output. The focus of this baseline is on a single letter (final "s"), which is a common recognition error, though not the only one. The attacks that are used to create the synthetic data, on the other hand, are applied to random text positions (see §4), none of which concerns this letter. Hence, no correction is made and both scores are zero.

BYT5 and RBS achieve a positive reduction in both metrics. RBS scores higher than LMR when the input is the HTR output. Also, it achieves a positive reduction when the input is synthetic (0.10 in CERr and 1.29 in WERr). Obviously, this method handles many error types, covering more than typical HTR mistakes. BYT5 is the best overall when applied to HTR output. It is more than five times better in terms of CER and more than eight times better in terms of WER compared to RBS. When evaluated on synthetic input, however, the error

[9] In preliminary experiments, MT5 performed considerably worse than ByT5.
[10] A more strict version of this rule uses a list of pronouns (e.g., αυτου) and conjunctions (e.g., και), testing if the token concatenates words from the two resources.

| | HTR OUTPUT | | SYNTHETIC | |
|---|---|---|---|---|
| | CERr ↑ | WERr ↑ | CERr ↑ | WERr ↑ |
| EDDI | -0.19 | -0.29 | -0.54 | -2.48 |
| LAMO | -5.88 | -0.80 | -5.95 | -3.13 |
| LMR | 0.02 | 0.06 | 0.00 | -0.00 |
| RBS | 0.44 | 1.82 | **0.10** | **1.29** |
| BYT5 | **2.53** | **14.97** | -7.72 | -23.14 |

Table 2: CERr and WERr scores of the baselines (top three rows), of the neural encoder-decoder (BYT5), and the rule-based error correction approach (RBS).

rates increase considerably, displaying a lower performance than RBS and all three baselines, most probably because the model is not trained on synthetic data. This is an indication that the synthetic data may not be very natural, and that rule-based systems are less useful in 'real-world' situations.

### 5.2 Inter-corrector agreement

In order to investigate closer the relationship between BYT5 and RBS, we compute the CER between the two corrected texts, one per system, of each recognised line. Low scores reflect a high agreement between the two approaches while high scores indicate very different outputs. By sorting the lines based on this score, we can assess the two approaches in different agreement zones. Figure 5 presents these results. Overall, BYT5 is more often above zero and bars are also much higher than RBS. When we look at the left of the diagram, there are almost no differences between the two in their performance, which is reasonable given that the two approaches agree (i.e., they will both be correct or they will both be wrong). As we move to the right, however, we can see that BYT5 achieves more and deeper negative bars. On the other hand, RBS follows a low-risk, low-gain strategy.

**Manual investigation** of the best and worst handled lines per method (Table 3) reveals that in the worst-case scenario per method (line 8 for RBS, hallucination in 15 for BYT5), the corrections of the other method were minimal (lines 7 and 16, resp.).

### 5.3 Sensitivity analysis on synthetic data

As was shown in Table 2, RBS achieves a positive CERr in the synthetic data while BYT5 underperforms in this setup. To explore the performance of the two methods further, we computed the mean CERr per attack type (Table 4). For all attack types,

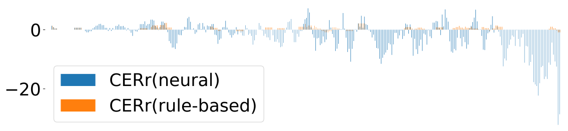

Figure 5: CERr (moving average for better readability with a window of size 5) of BYT5 and RBS per line. Lines have been sorted from the least (left) to the highest (right) agreement (CER) between the two.

BYT5 yields a negative average CERr, with its weakest performance observed when characters are added (Type III), and relatively better results when words are merged (Type V). On the other hand, RBS also struggles with two attack types, specifically when words are removed (Type I) and added (Type II). Its performance remains relatively consistent for the remaining three types of attack.

### 5.4 Enhanced HTR vs. post-correction

Enhancing HTR with more training data can allow a direct comparison between the performance gains from neural error correction and from increasing the HTR training data.

For the purposes of this experiment, we trained a new HTR model. To avoid the financial cost of train multiple instances, we opted for an open-source alternative to Transkribus. For the experiment described in §5.4, our HTR model achieved a similar performance with Transkribus on the same seven training pages. We release this model publicly at: `https://github.com/htrec-gr/htr`. The architecture of this HTR model is a Swin (Liu et al., 2021) encoder with a BERT-based decoder (Devlin et al., 2019). For the experiment we used a single GPU card, i.e., NVIDIA Tesla V100 (16GB), and the model had 142 million parameters. It was trained for 75 epochs (12 hours). We used as seed 42, batch size of 48, AdamW optimizer, and Transformers V4.25.1. For language generation, we used a max-length of 200 characters, early stopping and a greedy decoding strategy.

Figure 6 shows the CERr and WERr as we transition from 7 pages of training data (our baseline) to 70 pages overall. When training with 28 (+21) pages, CERr goes up to 9.73 and WERr to 12.62. This means that the WERr of BYT5 (correcting the errors of a 7-page-trained HTR model) is better by two points (14.97; Table 2). When training with more pages (e.g., 70), however, CERr and WERr reach up to 15.72 and 27.15 respectively, outperforming the gains from error correction. It is worth

| | | Transcribed, Recognised, or Corrected line | CERr | |
|---|---|---|---|---|
| | Human: | σωματος κρειττων τοσουτον των χρημασι βοη | | |
| | HTRed: | ωματος κρειτγων τοσουτον των χρημασιβοη | | |
| | BYT5: | εωματος κρειται το σουτον των χρημασι βοη | -4.88 | |
| B | RBS: | σωματος κρειτγων τοσουτον των χρημασι βοη | | 4.88 |
| | Human: | λεντιον διεζωσεν εαυτον | | |
| | HTRed: | λεντιον διεζωσενεαυτόν | | |
| | BYT5: | λεντιον διεζωσεν εαυτον | 0.00 | |
| W | RBS: | λεντιον διεζωσενε αυτόν | | -4.37 |
| | Human: | ψεκτως ποιουσιν εαυτους σαρκα | | |
| | HTRed: | ψκε κτῶς ποι ουσιν εαύτοις σάλρ κα | | |
| B | BYT5: | ψκεκτως ποιουσιν εαυτοις σαλοκα | 17.24 | |
| | RBS: | ψκε κτῶς ποι ουσιν εαύτοις σάλρ κα | | 0.00 |
| | Human: | ῶν δὲ συνεπινοουμενην έχων τῆ ὑπαρξει | | |
| | HTRed: | ῶῶν δὲ συνεπινοουμενην δέχων τῆ ὑπαΈρξει | | |
| W | BYT5: | συνεπινοουμενην συνεπινοουμενην συνεπινοουμενη... | -108 | |
| | RBS: | ῶῶν δὲ συνεπινοουμενην δέχων τῆ ὑπαΈρξει | | 0.00 |

Table 3: Error analysis by focusing on the best (B) and worst (W) correction per method based on the achieved CERr. The first two rows per quadruplet show the respective transcription and recognition.

| | Type of attack | BYT5 | RBS |
|---|---|---|---|
| I. | Remove words | -6.36 | 0.00 |
| II. | Add words | -7.95 | -0.06 |
| III. | Add characters | -12.28 | 0.18 |
| IV. | Swap characters | -8.42 | 0.21 |
| V. | Merge words | -3.54 | 0.18 |

Table 4: Average CERr per attack type.

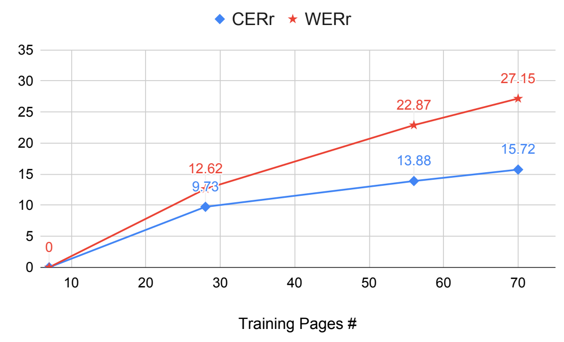

Figure 6: CERr and WERr scores (vertically) when the HTR model is trained on more pages (horizontally).

noting that this improvement requires a substantial increase of HTR training material, which may not be available (e.g., lack of images or transcriptions), making error correction a promising alternative.

## 6 Discussion

The challenge in error-correcting the HTR output of Byzantine manuscripts and papyri has attracted a significant number of registrations and submissions (§3.2.3), the best of which were discussed in this work. Characteristics of Byzantine Greek and the respective scripts have been discussed in §3.1, in order to highlight the difficulties that recognition and error-correction algorithms need to tackle. The variety of scripts and scribes in this language, along with its evolution, is likely to have caused a varying recognition error rate over time (Figure 3). This error rate variety poses a significant challenge to post-correction methods, which should be able to handle lines that comprise from few to many errors (different types).

When assessing error correction in recognized printed and handwritten material, it's crucial to consider the error rate. As detailed in §2, prior studies have predominantly focused on printed material, characterised by relatively low recognition error rates. However, our findings illustrate a significant variation in the error rate for HTR output, encompassing both accurate recognitions and those with numerous errors (Fig. 4).

We also show that a rule-based approach outperforms the baselines (Table 2), or even a neural encoder-decoder in the case of synthetic data (Table 4). Therefore, error-correcting the HTR output can also be seen as a knowledge-intensive NLP task, for which knowledge-based approaches can be successful (Lewis et al., 2020).

The experimental results presented in Table 2, show that post-correcting the HTR output for Byzantine Greek can reduce the error rate by approximately 2.5 units at the character and 15 units at the word level. This means that error correction can be employed during the recognition of the text in the images of Byzantine manuscripts and papyri, to facilitate human experts with the tedious semi-automated transcription task (i.e., correcting the HTR output). This gain is recorded by post-correcting errors, but the encoding-decoding of BYT5 could possibly be integrated also into the HTR pipeline, incorporated as one of the tasks in a multitask approach (i.e., image to text to text).

## 7 Conclusions

We presented a challenge of error-correcting HTR output for Byzantine Greek, publicly releasing data with both synthetic and actual HTR errors. A pre-trained BYT5 encoder-decoder model, fine-tuned on recognised (input; encoded) and transcribed (output; decoded) texts, achieves a notably high performance, effectively reducing errors. A comparable reduction of errors could have been achieved if the HTR model had been trained on approximately 30 additional pages. However, generalisation remains a concern, as evidenced by the model's performance on synthetic data, where errors were introduced instead of corrected. A rule-based approach, on the other hand, showed promise by well performing on synthetic data but not on real-world data. Future work will focus on challenging error-correction systems based on HTR models trained on data from specific centuries, aiming to address the diverse range of errors encountered.

## Acknowledgements

This work has been partially supported by project MIS 5154714 of the National Recovery and Resilience Plan Greece 2.0 funded by the European Union under the NextGenerationEU Program.

## Limitations

- As observed in the results, the performance of the systems varies significantly across centuries, suggesting that century-specific factors need to be considered when designing effective error-correcting systems.

- It's evident that post-correction is often hindered by the low quality of the HTR output. Therefore, there is a need for more advanced approaches that incorporate error detection (Pavlopoulos et al., 2023) and correction before the output is generated, possibly in conjunction with a post-correction module.

- While the results demonstrate the potential of error-correcting systems for some Byzantine Greek corpora, the generalisation potential in the context of low-resource data remains to be explored. This can be achieved by extending this approach to additional corpora and other languages, allowing for a more comprehensive understanding of its effectiveness across different linguistic domains. Still, we hope that this study will be beneficial for the development of new error-correction strategies aimed at improving the quality of recognitions, especially in scenarios with limited data availability.

## Ethics Statement

This work involves the use of publicly available datasets and does not involve human subjects or any personally identifiable information. We declare that we have no conflicts of interest that could potentially influence the outcomes, interpretations, or conclusions of this research. All funding sources supporting this study are acknowledged in the acknowledgments section (hidden to preserve anonymity). We have made our best effort to document our methodology, experiments, and results accurately and are committed to sharing our code, data, and other relevant resources to foster reproducibility and further advancements in research.

## References

Chantal Amrhein and Simon Clematide. 2018. Supervised ocr error detection and correction using statistical and neural machine translation methods. *Journal for Language Technology and Computational Linguistics (JLCL)*, 33(1):49–76.

Christopher Bryant, Mariano Felice, and Edward Briscoe. 2017. Automatic annotation and evaluation of error types for grammatical error correction. Association for Computational Linguistics.

Christopher Bryant, Zheng Yuan, Muhammad Reza Qorib, Hannan Cao, Hwee Tou Ng, and Ted Briscoe. 2022. Grammatical error correction: A survey of the state of the art. *arXiv preprint arXiv:2211.05166*.

Guillaume Chiron, Antoine Doucet, Mickaël Coustaty, and Jean-Philippe Moreux. 2017. Icdar2017 competition on post-ocr text correction. In *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, volume 1, pages 1423–1428, Kyoto, Japan.

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

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

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Philip Kahle, Sebastian Colutto, Günter Hackl, and Günter Mühlberger. 2017. Transkribus-a service platform for transcription, recognition and retrieval of historical documents. In *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, volume 4, pages 19–24. IEEE.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, et al. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th annual meeting of the association for computational linguistics companion volume proceedings of the demo and poster sessions*, pages 177–180, Prague, Czech Republic.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474.

Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. 2021. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*.

Lijun Lyu, Maria Koutraki, Martin Krickl, and Besnik Fetahu. 2021. Neural ocr post-hoc correction of historical corpora. *Transactions of the Association for Computational Linguistics*, pages 479–493.

Thi Tuyet Hai Nguyen, Adam Jatowt, Nhu-Van Nguyen, Mickael Coustaty, and Antoine Doucet. 2020. Neural machine translation with bert for post-ocr error detection and correction. In *Proceedings of the ACM/IEEE joint conference on digital libraries in 2020*, pages 333–336.

Stratis Papaioannou. 2021. *The Oxford Handbook of Byzantine Literature*. Oxford University Press.

John Pavlopoulos, Vasiliki Kougia, Paraskevi Platanou, and Holger Essler. 2023. Detecting erroneously recognized handwritten byzantine text. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 7818–7828, Singapore. Association for Computational Linguistics.

Elpida Perdiki. 2022. How to (auto) collate big manuscript data with minimal htr training. *Journal of Data Mining and Digital Humanities*.

Paraskevi Platanou, John Pavlopoulos, and Georgios Papaioannou. 2022. Handwritten paleographic greek text recognition: A century-based approach. In *Proceedings of the Language Resources and Evaluation Conference*, pages 6585–6589, Marseille, France. European Language Resources Association.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, Peter J Liu, et al. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(140):1–67.

Christophe Rigaud, Antoine Doucet, Mickaël Coustaty, and Jean-Philippe Moreux. 2019. Icdar 2019 competition on post-ocr text correction. In *2019 international conference on document analysis and recognition (ICDAR)*, pages 1588–1593, Sydney, Australia.

Robin Schaefer and Clemens Neudecker. 2020. A two-step approach for automatic ocr post-correction. In *Proceedings of the The 4th Joint SIGHUM Workshop on Computational Linguistics for Cultural Heritage, Social Sciences, Humanities and Literature*, pages 52–57, held online.

Sarah Schulz and Jonas Kuhn. 2017. Multi-modular domain-tailored ocr post-correction. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2716–2726, Copenhagen, Denmark.

Noam Shazeer and Mitchell Stern. 2018. Adafactor: Adaptive learning rates with sublinear memory cost. *CoRR*, abs/1804.04235.

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

Linting Xue, Aditya Barua, Noah Constant, Rami Al-Rfou, Sharan Narang, Mihir Kale, Adam Roberts, and Colin Raffel. 2022. Byt5: Towards a token-free future with pre-trained byte-to-byte models. *Transactions of the Association for Computational Linguistics*, 10:291–306.

Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. 2020. mt5: A massively multilingual pre-trained text-to-text transformer. *arXiv preprint arXiv:2010.11934*.

## A Dataset configuration

As is shown in Table 5, we compiled a parallel corpus of 1,800 lines for training purposes. Each line comprises a transcription (ground truth) and a recognition, resulted from an under-trained HTR model (trained on the transcriptions of seven held-out pages). Evaluation was performed on synthetic (153 lines) and actual (180 lines) data, resulting to a parallel corpus of 2,133 lines, overall, which we publicly release, along with the HTR model that we used to produce the recognitions.

| Purpose | # Pages | # Lines | Dataset |
|---|---|---|---|
| Training | 98 | 1800 | HPGTR |
| Evaluation | 8 | 180 | HPGTR |
| Evaluation | 7 | 153 | Synthetic |
| Total | 113 | 2,133 | — |

Table 5: Data configuration for the challenge. Each page comprises several lines (texts) and each line has been transcribed and recognised. The transcription of lines used for evaluation was kept hidden from the participants during the testing phase.

## B ByT5

We (i.e., a participant at the time of the challenge) opted for a batch size of 1 (i.e., a single line) and a learning rate of 1e-4. Optimum performance was achieved at one and a half epochs. As is shown in Table 6, BYT5 was trained for more epochs but results deteriorated.

Table 6: CERr and WERr of ByT5 when it was trained for more epochs.

| | HTR OUTPUT | | SYNTHETIC | |
|---|---|---|---|---|
| EPOCHS | CERr ↑ | WERr ↑ | CERr ↑ | WERr ↑ |
| 1.5 | 2.53 | 14.97 | -7.72 | -23.14 |
| 3 | -2.91 | 8.62 | -15.41 | -36.05 |
| 12 | -8.45 | 6.08 | -18.85 | -43.40 |

## C RBS

Algorithm 1 presents the pseudocode for RBS. A rule based system, however, is only as good as the corpus size it has access to. We hypothesize that the system's performance would improve with a bigger corpus. To that end, we provide over 100 books of text in ancient Greek [11] and Byzantine [12], scraped from various online sources. Due to time constraints, these were not utilized by RBS, a task that will be explored in future work. The biggest collection, titled 'Σύνοψις Ιστοριών' from I.Skylitzis totals 5 books, 153,709 words and 885,259 characters [13]. Furthermore, we provide a lexicon [14] of over 42,107 ancient Greek words independent of the collection of books, which was also not utilized by RBS.

[11] http://users.uoa.gr/~nektar/history/tributes/ancient_authors/index.htm

[12] https://byzantium.gr/keimena/keimena.php

[13] https://wordcounter.tools/

[14] https://www.greek-language.gr/digitalResources/ancient_greek/tools/liddel-scott/search.html?start=20&lq=

**Algorithm 1** Rule-Based System (RBS)

```
Require: $corpus \leftarrow list(words) \geq 0$
  for $sent \leftarrow example\_system\_transcr$ do
      $sent \leftarrow drop\_duplicate\_char(sent)$
      for $token \leftarrow sent$ do
          for $gold \leftarrow corpus\_1$ do
              if $token$ in $gold$ then
                  $gold, subtoken \leftarrow split\_token(token)$
                  $sent \leftarrow replace\_token\_in\_sentence(token, [gold, subtoken])$
              end if
          end for
          $list[(gold_1, gold_2)] \leftarrow create\_pairs(corpus)$
          for $pair \leftarrow list[(gold_1, gold_2)]$ do
              $combination \leftarrow pair[0] + pair[1]$
              if $token$ in $combination$ then
                  $gold_1, gold_2 \leftarrow split\_combination(token)$
                  $sent \leftarrow replace\_token\_in\_sentence(token, [gold_1, gold_2])$
              end if
          end for
          $token \leftarrow replace\_freq\_tokens(token)$
          $list\_and \leftarrow$ ['ϰαι', 'ϰαὶ', 'ϰαί']
          for $gold \leftarrow corpus + list\_and$ do
              if $edit\_distance(gold, token) == 1$ and (token not in $list\_and$) then
                  if gold in $list\_and$) then
                      if gold not in $(begin/end\_of\_the\_sentence)$ then
                          $token \leftarrow gold$
                      end if
                  else if $N$ is odd then
                      $token \leftarrow gold$
                  end if
              end if
              if $edit\_distance(gold, token) == 2$ and $length(token) \geq 8$ then
                  $token \leftarrow gold$
              end if
          end for
          $list\_articles \leftarrow$ ['τὴν', 'ϰατα', 'τὰ', 'τῶν']
          if token in $list\_articles$ then
              if position(token,gold) in $begin\_or\_end\_of\_token$ then
                  $gold, subtoken \leftarrow split\_article(token)$
                  $sent \leftarrow replace\_token\_in\_sentence(token, [gold, subtoken])$
              end if
          end if
          if length(token)==1 then
              $sent \leftarrow drop\_token(token)$
          end if
          for $i \leftarrow range(0, len(sent\_tokens) - 1)$ do # R3
              $w1, w2 \leftarrow sent\_tokens[i], sent\_tokens[i + 1]$
              $bigram = w1 + w2$ # no white space between the consecutive words
              for $g \leftarrow corpus$ do # for each gold word in the corpus
                  if $edit\_distance(g, bigram) == 1$ & $w1$ not in {'ο','η','το','τα'} then
                      token ←g+' '+w2
                  end if
              end for
          end for
      end for
  end for
```