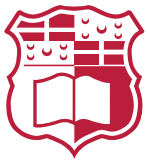
Spell Checking for the Maltese Language

Alana Busuttil

Supervisor: Dr. Claudia Borg

September 2024

Submitted in partial fulfilment of the requirements for the degree of Master of Science by Research.



L-Università ta' Malta
Faculty of Information & Communication Technology

Abstract

This study presents the development of a Grammar Error Correction (GEC) system for the Maltese language. A GEC system, or spell checking system, improves writing quality by identifying and correcting spelling and grammar errors in text. Modern spell checkers are able to improve writing across various contexts, ranging from casual text messages to formal documents. As a low-resourced and underrepresented language in the digital world, Maltese lacks a robust digital presence, highlighting the urgent need for a dedicated spell-checking system. This research seeks to contribute to the development of a spell checker for the Maltese language.

A key issue identified through previous efforts for GEC systems for Maltese is the lack of data available. Therefore, the creation of a larger, more representative dataset was necessary. A data collection campaign was launched to gather authentic human errors. The errors collected were statistically analysed, and used to inform the creation of a synthetic dataset. As a result, two distinct datasets—containing authentic human errors, synthetic errors, and a hybrid of both—were developed and used to train the system. The created system consisted of a transformer based implementation, in which pre-trained Maltese language models were implemented for both the encoder and decoder components. The final system outperformed previous spell-checking systems, setting a new benchmark in Maltese GEC.

The final system created consistently corrected errors related to capitalisation and Maltese-specific characters, indicating a strong level of contextual understanding. Despite these advancements, the system's overall performance remains below that of widely used commercial spell checkers. Nonetheless, the resources created and the findings of this study provide a foundation for future research in Maltese GEC.

Acknowledgements

I would like to take this opportunity to express my deepest gratitude to my supervisor, Dr. Claudia Borg. Without her inspiring confidence, I would achieve only a shadow of what I am truly capable of. In the words of Fyodor Dostoevsky, “I want to talk about everything with at least one person as I talk about things with myself,” I am fortunate enough to be surrounded by community of people with whom this is possible. My heartfelt thanks go to my mother, father, sister, closest friends, and my partner, Sean, whose very presence is a pillar of strength in my life.
Thank you all.

Contents

List of Figures

List of Tables

1 Introduction

Natural Language Processing (NLP) is a branch of Artificial Intelligence (AI) focused on the interaction between computers and human languages. NLP systems enable computers to interpret and respond to human language in a way that is meaningful and useful. Key tasks within NLP include Named Entity Recognition (NER), Sentiment Analysis, Machine Translation, Speech Recognition, Text-to-Speech, and Grammar Error Correction (GEC).

GEC involves two main sub-tasks: error identification and error correction [1]. In this process, a model takes in text with errors (the source) and predicts a corrected version (the output). This corrected text is then compared against one or more reference sentences (targets). The possible presence of multiple target sentences highlights the complexity and nuance involved in spell-checking systems, which must consider various aspects of correction, including capitalisation, syntax, punctuation, grammar, and context. The broad nature of error correction has led some to propose that the term "Language Error Correction" might be more fitting than "Grammar Error Correction" [1]. Nonetheless, the field continues to be referred to as GEC, a term that is likely rooted in earlier approaches, which only corrected select errors [2].

A notable challenge in GEC is its inherent low-resourced nature [1]. Compared to other NLP tasks like NER, there is significantly less GEC corpora available, furthermore, the use of synthetic data is not always representative of authentic errors, often having declining effects on model performance [3]. While GEC draws on methods and materials from Machine Translation—where the source text is "translated" into the target text [1], this can be both advantageous and problematic. While such borrowing provides useful resources and techniques, it can also lead to over-corrections or false positives [4]. Other approaches include the use of different systems for detection and correction [5] or rule-based approaches [2]. The challenges identified are further pronounced for low-resource languages, like Maltese.

Research on GEC for Maltese is sparse, with only a few studies addressing this area [6] and just one employing neural methods [4]. These studies commonly highlight the lack of authentic error corpora and its detrimental effect on model performance. Furthermore, Maltese is a complex language, consisting of morphological systems influenced by both Semantic and Romantic languages. The morphological complexity of the language makes the options of rule-based and statistical approaches less viable, since they will not be able to cover all aspects of the language [4] [6] [7].

1.1 Problem Definition and Motivation

Maltese is classified as a low-resourced language, having limited technological support. This is confirmed by a Meta-Net white paper series, which highlights the challenges Maltese faces in terms of technological tools and resources [8]. A graph from the European Language Grid [1], presented in figure 1.1, illustrates this disparity, comparing technological factors across the official EU languages. Maltese, with a score of 4,861, ranks significantly lower than higher-resourced languages such as English, which scores 78,413; highlighting the need for increased technological investment and development for Maltese. Despite this, a recent study by Marmara' [9] concludes that a total of 95.4% of the representative population consider Maltese their first language, therefore, although lacking in online representation and resources, the Maltese language is still thriving.

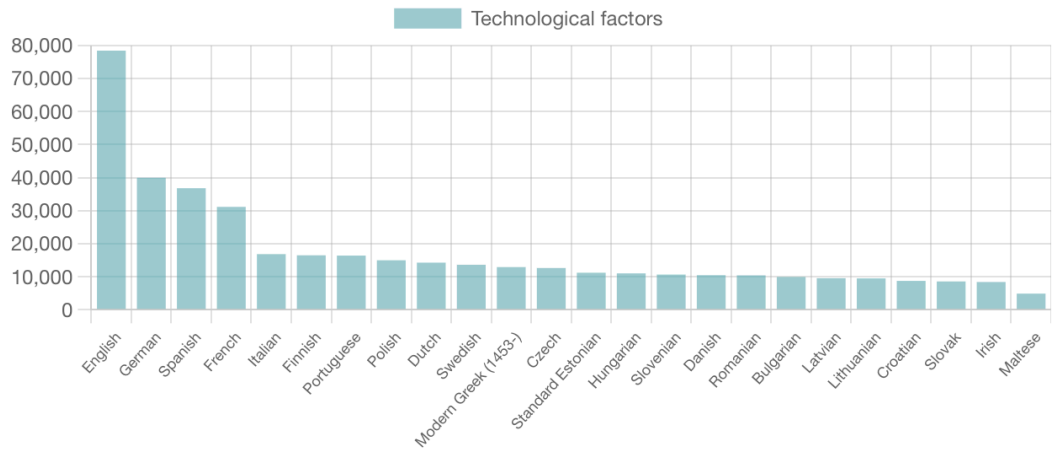


Figure 1.1 Technological Tools for Official EU Languages.

Furthermore, there is a clear demand for better online resources, including a spell checker, with 94% of the representative population expressing a need for this tool [9]. However, previous efforts in Maltese GEC have been limited, consisting of an online rule-based approach [2], a statistical approach [6], and a more recent neural approach [4]. These approaches have not yet achieved the performance levels seen in high-resource languages, namely due to the challenges posed by limited data and resources.

This study seeks to address these gaps by focusing on enhancing the technological infrastructure for Maltese, particularly in spell checking. It aims to build on previous approaches by leveraging existing resources, such as Maltese pre-trained BERT models [10], the MLRS corpus [**MLRS**], and Mozilla's CommonVoice dataset

[1] European Language Grid: https://live.european-language-grid.eu
[2] Maltese online rule-based checker: https://spelling.mt/index.html

[3], as well as creating better resources to support future research. By integrating these resources and improving upon earlier methods, this research aims to advance spell-checking technology for Maltese, thereby contributing to the languages overall technological development.

1.2 Aims and Objectives

The aim of this research is to develop a functional GEC system for the Maltese language, using techniques suitable for low-resource environments. The objectives are based on the limitations presented in Debattista [4]'s attempt at GEC for the Maltese language. Debattista [4]'s main limitation was the quality and availability of data, with two key issues identified:

- The source data from the MLRS corpus contained numerous grammatical errors and repetitive text, requiring manual cleaning.
- The final corpus did not adequately represent real-world spelling errors; there was insufficient statistical data to generate a realistic synthetic corpus.

Building upon these, the objectives identified were as follows:

1. **The First Objective: Collect a diverse set of authentic human errors to construct a more representative corpus that accurately reflects the range of errors found in real-world typing. Develop a statistically informed synthetic dataset, based on the authentic human dataset.**

   The aim of this objective is to create a collection of corpora that represent everyday spelling and grammatical errors for training a GEC system. This will be achieved through setting up a data-collection campaign in order to collect as many authentic human errors as possible, resulting in a human error corpus. Following this, a statistical analysis will be performed on the collected data, which informs the creation of a synthetic corpus consisting of a source and target. Finally, a mixed corpus will be created from the synthetic and human data. Furthermore, the target corpus will be checked for errors and cleaned accordingly.

[3] Mozilla CommonVoice for Maltese: https://commonvoice.mozilla.org/en

2. **The Second Objective: Replicate existing work to serve as a baseline for Maltese GEC.**

   The aim of this objective is to build on insights from prior GEC systems to inform the development of a more effective approach for Maltese. This includes replicating previous neural attempts at GEC systems for the Maltese language [4] to serve as a baseline.

3. **The Third Objective: Develop an approach for the Maltese language, leveraging the capabilities of pre-trained Maltese models. Train and fine-tune the created GEC system for the Maltese language using the newly created corpora and employ effective techniques tailored for low-resource environments.**

   The aim of this objective is to develop an optimised approach for GEC tailored to the Maltese language. This will be achieved through creating a system that leverages pre-trained Maltese models and training it with the corpora compiled through the first objective. The optimal system will then be selected and further extensive fine-tuning will be performed.

4. **The Fourth Objective: Evaluate previous systems for Maltese GEC. Conduct a comprehensive comparison of the final approach with previous systems.**

   This aim of this objective is to deepen understanding of the model's performance relative to previous neural GEC efforts for the Maltese language [4]. A comparative evaluation of the final approach and Debattista [4]'s method will be conducted, focusing on datasets, model performance, and overall effectiveness.

1.3 Proposed Approach

To achieve the objectives defined in this research, we reviewed both previous approaches for the Maltese language, as well as state-of-the-art methods for GEC. Our findings reveal that most high-performing GEC systems rely on encoder-decoder models, implemented through ensemble, pipeline, or hybrid systems [11] (these approaches are differentiated in more detail in section 2.6). The system combinations often integrate neural and statistical models to enhance performance. Additionally, combining pre-trained models, such as BERT—an 'encoder-only' transformer architecture—has proven particularly advantageous in low-resource environments, as it enhances the model's language understanding [12] [1]. Building on these insights, the proposed approach combines these strategies, focusing on boosting

performance while overcoming the challenges posed by limited resources. The key steps in the system development are outlined as follows:

1. Develop a data collection interface to gather a broad spectrum of authentic human errors, ensuring the dataset accurately represents common mistakes.
2. Conduct statistical analyses on the collected data to identify and categorise prevalent errors, providing a basis for creating a representative synthetic error corpus.
3. Create a statistically informed source-target error corpus that reflects authentic human errors, which will serve as the foundation for training.
4. Integrate pre-trained Maltese BERT models at the core of the encoder-decoder system, leveraging their language understanding capabilities.
5. Train the system using the source-target error corpus to learn from real-world representative data and improve error correction accuracy.
6. Fine-tune the system to optimise its performance, ensuring it effectively corrects grammatical errors in Maltese.
7. Compare the final fine-tuned approach with Debattista [4]'s previous attempt at GEC for the Maltese language in order to identify the attempted approach's strengths and short-comings.

1.4 Document Structure

The presented dissertation is structured as follows: the introduction in chapter 1 outlines the research problem identifies the aims and objectives. A focus on the challenges of developing GEC systems for the Maltese language is highlighted, and is further defined in the background and literature review in chapter 2, along with key concepts and approaches related to GEC, an overview of rule-based, statistical and neural approaches, performance-enhancing techniques, techniques for synthetic dataset creation and existing GEC efforts for the Maltese language. The background and literature review is followed by the methodology in chapter 3. The methodology presents the development process for the dataset, such as the data collection campaign, the statistical analysis performed, and the creation of the synthetic corpus. Furthermore, the development of the GEC system is detailed, including the inclusion of pre-trained Maltese models, and system fine-tuning. The model performance is discussed in the results and discussion in chapter 4. The results and discussion presents different model variations trained on the different

corpora in order to determine the highest performing system and corpus. The final system is further compared to previous neural GEC approaches, highlighting the improvements and downfalls of the created system comparatively. Finally, the conclusion and future work in chapter 5 revisits the defined objectives, identifies the gaps in the research and suggests potential future improvements.

2 Background and Literature Review

The purpose of this chapter is to gain insight into previous attempts at GEC, and to identify potential techniques to build a successful GEC model. This chapter is organised as follows: section 2.1 defines the Maltese language and identifies the need for a spell checker, and section 2.2 provides a formal definition of GEC. Section 2.3 outlines the main models used for GEC, as well as pre-trained models relevant to the Maltese language, section 2.4 defines a chronological account of popular shared tasks for GEC, and their highest scoring approaches, and section 2.4.1 defines benchmark corpora used, including the official corpora used in shared tasks. Section 2.5 defines popular and recommended metrics, and section 2.6 gives a detailed account of the different approaches to GEC observed, starting from earlier, rule-based and classifier approaches, and ending with more modern, neural approaches such as transformers. Following this, performance boosting techniques identified throughout the presented works are listed, along with common techniques used in low-resourced environments. Section 2.9 gives a brief overview on the different approaches observed for synthetic corpus creation; identifying popular techniques, and section 2.10 gives an overview of work done on GEC for the Maltese Language as of yet; comprising of a statistical approach by Mizzi [6], and a more recent neural, approach by Debattista [4], as well as the corpora available. Finally, section 2.11 provides a summary for the entire chapter, giving the main takeaways.

2.1 The Maltese Language

Maltese is the national language of the Maltese archipelago and an official European Uunion (EU) language, having a rich and unique linguistic heritage [8]. Derived from late medieval Sicilian Arabic and influenced by Romance languages, particularly Italian, English, and French; Maltese is considered a mixed language. Like other Semitic languages, it features root-and-template morphology. However, Maltese is distinguished by several key characteristics, including free word order, mixed morphology, an aspect-based temporal system, and the absence of a morphological infinitive [8]. The Maltese alphabet, unlike other Semitic languages, is based on the Latin script and includes additional letters marked by diacritics and digraphs, these being ċ, ġħ, ż, ġ, and ħ [8].

Marmara' [9] performed a population-representative study on the comparative use of Maltese to our second national language, English. It was concluded that the majority of the population are able to understand both Maltese, with 98.9% and

English, with 93.2%. With regards to written Maltese, it is mainly used for: writing emails, social media posts and interactions, writing cards, and personal writing, whilst English is the main language used for writing mobile messages. Consequently, 94% of the representative population express the need for a spell checker for the Maltese language, in order to support day-to-day writing tasks such as mobile messages (86.4%), as well as email communications (70.3%), social media posts and interactions (62.5%), writing in education settings (54.0%) and personal writing (48.1%).

2.2 Grammar Error Correction

GEC is the systematic process of identifying and rectifying grammatical errors in written text to support clarity and correctness. This process involves several components, including grammatical error detection, which focuses on improving the likelihood of identifying errors [1]. GEC systems, or spell checkers, are helpful to non-native and native language users alike; from occasional errors, such as punctuation and minor spelling mistakes, to the correction of grammar and sentence structure. GEC tasks are therefore not limited to grammatical errors, but further expand to orthographic and word-choice errors, as represented in table 1 [1]. In their survey study, Bryant *et al.* [1] comment on the vast number of variations that exist within the task of GEC; from minor changes, such as keyboard proximity errors, to more extensive reviews of paragraphs and larger bodies of text, such as contextual errors, as represented in table 2.1.

Table 2.1 Examples of Error Types [1]

Type	Error	Correction
Preposition	I sat in the talk	I sat in on the talk
Morphology	dreamed	dreamt
Determiner	I like the ice cream	I like ice cream
Tense/Aspect	I like kiss you	I like kissing you
Agreement	She likes him and kiss him	She likes him and kisses him
Syntax	I have not the book	I do not have the book
Punctuation	We met they talked and left	We met, they talked and left
Unidiomatic	We had a big conversation	We had a long conversation
Multiple	I sea the see from the seasoar	I saw the sea from the seesaw

Bryant *et al.* [1]'s table was further adapted to represent a subsect of common Maltese errors, as represented in table 2.2.

Table 2.2 Table adapted from Bryant *et al.* [1] for the Maltese language

Type	Error	Correction
Preposition	qiegħed għall-mejda	qiegħed *fuq il*-mejda
Determiner	Nħobb ġelat	Nħobb *il*-ġelat
Tense/Aspect	Ilbieraħ nsiefer Londra	Ilbieraħ *sifirt* Londra
Syntax	għandi il-ktieb mhux	*il-ktieb mhux għandi*
Punctuation	Iltaqajna, tkellimna u tlaqna	Iltaqajna, tkellimna, u tlaqna
Unidiomatic	Kellna diskursata kbira	Kellna diskursata *twila*

2.3 Computational Architectures Used for GEC

The section presents an overview of the main model architectures that will be discussed throughout. The models can be split into two main types; statistical models, described in section 2.3.1 and neural models, described in section 2.3.2.

2.3.1 Statistical Models

Statistical GEC utilises Statistical Machine Translation (SMT) models. SMT is inspired by the noisy channel model [13]. The noisy channel model views translation as a process in which a 'corrupted' source language is passing through a channel, and the goal is to decode it into its corrected target language, as visually depicted in figure 2.1. This can be adapted for GEC systems, in which the incorrect sentence is 'translated' into its corrected form. The 'translation' from a corrupted source to its corrected target is achieved through the probabilistic creation of candidate translations, $P(C))$, which are then re-ranked according to likelihood, $P(E|C)$, and the highest scoring candidate is selected, $\hat{C}$. Candidate sentences are generated through a decoder using beam search [14].

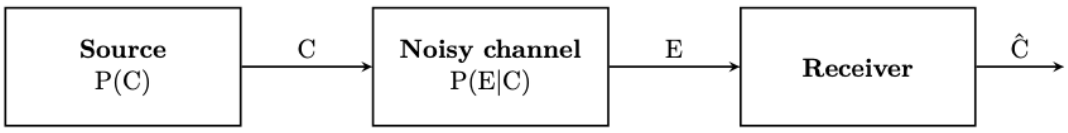


Figure 2.1 Noisy Channel Model as presented by [13].

Beam search tracks a fixed number of the most likely sequences at each step, known as the beam width. Instead of exploring all possible sequences, it expands only the top sequences by retaining the most probable options at each step. Language Models (LMs) play a crucial role in this process by predicting the probability

of word sequences. As new evidence is introduced, Bayes' Rule is applied to update the probabilities of translation options, leading to more accurate translations. Bayes' Rule is defined as follows:

$$P(H|E) = \frac{P(E|H) \cdot P(H)}{P(E)}$$

where $P(H|E)$ is the posterior probability of the hypothesis H given the evidence E, $P(E|H)$ represents the likelihood of the evidence E given the hypothesis H, $P(H)$ is the prior probability of the hypothesis H before considering the evidence E, and $P(E)$ is the prior probability of the evidence E before considering the hypothesis H. In the context of SMT for GEC, this is applied as follows [15]:

$$\hat{C} = \arg\max_C P(C \mid E) = \arg\max_C \frac{P(E \mid C)P(C)}{P(E)} = \arg\max_C P(E \mid C)P(C) \quad (2.1)$$

In which the goal is to reconstruct the correct sentence C from an erroneous sentence E by utilising a LM $P(C)$ and a Translation Model (TM) $P(E|C)$.

2.3.2 Neural Architectures

Neural GEC systems generally employ an encoder–decoder framework, in which the encoder takes an input and compresses it into a fixed-size vector, from which the decoder generates an output sequence [1]. An encoder encodes an input into a context vector representation containing the information needed for the task. The context vector is then iteratively decoded into an output sequence by the decoder. The decoder uses attention mechanisms to focus on the relevant parts of the vector during generation. Examples of commonly used sequence-to-sequence models include Recurrent Neural Network (RNN) [16], Long Short Term Memory (LSTM) [17] and Transformers [18].

RNN

A RNN can process sequential data whilst retaining historical information through its hidden state [19]. RNNs have the capacity to consider the context of earlier inputs when making predictions through the use of memory feedback loops. As visually represented in figure 2.2, the RNN takes the input and hidden state from its previous step to produce an output and update the hidden state. This design makes RNNs effective for processing sequential data, such as language modeling and time series prediction. However, RNNs can struggle with longer sequences

due to difficulties in retaining information over many steps, a challenge known as the vanishing gradient problem [19].

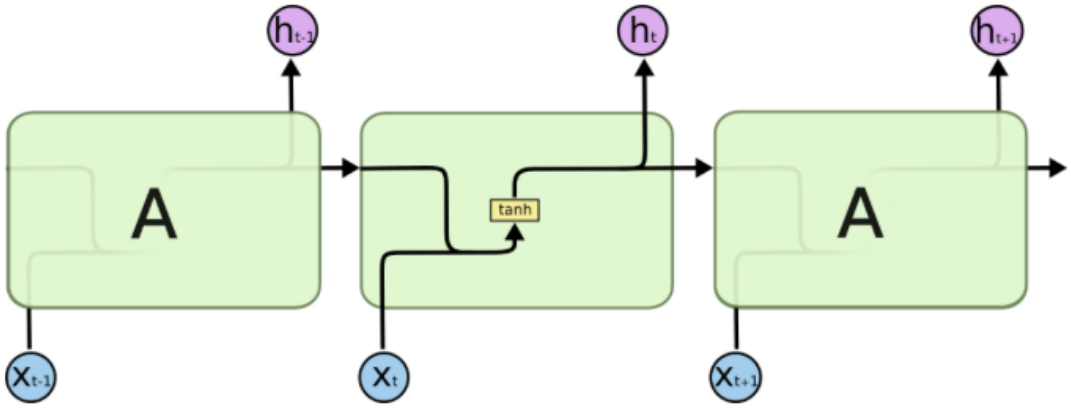


Figure 2.2 RNN Cell [19].

LSTM

As described by Hochreiter and Schmidhuber [17], the LSTM architecture is designed to address the vanishing gradient problem present in RNNs. LSTM models learn and retain information over longer sequences, making them effective for tasks involving long-term dependencies. A key feature of the LSTM model is the memory cell. The memory cell supports retention of historical information over time, this is achieved by passing information through a linear chain of repeating modules called LSTM units. Each LSTM unit consists of three main components: an input gate, a forget gate, and an output gate, the gates regulate the flow of information into and out of the memory cell. The input gates regulates what input information is stored in the memory cell, this is achieved by comparing the current input and previous output to determine the relevance of the new information. The forget gate regulates what information should be discarded from the memory cell, this is achieved by considering the previous output and the current input to decide the relevance of previously stored information. The output gate determines which parts of the memory cell should be outputted as the final result, this is achieved by considering the current input and the previous output to determine the relevant information to be sent forward, as visually represented in figure 2.3.

The LSTM architecture utilises the Back-Propagation Through Time (BPTT) algorithm for training [17]. This involves propagating errors back through time, updating the model's weights and biases to improve the prediction performance. Overall, the LSTM architecture consists of memory cells and gates, allowing the network to capture long-term dependencies. This combination of components enables LSTMs to more effectively process sequential data [19]. As described

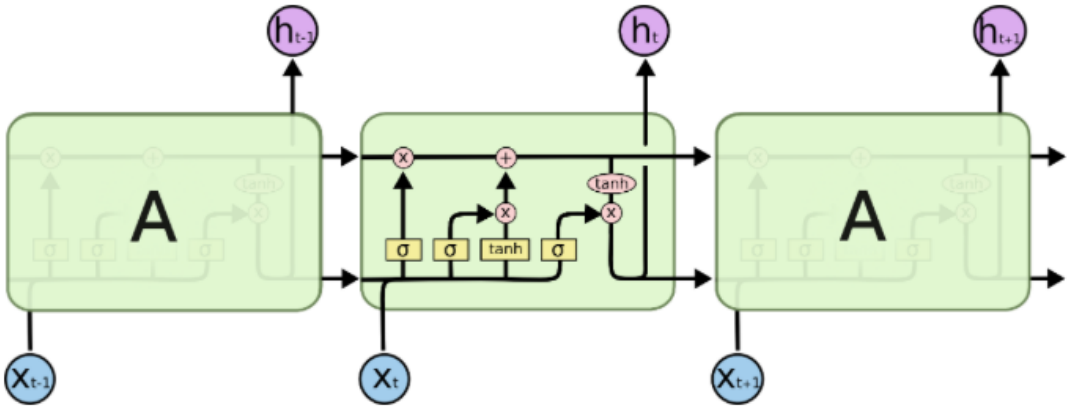


Figure 2.3 LSTM Cell [19].

by Peters *et al.* [20], Embeddings from Language Models (ELMo) is a bidirectional LSTM, combining both lower-level and higher-level states, the model captures both context-dependant aspects of word meaning and model aspects of syntax. The bidirectional language model (biLM) is capable of efficiently encoding a range of syntactic and semantic information about words-in-context.

Transformers

The Transformer is the first network to fully rely on a self-attention mechanism to process input sequences, without the need for recurrence [18]. This design allows for better parallelization across multiple GPUs. The transformer model [18] has improved sequence transduction, particularly in Machine Translation (MT) tasks. Transformers build upon the limitations of RNNs, such as parallelisation and efficiency over longer sequences of text. Vaswani *et al.* [18] propose a model based entirely on an attention mechanism, allowing it to capture broad dependencies between inputs and outputs without relying on recurrence. This approach not only improves parallel processing but also achieves superior translation quality. Vaswani *et al.* [18]'s experiments show significant improvements in BLEU scores for English-to-German and English-to-French translations.

A Transformer processes sequences using embeddings combined with positional encodings to retain the order of the data, unlike RNNs, which handle data sequentially [18]. The key feature of a Transformer is its multi-head attention mechanism, allowing the model to focus on different parts of the input simultaneously. Each attention head captures different relationships and dependencies within the sequence, which is more effective than using a single head. After attention, the data is passed through a feed-forward neural network, which applies non-linear transformations to each position independently. To stabilise training, the Transformer uses layer normalisation, helping to maintain gradient flow dur-

ing back-propagation. The architecture consists of multiple layers of attention and feed-forward networks in both the encoder, which processes the input, and the decoder, which generates the output, as visually represented in figure 2.4 [18]. One major advantage of the Transformer is its ability to process sequences in parallel, leading to faster training times, especially with large datasets [18].

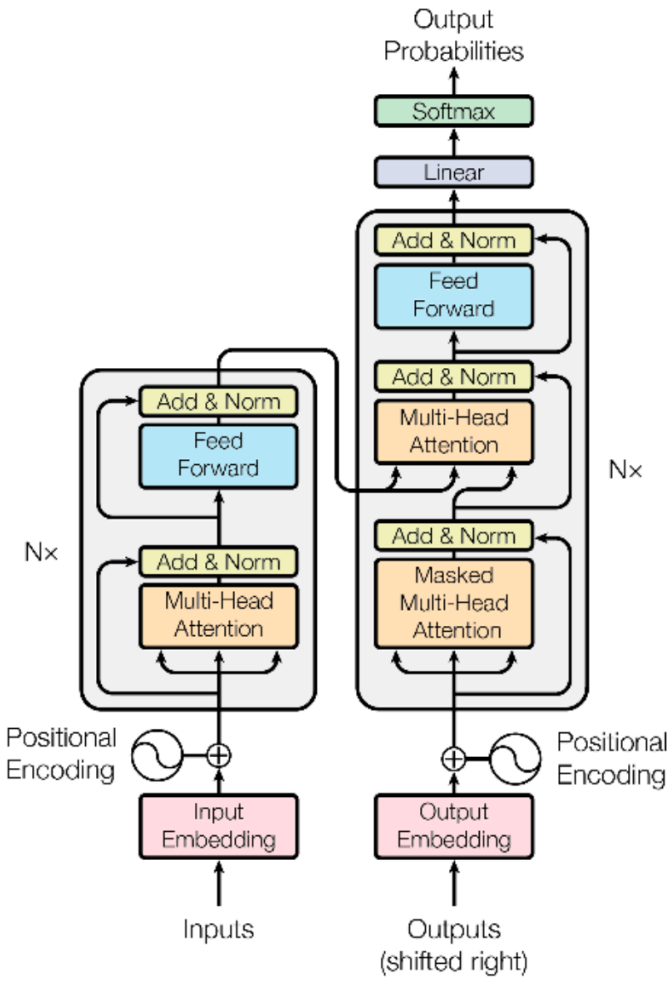


Figure 2.4 Transformer Architecture [18].

The Transformer model led to the creation of Bidirectional Encoder Representations from Transformers (BERT), and Multilingual Bidirectional Encoder Representations from Transformers (mBERT) models, which are built on a stack of Transformer encoder layers.

2.3.3 Language Models

A Language Model (LM) can be defined as a neural model that represents linguistic information. Bengio *et al.* [21] proposed one of the earliest neural LMs to over-

come the curse of dimensionality—referring to the challenge where model performance deteriorates as data dimensionality increases, a significant issue in language processing. Bengio *et al.* [21] introduced a neural model that learns a distributed representation of words, allowing the model to capture information from neighboring words or sentences. Today, LMs are integral to many state-of-the-art NLP approaches [1], including models such as Bidirectional Encoder Representations from Transformers (BERT) and Multilingual BERT (mBERT) [22].

BERT and mBERT

The architecture of the BERT model allows for adaptation for numerous NLP tasks, with minimal modifications to its architecture [22]. By simply adding an output layer, BERT can be adapted to specific tasks. The model uses a multi-layer bidirectional Transformer encoder, where each layer includes self-attention mechanisms and feed-forward neural networks [22]. Self-attention helps the model capture dependencies between words in a sentence, whilst the feed-forward networks model complex relationships [18]. BERT is pre-trained on a corpus of unlabeled English text, learning to predict masked words and understand sentence-level representations, which helps to realise the contextual relationships between words. BERT is able to process 110 million parameters, totaling to 714 megabytes.

BERT's pre-training involves two main tasks: Masked Language Modeling (MLM) and Next Sentence Prediction (NSP) [22]. In MLM, some tokens in the input are randomly masked, and the model is trained to predict them based on the surrounding context. NSP, on the other hand, trains the model to predict whether two sentences are consecutive, helping it understand sentence relationships. After pre-training, BERT can be fine-tuned for specific tasks by adding task-specific layers and training the entire network with labelled data [22].

mBERT, the multilingual version of BERT, is pre-trained on a corpus of text in 104 languages [22], and is able to process 178 million parameters, totaling to 440 megabytes. Like BERT, mBERT uses WordPiece tokenisation, which breaks words into sub-words and assigns tokens to each, enabling it to handle Out Of Vocabulary (OOV) words and capture detailed information. A key advantage of mBERT is its ability to transfer knowledge across languages. By learning language-agnostic representations during pre-training, mBERT can be fine-tuned for various tasks in different languages, making it a powerful tool for multilingual NLP applications [22]. Pre-training the BERT and mBERT models on the Maltese language gave rise to BERTu and mBERTu models.

BERTu and mBERTu

The BERT and mBERT models do not include Maltese in their training data [23]. Analysing Maltese text using mBERT relies on its similarity to the languages included in the model. To address this limitation, Micallef *et al.* [10] pre-trained the BERT and mBERT models on the Maltese language, coined BERTu and mBERTu respectively. Both models were pre-trained using the Korpus Malti v.4.0, consisting of 52K tokens. BERTu was pre-trained over 1M steps, whilst mBERTu was pre-trained over 250K steps. Both the BERTu and mBERTu models demonstrate state-of-the-art performance in Maltese NLP tasks such as Dependency Parsing, Part-of-Speech Tagging, Named Entity Recognition, and Sentiment Analysis. Overall, the development of the mBERTu and BERTu models has led to improved performance in various NLP tasks for the Maltese language compared to the original mBERT and BERT models [24].

2.4 Shared Tasks in Error Correction

Shared tasks play a crucial role in advancing research and development in various fields, including GEC [1]. Several shared tasks highlight significant milestones in GEC, such as the Helping Our Own, HOO 2011 [25] and HOO 2012 [26] Shared Tasks, the Conference on Natural Language Learning, CoNLL- 2013[27] and CoNLL-2014 [28] Shared Tasks, and the Building Educational Applications, BEA-2019 Shared Tasks [29].

2.4.1 Key GEC Corpora for English error correction

Key GEC corpora for English error correction include The National University of Singapore Corpus of Learner English (NUCLE) corpus [30], which was the official corpus for the CoNLL-2013 [27] and CoNLL-2014 [28] shared task, as well as one of the official corpora for the BEA-2019 shared task [29]. The NUCLE corpus contains 1,397 argumentative essays (approximately 1.16m words) written by NUS undergraduate students [30]. The Write&Improve+LOCNESS corpus and the LOCNESS corpus [29], were released as the official corpus for the BEA-2019 shared task, containing roughly equal amounts of data from different ability levels: Beginner, Intermediate, Advanced and Native, which were submitted to the Write & Improve online essay-writing platform, totaling to 3,600 essays (approximately 755k words) written by international learners of all ability levels (A1-C2) and 100 essays (approximately 46.2k words) written by native British/American English undergraduates. Other commonly used datasets include the The Johns

Hopkins Fluency-Extended GUG corpus (JFLEG) [31], totaling to 1,501 sentences (approximately 28.1k words) and the Lang-8 corpus [32], totaling to 100,000 submissions (approximately 11.8m words).

2.4.2 HOO 2011 & HOO 2012 Shared Tasks

The HOO 2011 pilot shared task [25] focused on addressing textual errors introduced by non-native speakers of English. Building on this foundation, the HOO 2012 shared task [26] narrowed its focus to more specific error types, particularly preposition and determiner errors. The HOO 2011 task utilised a set of fragments of text published by the Association for Computational Linguistics, ACL, whilst the HOO 2012 task utilised the publicly available First Certificate in English dataset for training and evaluation. Systems in both the 2011 and 2012 tasks were evaluated based on how closely their proposed corrections matched the gold-standard edits, using F-score as the primary metric. These tasks were instrumental in advancing the field of GEC by targeting common error types and establishing benchmarks for system evaluation.

2.4.3 CoNLL-2013 & CONLL-2014 Shared Tasks

The CoNLL-2013 [27] and CoNLL-2014 [28] shared tasks on GEC aimed to evaluate algorithms and systems for automatically detecting and correcting grammatical errors in English essays written by second-language learners. The CoNLL-2013 task focused on five specific error types: article or determiner, preposition, noun number, verb form, and subject-verb agreement, while excluding other errors, like word choice [27]. In contrast, the CoNLL-2014 task expanded its scope to address 28 different error types [28]. Both tasks used the NUCLE corpus, the NUS Corpus of Learner English, as the training data [30]. The systems were evaluated by how well the corrections or edits matched the gold-standard edits, using F_1 scoring for CoNLL-2013, and $F_{0.5}$ for CoNLL-2014. The highest scoring system in the CoNLL-2013 shared task achieved an F_1 score of 42% when scored with multiple acceptable answers [33], the system included a separate classifier model built for each specific error type. The highest scoring system in the CoNLL-2014 achieved an 45.57% when scored with alternative answers [34], the highest scoring approach built upon its predecessor in the CoNLL-2013 shared task [33], adding more classifiers, and introducing model combination.

2.4.4 BEA-2019 Shared Task

The BEA-2019 Shared Task on GEC was a key benchmark for evaluating systems designed to automatically correct grammatical errors in English text [1]. Participants worked with the Write&Improve+LOCNESS corpus, which included a diverse range of English proficiency levels from both native and learner speakers, supporting the development of systems that can handle various types of errors effectively [29]. The systems were evaluated using the ERRANT F_0.5 metric [29]. Approaches in the task included hybrid systems, such as the combination of Neural Machine Translation (NMT) with Finite State Transducer (FST) [35], and an ensemble system using CNNs and Transformer-based models [36] - achieving $F_{0.5}$ scores of 0.6678 in both the Restricted and Unrestricted Tracks, referring to the restricted and unrestricted use of data and resources respectively. Furthermore, the model scored 0.5181 in the low-resource track, ranking highly in the competition [36].

2.5 Evaluation Metrics

The most common evaluation metrics for GEC systems include the MaxMatch (M2) scorer [37], ERRor ANnotation Toolkit, ERRANT scorer [38], Grammar and Language Error Evaluation, GLEU scorer [39] and the Bilingual Evaluation Understudy, BLEU scorer [40]. The CoNLL-2014 shared task was evaluated using the M2 scorer [28], the JFLEG corpus is generally evaluated using GLEU [31], and the BEA-2019 shared task was evaluated using the ERRANT scorer [29].

2.5.1 M2 Scorer

The M2 Scorer is a reference-based metric that compares the system output edits with the target edits. It counts a True Positive (TP) when an edit matches the target, a False Positive (FP) when an edit does not match the target, and a False Negative (FN) when a necessary edit is missing from the system output [30]. The TP, FP and FN values can be used to calculate the precision; as follows:

$$P = \frac{TP}{TP + FP} \tag{2.2}$$

The recall, as follows:

$$R = \frac{TP}{TP + FN} \tag{2.3}$$

And finally, using the precision and recall, the $F_{0.5}$ score, as follows:

$$F_{0.5} = \left(1 + 0.5^2\right) \times \frac{P \times R}{(0.5^2 \times P) + R} \tag{2.4}$$

2.5.2 ERRANT Scorer

The ERRANT scorer is similar to the M2 scorer. It is also a reference-based metric, and measures performance according to an edit-based $F_{0.5}$ score. However, the ERRANT scorer uses a linguistically-enhanced Damerau-Levenshtein alignment algorithm, in which it identifies the edits in the generated text, and classifies them through a rule-based evaluator [38]. This allows for a more granular calculation, with the caveat that it is only available for the English language. The rule based evaluator allows for the following classification of errors:

- Edit Operations, consisting of 3 labels; missing edits, replacement edits, unnecessary edits.
- Main Type, consisting of 25 labels, such as spelling errors or verb tense errors.
- Parallel decoding, consisting of 55 labels, such as missing nouns or replacement nouns.

2.5.3 BLEU and GLEU Scorer

The GLEU scorer is a reference-based evaluation metric, similar to the BLEU (Bilingual Evaluation Understudy) scorer, but with key differences tailored for GEC tasks. Like BLEU, GLEU compares the n-grams of the model's output (hypothesis) with the reference (corrected) sentences. However, GLEU also penalizes n-grams that match the original (incorrect) sentence, ensuring that only meaningful corrections are rewarded, rather than unnecessary changes. Unlike BLEU, GLEU does not require detailed edit annotations, relying only on the corrected target sentences.

The GLEU score considers the source sentences (original sentences) O = o_1, ..., o_k, the model output (the hypothesis) H = h_1, ..., h_k, and the target sentences (the reference sentences) R = r_1, ..., r_k. Each original, output and target sentence, denoted by o_i, h_i and r_i respectively, represent n-grams of length n = 1, 2, ..., N within the sentences. The precision term, p_n, is then calculated as follows [1]:

$$p_n = \frac{\sum_{i=1}^{|H|} \left(\sum_{g \in \{h_i \cap r_i\}} \text{count}_{h_i,r_i}(g) - \sum_{g \in \{h_i \cap o_i\}} \max\left[0, \text{count}_{h_i,o_i}(g) - \text{count}_{h_i,r_i}(g)\right] \right)}{\sum_{i=1}^{|H|} \sum_{g \in \{h_i\}} \text{count}_{h_i}(g)} \tag{2.5}$$

In which $\text{count}_a(g)$ represents the number of occurrences of n-gram g in a, and $\text{count}_{a,b}(g)$ represents the number of occurrences of n-gram g in a and the number of occurrences of n-gram g in b.

Furthermore, the GLEU score has a Brevity Penalty,BP, in which outputs which are shorter than the source sentences are penalised as follows [1]:

$$\text{BP} = \min\left(1, \exp\left(1 - \frac{|\text{reference length}|}{|\text{output length}|}\right)\right) \tag{2.6}$$

In which 'reference length' is the length of the reference sentence, 'output length' is the length of the generated output sentence and 'exp' represents the exponential function.

2.6 GEC Approaches

The section presented covers the different approaches to GEC, starting from an overview of rule based and classifier approaches in subsection 2.6.1, to more modern, neural approaches in subsection 2.6.3. Subsection 2.6.2 provides an overview of SMT approaches. The mentioned subsections compare and contrast the different approaches taken in terms of performance and model capabilities, as well as in chronological order. Section 2.7 provides a compilation of performance boosting techniques, whilst 2.8 covers common approaches in low-resourced environments. Subsection 2.9 covers common approaches for synthetic dataset creation. Finally, subsection 2.10 covers GEC approaches for the Maltese language, including corpora available.

An important distinction exists between hybrid models, model ensembling, and model pipelining. According to Wang *et al.* [11], hybrid models integrate various sub-systems into a unified correction process; in contrast, model ensembling involves combining the outputs of several independent systems to produce a single final result, and model pipelining, on the other hand, refers to the process of sequentially passing the output of one system to the next. These definitions will serve as the standard references for hybrid, ensemble, and pipelined systems going forward.

2.6.1 Rule Based and Classifier Approaches

Early GEC systems involved hard-coded rules that encode grammatical aspects of language [41]. An example of such an approach is Park *et al.* [2], who developed a GEC system using a set of grammar rules implemented in Prolog. The system

targeted common errors, including incorrect capitalisation, missing sentence fragments, and extra elements.

As rule-based methods evolved, classifiers were introduced, in which contextual features of an incorrect word inform correction predictions; here, errors are identified and corrected through a comparison of the best candidate predicted by the classifier and the original word in the text [1]. An example of an early classifier approach is Lee [42]'s 'Automatic Article Restoration'. Lee [42] developed a classifier to restore missing articles, a common mistake among non-native English speakers [42]. Their approach applies a log-linear model to estimate the probabilities of different articles based on the noun phrase's context. The final result was an accuracy of 87.7% in article generation, which refers to the generation of the correct article; between 'null', referring to no article, 'an' and 'the', 11.9% in article insertion, 19.3% in article deletion and 11.8% in article substitution. Similar approaches have been attempted for prepositions [43], noun number [44] and word forms [45].

In classification-based approaches to GEC, the effectiveness of the model often depends on selecting the right features that capture relevant linguistic information. Since different types of grammatical errors, like article misuse [42], or noun number [44], are influenced by different aspects of language, it's essential to tailor classifiers specifically to each error type, creating a system of classifiers for each error type to maximise performance [46]. More recent approaches include the use of neural networks for classifiers. Wang *et al.* [46] introduced the Deep Context Model, where each type of error has a dedicated model that learns from variable-length context embeddings surrounding the target word; and then predicts said target. In total, 5 deep context models were trained separately for article, preposition, verb form, noun number and subjective agreement errors, with an overall $F_{0.5}$ score of 41.4.

Classifiers are case-specific; their corrections can only target specific errors, furthermore, layering these classifiers is a complex task due to order-specific corrections [1]. Another disadvantage is the fact that classifiers consider only local context, under the assumption that the error is singular and the surrounding context is correct. Classifiers are considered useful for practical errors, such as articles, prepositions, noun numbers and verb forms, but are poor performing compared to more recent approaches [1], such as statistical models, and neural models.

2.6.2 Statistical Approaches

Although originally developed for translation tasks, SMT can be adapted for GEC by treating the 'incorrect' sentence as if it were being translated into its 'corrected'

version [1]. SMT is effective in addressing multiple types of errors simultaneously without extensive feature engineering [1]. Unlike rule-based systems, SMT can handle complex and interacting errors, making it useful for correcting a wide range of grammatical mistakes [1]. One early application of SMT in GEC was by Brockett *et al.* [47], who investigated the use of phrasal SMT to correct grammatical errors made by learners of English as a Second Language (ESL). Brockett *et al.* [47] study focused on correcting mass noun errors, specifically targeting 14 countable and uncountable nouns, the presented approach successfully corrected 61.81% of errors in a set noun errors sourced online.

Building on this foundation, Felice *et al.* [48] and Junczys-Dowmunt and Grundkiewicz [49] introduced GEC-specific features to SMT, recognising that the correction is generally similar to the original sentence, and most errors made usually resemble their corrected form. Felice *et al.* [48] developed a hybrid model that combined SMT with rule-based methods, using techniques such as character-level Levenshtein distance for word alignment and a 4-gram target language model, ultimately scoring a 0.3733 $F_{0.5}$ on the CoNLL-2014 test set, whilst Junczys-Dowmunt and Grundkiewicz [49] trained large-scale language models on the NUCLE corpus, and fine-tuned a phrase-based SMT system according to the M^2 metric, ultimately scoring a 49.49 M^2 on the CoNLL-2014 test set.

Despite their relative success, SMT-based approaches produce phrases that are locally well-formed, but result in poor overall grammar. Additionally, they tend to make unnecessary corrections, such as overcorrecting phrases to more common versions. Furthermore, SMT systems struggle to handle long-range dependencies and have difficulty adapting to more specific error types [41]. Lastly, the performance of SMT-based approaches is highly dependent on the quantity and quality of training data, which is particularly scarce in the context of GEC [1].

2.6.3 Neural Approaches

In contrast to SMT, the NMT correction process involves a single neural network, eliminating the need for complex feature engineering specific to GEC [50]. Training a neural GEC system is, furthermore, an end-to-end process, meaning it does not require any additional components. Although achieving state-of-the-art performance in GEC, NMT systems have shortcomings, most notably their data requirements. In order to successfully correct complex errors, neural approaches require large amounts of data. Moreover, these models are black-box, which means that there is no exact explanation of the reasoning behind the corrections made[1].

RNN

Yuan and Briscoe [50] present an NMT-based approach for GEC. The authors [50] apply a bidirectional RNN encoder and an attention-based RNN decoder, allowing the model to capture historical data and focus on relevant information in the source sentence. The approach outperformed baseline SMT-based [48] approaches with a M^2 score of 39.90 on the CoNLL-2014 test set, scoring higher than state-of-the-art approaches. Xie *et al.* [51] propose the use of an RNN encoder-decoder with an attention mechanism. Xie *et al.* [51]'s system operated on a character level in both the encoder and decoder to account for OOV instances. Building upon Xie *et al.* [51], Ji *et al.* [52] presents a hybrid neural model with nested attention layers. The proposed approach incorporating both word-level and character-level information; using a word-level sequence-to-sequence model as a backbone, and introducing character-level encoder, decoder and attention components, helping to account for both contextual information as well as OOV instances.

Transformers

Different transformer approaches have been attempted for GEC tasks; Zhao *et al.* [53] propose a copy-augmented architecture, in which unchanged words are copied from the source sentence to the target sentence; this approach outperformed then state-of-the-art approaches. Other adaptations made to encoder-decoder transformers include modifying attention mechanisms in the decoder to allow for more contextual information, such as Zhang *et al.* [54]'s SynGEC, representing syntax enhanced GEC, which integrates dependency syntactic information into the encoder part of GEC models. This was done through the manual design of a syntax representation scheme, representing grammatical and syntax errors in a tree structure; parse trees were then created from said scheme, which were used to train a GEC-oriented parser (GOPar). Zhang *et al.* [54]'s approach ultimately scored a 46.51 $F_{0.5}$ on Chinese datasets, achieving state-of-the-art results.

Due to the limitation of parallel GEC training data, transformer models can be pre-trained and adapted to GEC tasks for improved performance [12], Choe *et al.* [55] pre-train transformer models, along with introducing transfer learning techniques for better domain-adaptation and contextual understanding, achieving a 32.69 $F_{0.5}$ on the W&I+LOCNESS test set. More recently, advancements in large pre-trained language models have demonstrated that directly fine-tuning these models with GEC parallel data can achieve performance comparable to pre-training with synthetic data [56], Luhtaru *et al.* [57] tested the zero-shot capabilities of a multilingual MT model, NLLB for GEC. Furthermore, the model was fine-tuned on parallel data, synthetic error data and human-annotated error datasets.

Multiple languages were attempted, this included Estonian, German and Czech. Ultimately, the model had consistently higher $F_{0.5}$ scores than state-of-the-art approaches [57], with an 65.25 $F_{0.5}$ score on the CoNLL-2014 test set.

Another popular approach involves leveraging the capabilities of the BERT model by integrating them into encoder-decoder transformer approaches [58] [12] [59]. Kaneko *et al.* [12]'s research incorporates pretrained masked language models (MLM) into an encoder-decoder system for GEC. The approach involves BERT output as additional features for a GEC model, pipelining BERT to an encoder-decoder model. Kaneko *et al.* [12] score 69.8 $F_{0.5}$ score on the BEA 2019 test set. Vázquez *et al.* [60] propose replacing NMT encoders with BERT models through contextual alignment for representation spaces [61]. Furthermore, Guo *et al.* [59] proposes a MT solution incorporating BERT models into sequence-to-sequence models using adapters. Here, BERT models are used for both the encoder and decoder components. The ensemble is built upon a parallel sequence decoding algorithm named Mask-Predict.

Combined Approaches

Chollampatt and Ng [62] developed a GEC system based on SMT, significantly enhancing its performance by incorporating and adapting Neural Network Joint Models (NNJM). The ensembling performed led to a 3.62% improvement in $F_{0.5}$ scoring on the CoNLL-2014 test set. Similarly, Grundkiewicz and Junczys-Dowmunt [63] achieved state-of-the-art results on benchmarks such as CoNLL-2014 and JFLEG by employing a hybrid SMT-NMT pipeline, combining strengths of both statistical and neural approaches.

More recently, Yuan *et al.* [35] further advanced the field by integrating neural networks with Finite State Transducers (FST) in a GEC system. The final pipeline effectively utilised a combination of a CNN, Transformer, FST, and re-ranking techniques to improve error correction accuracy. Furthermore, Yuan *et al.* [64] explored the integration of different methodologies by developing a multi-class Grammatical Error Detection (GED) system for English. The multi-class system consisted of a pre-trained GED Transformer which was then pipelined into an encoder-decoder GEC model, achieving state-of-the-art results on the BEA 2019 test set.

2.7 Performance Boosting Techniques

The proposed section compiles performance boosting techniques commonly used for GEC, including different types of pre-training techniques, as described in 2.7.1, different approaches to candidate re-ranking, as described in 2.7.2, the use of mul-

tiple model systems, as described in 2.7.3, and character, sentence and document level approaches, which account for OOV instances and improved contextual understanding, as described in 2.7.4.

2.7.1 Pre-training

Pre-training models has been shown to improve convergence speed and model efficiency [11], giving models inherent understanding from related data, and minimising the extensive data requirements, this is especially helpful when considering that GEC is a low-resourced task [1]. Therefore, pre-training and transfer learning techniques can boost GEC performance [12] [55]. Wang *et al.* [11] identifies four different types of pre-training approaches: pre-training the model on GEC related data [53] (1), or pre-training with large monolingual data [59] (2) and incorporating pre-trained models into the system [12] (3), or pre-training the entire system [55] (4). Due to the nature of the task, systems pre-trained for GEC generally make use of synthetic data [65] [11].

2.7.2 Candidate Re-Ranking

Candidate re-ranking is generally used as a post-processing technique, in which the n-best outputs with the highest probabilities are re-scored in order to select the optimal candidate [11]. Candidate re-ranking allows for more linguistic information to be introduced to the GEC system, and the incorporation of outputs from different systems, allowing for model ensembling. Different re-ranking approaches include n-best re-ranking [66] [67] [64], Right To Left (R2L) re-ranking [68] and LM re-ranking [51] [48].

2.7.3 Multiple Model Systems

System combinations can be either: model ensembling, model pipe-lining or hybrid model systems, the distinctions between the three are described in section 2.6. System combination can potentially improve model performance [11], often achieving better predictions than any individual model system [69]. Various authors have experimented with model ensembling, pipelining and hybrid model systems. Kaneko *et al.* [12] create a model pipeline incorporating pre-trained BERT, Chollampatt and Ng [62] ensemble SMT and NNJMs, achieving state-of-the-art results, Yuan *et al.* [35] pipeline FSTs and neural networks, and Yuan *et al.* [64] pipeline a series of models to create a multi-class GEC system. All authors reported competitive scores resulting from the integration of multiple models.

2.7.4 Character, Sentence and Document - level Approaches

The considerations for character, sentence and document level depend on the context of the study. Character level GEC allows for better handling OOV words and rare word instances [51] , building on this, Ji *et al.* [52] combine both word and character level processing, capturing a wider range of errors. Both character-level and word-level approaches are unable to account for more contextual errors that are visible on a sentence or even document level. Most approaches take a sentence-level approach [49], [5]. Chollampatt *et al.* [58] attempt cross sentence GEC, in which previous sentences' encodings are incorporated during decoding through attention and gating mechanism, this allows for capturing larger context and therefore capturing a larger range of errors. Yuan and Bryant [70] incorporate both sentence-level and document-level GEC, concluding that whilst context is useful for GEC; very long contextual information does not support improved model performance.

2.8 GEC for Low Resourced Environments

In creating a larger and more diverse human-annotated corpus for Czetch, Náplava *et al.* [3] report a significant improvement in model performance with an increase in dataset size and quality. Low-resourced languages are therefore at a significant disadvantage. Bryant and Briscoe [7] re-evaluate statistical approaches for low-resourced environments, making use of minimal annotated data (approximately 1000 sentences). Here, a simple LM approach was considered, utilising confusion sets created from normalised log-probabilities. Although producing competitive results, with a final $F_{0.5}$ score of 31.37 on the CoNLL-2014 test set, this approach was not able to account for all error types [7]. Other approaches in low-resourced environments include pre-training and transfer-learning techniques, such as those proposed by Choe *et al.* [55], Katsumata and Komachi [56] and Zhao *et al.* [53], as well as leveraging unlabelled data such as Zhao *et al.* [53] and Yasunaga *et al.* [65]. In their low-resourced approach to GEC for Ukrainian, Palma Gomez *et al.* [71] pre-train a mT5 model on synthetic data, and further fine-tune on gold standard data, achieving a 63.4 $F_{0.5}$ score when trained on 35 million parameters in an ensemble system.

2.9 Synthetic Dataset Creation

GEC tasks require source and target training data; the source being the 'incorrect' sentence, and the target being the gold standard, 'corrected' sentence. A synthetic parallel corpus reflects this, consisting of the source and target; in which the source sentences are formed by adding noise to a corpus, whilst the target sentences are the original, correct sentences [1]. The proposed section discusses popular noising techniques used to create synthetic source datasets for GEC.

2.9.1 Rule-based Noising

Rule-based noising involves applying a series of transformations on a clean dataset based on pre-established rules. These rules can be determined empirically, probabilistically, or through observation of data [1]. Zhao *et al.* [53] apply a series of deletion, insertion, replacement, and shuffling errors to words in a sentence, Grundkiewicz *et al.* [68] take a similar approach, but also consider character-level deletion, insertion, replacement, and transposition.

Xie *et al.* [72] define different probabilistic noising techniques that can be applied to a dataset. These consist of: rank penalty noising, which involves penalising siblings from the same parent based off of their log-probability value and top penalty noising, which only penalises words with the highest log-probability value.

2.9.2 Back Translation

Another approach is back translation. Back translation involves training a reverse-model to generate ungrammatical sentences from their original, grammatical versions. Kiyono *et al.* [73] define 3 main noising techniques: noisy back translation [72]; which involves uniformly sampling noise at intervals throughout the dataset, sample back translation; which involves decoding sentences through sampling from the distribution of a reverse-model, and direct back-translation; which involves directly injecting noise into the grammatical sentences [53].

2.10 GEC for the Maltese Language

The proposed section provides an overview of available corpora for the Maltese language (subsection 2.10.1), following this, two previous approaches to spell checking for the Maltese language are explored. The first approach is a statistical approach by Mizzi [6] in subsection 2.10.2, consisting of basic statistical GEC tech-

niques, and the second approach is a more recent study by Debattista [4] in subsection 2.10.3, consisting of more modern NMT techniques.

2.10.1 Available Corpora

There are no established corpora available for GEC for the Maltese language. Debattista [4] utilised a dataset collected from the 'Qari tal-Provi' proof reading diploma course, totaling to 7,606 source and target sentences, with 148,070 source tokens and 148,408 target tokens, and created a synthetic source target corpus using the Maltese Language Resource Server (MLRS) corpus [74] [1]; which is a monolingual corpus for the Maltese language containing a collection of 130 million tokens compiled from publicly available documents and a small amount of user contributed materials, totaling to 3,623 source and target sentences, with 88,680 source tokens and 88,677 target tokens, and Mozilla's Common Voice corpus [2], which is a data collection initiative in which Maltese audio and its corresponding textual sentences are collected, resulting in 5,256 source and target sentences, comprising of 52,010 source tokens and 52,009 target tokens. The final dataset proposed by Debattista [4] consisted of 16,485 source and target sentences.

The dataset collected from the 'Qari tal-Provi' diploma course consisted of error correction data created for proofreading exercises by Vella [75], and more recently by Vella [76]. Both approaches resulted in source and target corpora. Vella [75]'s approach included a special parser in order to explore the XML markup of each document's 'Track Changes,' and obtain each string and its associated corrections. The incorrect and corrected strings were then used by Debattista [4] to create the final source and target sentences.

The error synthesis dataset attempted by Debattista [4] was created using a rule-based approach inspired by Junczys-Dowmunt *et al.* [77], applying a variety of synthesis strategies to grammatically correct text taken from the MLRS corpus [**MLRS**]. This dataset included a baseline of insertion, deletion, and transposition errors, along with more specialised strategies. Specialised strategies consisted of: the 'No Font' strategy, where the English keyboard was used to substitute Maltese characters, the 'Silent Letter' strategy, which involved removing silent letters from the Maltese alphabet, the 'Common Article' strategy, which kept prefixed vowels instead of dropping them, and the 'Corruption Noising' strategy, which involved deleting a word from a string of text. Error adaptation based off of the 'Qari tal-Provi' dataset was later applied to the synthesised dataset in order ensure homogeneous error patterns. Table 2.2 below shows the final split of error seeds

[1] `https://mlrs.research.um.edu.mt/index.php?page=corpora`
[2] Mozilla CommonVoice for Maltese: https://commonvoice.mozilla.org/en

introduced in the data.

Table 2.3 Error Seeds Introduced in the Data [4]

Error Strategy	Sentence Seed	Token Seed	Character Seed
Common Article	0.5	0.2	-
No Silent Letters	0.4	0.4	-
No Fonts	1	1	-
Key Proximity	0.2	0.1	0.05
Insertion	0.2	0.1	0.05
Duplicate	0.2	0.1	0.05
Substitution	0.2	0.1	0.05
Omission	0.2	0.1	0.05

2.10.2 Statistical Approach

Mizzi [6] implemented a statistical approach for spell checking for the Maltese language, consisting of a statistical n-gram spelling model. Mizzi [6] proposed a system that consisted of 3 main tasks: document normalisation, spelling verification and spelling correction, as represented in figure 2.5 [6].

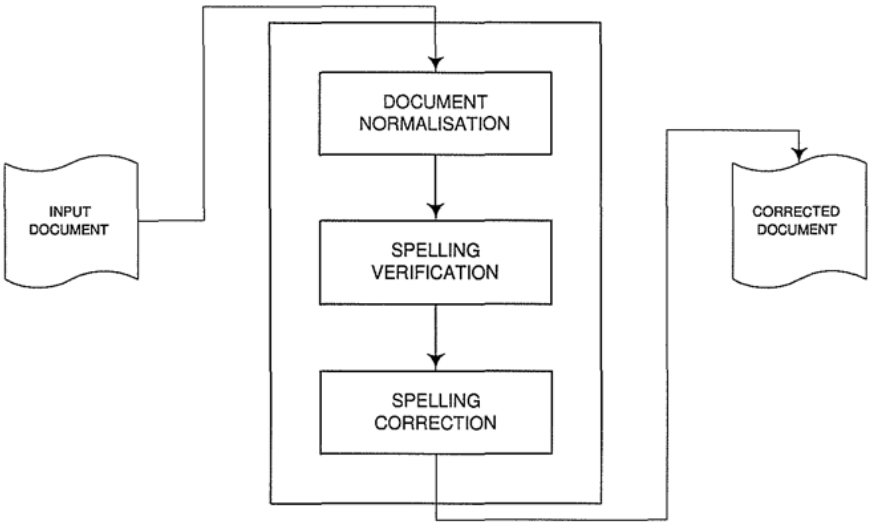


Figure 2.5 Overview of Maltese Spell Checker [6].

Document normalisation consisted of appropriate tokenisation of the data, spelling verification consisted of checking for spelling errors within the text. Here, a probabilistic approach was attempted using n-gram analysis. N-gram analysis splits a target word into is corresponding n-grams. Each n-gram is then assigned a weight reflecting its frequency of occurrence in the training lexicon used. These weights are then use to calculate the index of peculiarity for the complete word, as represented in equation 2.7, where for any word, w, composed of the characters c , c

...cn, we can express the probability p(w). Finally, spelling correction takes place through a Bayesian-hybrid method [78], which allows for the correction of both context words and allocations through the use of Bayesian classifiers.

$$p(w) = p\left(c_1\right) p\left(c_2 \mid c_1\right) p\left(c_3 \mid c_1 c_2\right) \ldots p\left(c_n \mid c_1 \ldots c_{n-1}\right) = \begin{cases} p\left(c_1\right) & \text{if } k = 1 \\ \prod_{k=1}^{n} p\left(c_k \mid c_1 \ldots c_{k-1}\right) & \text{if } k > 1. \end{cases} \tag{2.7}$$

Ultimately, Mizzi [6]'s approach was able to account for 50% of grammatical errors in the Maltese language, this was mainly attributed to the lack of a tagged corpus.

2.10.3 Neural Approach

Debattista [4] implemented an NMT approach for spell checking for the Maltese language, using the Marian NMT toolkit [77] to create the GEC model. As displayed in figure 2.6, pre-trained BERTu and mBERTu embeddings were used over the baseline BERT and mBERT models, the inclusion of pre-trained models significantly improved the system output.

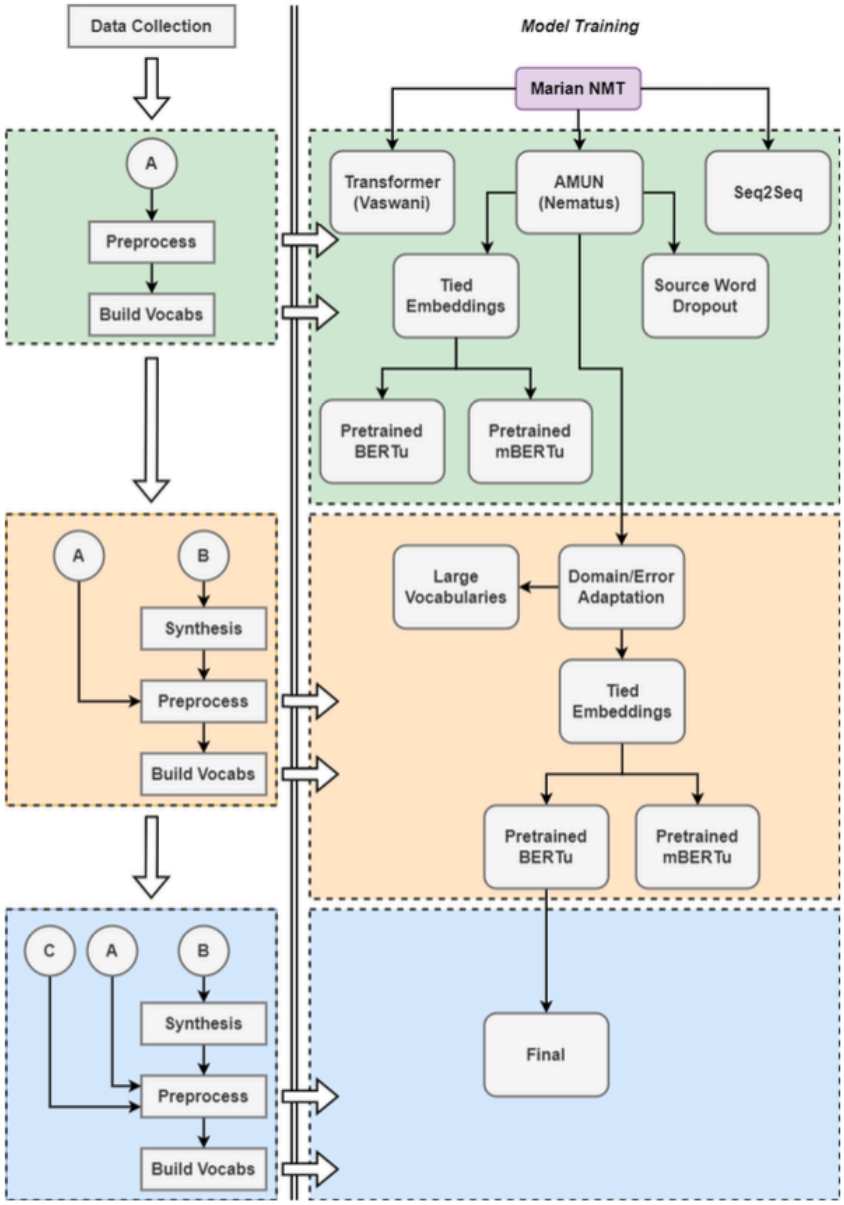


Figure 2.6 Overview of Maltese Spell Checker by Debattista [4] .

Source and target datasets consisted of a blend of human annotated data sourced from the 'Qari tal-Provi' proofreading university course, as well as synthetically generated data. Ultimately, the model achieved a 31.84% $F_{0.5}$ score for error correction. The system suffered from translation bias, producing large amounts of false negatives, domain bias, due to it being trained with political and news-related data as a result of the low resources available, and was also unable to account for OOV instances; this was largely attributed to the lack of training resources available for Maltese.

2.11 Main Findings

The proposed literature review offers valuable insights into the development of GEC systems, with a particular focus on GEC for the Maltese language, which is recognised as a low-resourced language (section 2.1). Consequently, techniques and approaches aimed at boosting performance in low-resourced environments (section 2.8) are examined. The primary objective of this research is to support the creation of a spell-checking system for the Maltese language. To this end, studies on the necessity of such a resource (section 2.1) and standard definitions of GEC [1] (section 2.2) were identified.

Building on these foundational definitions and research, the primary architectures utilised in the field are reviewed in section 2.3, followed by an exploration of their application in GEC tasks (sections 2.6 and 2.4). In retrospect, it is clear that the transformer model is the most successful in the field, consistently achieving higher scores compared to its statistical, RNN, rule-based, and classifier counterparts. Subsequently, the BERT architecture and its integration into GEC were examined, once again yielding competitive results. In addition to these approaches, sections dedicated to performance-enhancing techniques (section 2.7), synthetic corpora creation (section 2.9), standard metrics (section 2.5), and benchmark corpora (section 2.4.1) are also provided. Section 2.10 provides insights into GEC for the Maltese language.
To conclude, the research highlights 3 main research gaps:

1. The need for a larger, higher-quality dataset for Maltese
2. The effectiveness of transformer approaches and pre-trained models like BERTu and mBERTu in improving system performance
3. The popularity and success of multi-model systems in the field.

The findings suggest that a multi-model, pre-trained system [12, 59] would likely yield the best results for a GEC system. Additionally, in low-resourced environments, the research underscores the importance of a more diverse and extensive dataset [3, 4, 6].

3 Methodology

The proposed methodology covers the development of the final GEC model. The different replications and ideas surrounding each are considered in section 3.1, giving a justification for the final selected approach. Following this, the implementation of the selected approach is described in section 3.2, including the necessary adaptations made considering the change from MT to GEC, the low-resourced environment and Maltese language. Section 3.3 details the corpus creation; from the collection of authentic human errors through a created interface, to the use of the authentic error corpus to create a statistically informed synthetic corpus, this leads up to section 3.4, which describes the final datasets used in this work. The scoring strategy decided upon is covered in section 3.5. Finally, section 3.6 describes the technologies used.

3.1 Replicated Work

Following the research findings in chapter 3, the central idea was to create a system that leverages pre-trained BERT model capabilities, adapting BERTu and mBERTu models for the task of GEC. The different approaches for which adaptation for Maltese GEC was attempted and are discussed below. These approaches all incorporate pre-trained models or multi-model systems, and were viewed as potential candidates for a GEC system for Maltese.

3.1.1 GEC System using the Marian NMT toolkit

The approach by Debattista [4] was replicated for model comparison and data analysis, being the previous and only neural GEC system for the Maltese language. The system used by Debattista [4] was based off of a system implemented for the BEA 2019 shared task [29] by Grundkiewicz *et al.* [68], making use of the Marian NMT toolkit [68]. The system was adapted for GEC, this involved training on a Maltese corpus and adapting the system to include Maltese BERTu and mBERTu models during its pre-training stages. Replication included plugging the readily pre-trained model provided by Debattista [4] into the system to achieve more accurate results, this approach was successfully replicated yielding a final $F_{0.5}$ score of 24%, compared to Debattista [4]'s final $F_{0.5}$ score of 31.84%. The discrepancy between the achieved score and the final score reported by Debattista [4] could be an attribute of hardware differences, or the use of different software versions, considering that the replication took place a year after the study was complete. Debattista [4]'s

approach served as a baseline system, and was therefore used for comparison purposes in terms of system and dataset performance. Throughout this research, this system will be referred to as Debattista [4]'s system.

3.1.2 Pipelining Encoder-Decoder Models and Pre-trained MLMs

The approach by Kaneko *et al.* [12] involves pipe-lining pre-trained Masked Language Models (MLMs) into encoder-decoder models in order to boost model performance [12]; achieving state-of-the-art results on BEA-2019 and CoNLL-2014 benchmarks. The setup consisted of pre-training a MLM on a GEC corpus. The output of the fine-tuned MLM model was then used as an additional feature to the GEC model, therefore maximising the capabilities of MLMs. Here, different BERT models were experimented with, ultimately achieving the highest scoring results with the BERT-fuse model. The idea was to replicate this approach and implement the BERTu and mBERTu models in the pipeline. Code replication was attempted, but was unsuccessful due to deleted external files. Observing Kaneko *et al.* [12]'s research shifted our research focus into incorporating the Maltese BERT models into encoder-decoder structures.

3.1.3 Addressing the Differences Between BERT and MT Encoder Spaces for Translation Tasks

The main idea following Kaneko *et al.* [12]'s approach was to somehow directly incorporate pre-trained BERT models into an encoder-decoder pipeline. Our research focus was the possibility of replacing an encoder component in an encoder-decoder system with a BERT model, creating a BERT-decoder system. The BERT-decoder system would eventually involve the BERTu and mBERTu models. Vázquez *et al.* [60] attempts to tackle the differences between BERT models and MT encoder spaces, ultimately suggesting how to leverage pre-trained BERT in encoder-decoder translation tasks. This approach describes splitting an encoder-decoder model and replacing the encoder side with BERT using alignment techniques [61]. However, the implementation itself was not a complete system, but rather a series of analysis understanding the intrinsic differences between BERT and machine translation encoders. Replication of the experiments was successful. Furthermore, Vázquez *et al.* [60] suggested the merging of BERT into an encoder-decoder space through aligning the representation spaces through a study conducted by Cao *et al.* [61]. Both Vázquez *et al.* [60] and Cao *et al.* [61]'s approaches were successfully replicated, however, both implementations did not provide a complete system usable for GEC, and were therefore put aside in search for a more straight forward

approach.

3.1.4 Incorporating BERT into Parallel Sequence Decoding with Adapters

Whilst most approaches that incorporate BERT models do so in just the encoder side of the system [12], Guo *et al.* [59] propose leveraging BERT capabilities for both encoder and decoder components. Their approach consists of incorporating BERT in parallel sequence decoding tasks using adapters. Parallel sequence decoding refers to multiple parts of the model output being decoded in a constant number of iterations, rather than one by one [79], this is described in further detail in section 3.2. The approach was originally developed for MT tasks; however, it was adapted for GEC through employing model optimisation techniques [1]. This included training the system on GEC-specific datasets, adjusting hyperparameters, and modifying system size, as later detailed in section 3.2. Guo *et al.* implement BERT models as both encoder and decoder components, allowing for in-depth experimentation and testing of downstream tasks. Each component of the system can be considered a plug-in unit, meaning that the system can be adapted to include BERTu and mBERTu models. The system was successfully replicated and the data was successfully substituted with Debattista [4]'s dataset. This approach was ultimately opted for, specifically for the following reasons:

- The technique was original and appeared to be a promising approach to investigate.
- The framework proposed was light-weight and adaptable, where each component of the system can be considered as a plug-in unit. It allows for a more seamless incorporation of the BERTu and mBERTu models, as well as streamlined task adaptation to GEC.
- The system incorporates BERT for both the encoder, and decoder components, allowing further leveraging of pre-trained Maltese models and potentially providing better support for the low-resources available.

3.2 Implementation

The proposed section describes the system structure implemented, including the changes made to support GEC for the Maltese language. The system can be summarised into 3 main components [59]:

- Incorporating pre-trained BERT-based models
- Fine-tuning with adapters
- Parallel decoding

3.2.1 Incorporating Pre-trained BERT-based Models

Unlike previous studies that incorporate BERT into text generation [12, 60], Guo *et al.* [59] introduce BERT models as both the encoder, and decoder components of the system, as depicted in figure 3.1 [59]. The system was evaluated for machine translation in the following languages: bidirectional translation from English to German, bidirectional translation from English to Spanish, Italian and Dutch, and English to French. For our experiment, the translation language is Maltese, adapting the system to translate from the 'incorrect' source to the 'corrected' target. In order to implement this, the BERTu and mBERTu models [10] were used. The integration of BERTu [1] and mBERTu [2] models was done through adapting the HuggingFace models and dictionaries to fit the code implementation, zipping the models into tar.gz extension folders, and linking them in the systems' pre-trained model archive map and pre-trained vocab archive map.

3.2.2 Fine-Tuning With Adapters

Guo *et al.* [59] define adapters as light-weight neural networks added into the internal layers of the pre-trained models in order to support downstream tasks; to adapt BERT models for Natural Language Generation (NLG), adapter layers are inserted into each BERT layer, as represented in figure 3.1.

Incorporation of the adapters was achieved through adaptation of the loss function, introducing a loss function such as the one used for conditional MLM (equation 3.4), as follows [59]:

$$L\left(y^m \mid y^r, x; \theta_{\text{AENC}}, \theta_{\text{ADEC}}\right) = -\sum_{t=1}^{|y^m|} \log P\left(y_t^m \mid y^r, x; \theta_{\text{AENC}}, \theta_{\text{ADEC}}\right) \tag{3.1}$$

in which θ_{AENC} and θ_{ADEC} represent the parameters of the encoder adapters and decoder adapters respectively. The encoder adapter layer is constructed with layer normalisation. The two layers include two feed forward networks between them [59]:

[1] BERTu model: https://huggingface.co/MLRS/BERTu
[2] mBERTu model: https://huggingface.co/MLRS/mBERTu

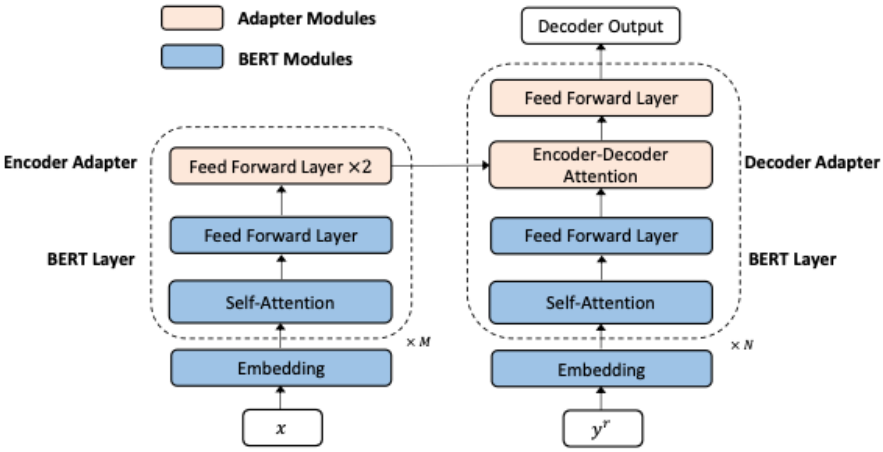


Figure 3.1 Overview of Encoder-Decoder System Using Adapters [59]

$$Z = W_1 \cdot (\mathsf{LN}(H)), \quad H_{\mathsf{AENC}} = H + W_2 \cdot (\mathsf{ReLU}(Z)) \tag{3.2}$$

where H and H_{AENC} represent the input and output hidden states of the adapter respectively, LN represents layer normalisation, and W1 and W2 represent the parameters of the feed-forward networks. The hyper-parameters introduced are the dimensions of the internal hidden state Z. The output hidden state (H) is computed as [59]:

$$H_{l+1}^{E} = \mathsf{AENC}\left(\mathsf{XbERt}\left(H_{l}^{E}\right)\right) \tag{3.3}$$

where H_{l+1}^{E} represents the output hidden state of the l^{th} encoder layer. The hidden state from the last layer of the encoder is considered the input to the decoder side adapter, represented as (H^E). For the decoder side adapter, multi-head cross attention is computed for the input, allowing the decoder to model conditional source information. The hidden state of the $(l+1)^{th}$ layer is calculated as [59]:

$$H_{l+1}^{D} = \mathsf{AdEC}\left(\mathsf{YbERT}\left(H_{l}^{D}\right), H^{E}, H^{E}\right) \tag{3.4}$$

In which AdEC represents the attention based adapter, which is fed the hidden output (H_l^D) on the l^{th} decoder layer. During fine-tuning, the pre-trained BERT layers are frozen and the adapter components are trained.

3.2.3 Parallel Sequence Decoding

Parallel sequence decoding refers to multiple parts of the model output being decoded in a series of iterations, rather than sequence-by-sequence [79]. The frame-

work is built upon a parallel sequence decoding algorithm named Mask-Predict [59]. In Mask-Predict, encoder-decoder architectures are trained as a conditional masked language model as shown in equation 3.5, in which (x, y) is a sample of parallel training pairs from the dataset, y^m and y^r represent the masked and residual target tokens respectively, and ${}_enc$ and ${}_dec$ represent the parameters of the encoder and decoder respectively. This allows for the model to make predictions in parallel [79].

$$L_{\text{CMLM}}\left(y^m \mid y^r, x; \theta_{\text{enc}}, \theta_{\text{dec}}\right) = -\sum_{t=1}^{|y^m|} \log P\left(y_t^m \mid y^r, x; \theta_{\text{enc}}, \theta_{\text{dec}}\right) \tag{3.5}$$

Given a training pair (x, y), a set of tokens in y is randomly masked using the [MASK] token, with the number of masked tokens uniformly sampled between 1 and |y|. During inference, the target sequence is generated iteratively using mask-and-predict. Here, the encoder predicts the length of the target sequence, for which all positions are initialised with the [MASK] token and passed as input to the decoder. Once the decoder generates its prediction, low-probability tokens in the output are selected and replaced with the [MASK] token. This updated sequence is then used as the input for the next iteration. This process continues until a stop condition is met. The number of [MASK] tokens selected in each iteration follows a linear decay function. The stop condition is triggered when either the maximum number of iterations is reached, or the target sequence remains unchanged between two consecutive iterations [59].

3.2.4 Adaptations for Low-Resourced Environments

To improve system performance in low-resource environments, several performance-boosting techniques were applied as compiled in section 2.7. The system utilised pre-trained BERTu and mBERTu models, which were further fine-tuned for GEC, the adapter modules were both pre-trained and fine-tuned specifically for GEC, and the overall system was further fine-tuned for the task at hand. The original parameters implemented by Guo *et al.* [59] served as the baseline, with all improvements measured relative to these initial settings.

Various configurations were tested, drawing on recent research focused on GEC in low-resource settings [1, 11], as well as integration strategies for BERT models [12], and prior experience with fine-tuning BERT for the Maltese language [24]. Previous GEC approaches for Maltese [4] were conducted in different contexts and utilised significantly less data, making most of the hyper-parameters from those studies unsuitable for the current research.

Leveraging BERT Models

Integrating pre-trained models, specifically pre-trained BERT and mBERT, into a multi-model system has been shown to yield improved results [11] [12], therefore BERTu and mBERTu models were incorporated for both the encoder and decoder components. In additon to BERTu and mBERTu, the system investigates other model ensembles, utilising BERT and mBERT. Below is a list of all encoder-decoder model ensembles utilised in the system:

- BERTu and BERTu
- mBERTu and mBERTu
- BERT and BERT
- mBERT and mBERT

Fine-Tuning the BERT Models

The parameters of the BERT, BERTu, mBERT and mBERTu models were adjusted to improve model performance. The process of adjusting the model size was an iterative one, in which the system was tested using different sizes in order to deduce the optimal configurations. The final model has a smaller hidden size, hidden layers, attention heads and intermediate size to decrease model overfitting. Furthermore, the dropout rate was increased in order to help prevent model hallucinations, as represented in table 3.1

Table 3.1 Comparison of Original and Adapted Model Configurations

Configuration	Original	Adapted
Hidden size	768	256
Number of hidden layers	12	6
Number of attention heads	12	8
Intermediate size	3072	1024
Hidden activation function	gelu	gelu
Hidden dropout probability	0.1	0.3
Attention dropout probability	0.1	0.3
Max position embeddings	512	512
Type vocab size	2	2
Initialiser range	0.02	0.01
Layer normalization epsilon	1e-12	1e-12

Pre-Training and Fine-Tuning Adapter Modules

The adapter modules were pre-trained on the generated source and target GEC data [71]. Furthermore, these modules underwent further fine-tuning through extensive hyper-parameter tuning, where the original parameters were adjusted to better align with the requirements of the GEC task. This tuning adhered to the recommendations of Guo *et al.* [59] for BERT-base models, which proposed the following settings for both the encoder and decoder sides, with the exception of the decoder-side feed-forward network (FFN) hidden dimension, which was made equal to the BERT models hidden dimensions.

- Hidden dimensions: 768
- FFN hidden dimensions: 3072
- Attention heads: 12
- Encoder/Decoder layers: 12

The final parameters for the encoder were as follows, similar to the BERT fine-tuning, the hidden dimensions, attention heads and encoder layers were decreased:

- Hidden dimensions: 768
- FFN hidden dimensions: 1024
- Attention heads: 8
- Encoder layers: 8

The final parameters for the decoder were as follows, here, the decoder layers were further decreased, and it was ensured that the FFN hidden dimensions was equal to the BERT models hidden dimension.

- Hidden dimensions: 768
- FFN hidden dimensions: 1024
- Attention heads: 8
- Decoder layers: 6

Additionally, similar to Debattista [4]'s approach, dropout was introduced, this helps to reduce model overfitting and improve learning in low resourced environments:

- Attention dropout: 0.4
- Activation dropout: 0.4
- Dropout: 0.4

Fine-Tuning the Overall System

The initial configuration was based on the work of Guo *et al.* [59], which focused on English-German machine translation, as shown in table 3.2. The initial set of hyper-parameters encountered several challenges, including model hallucination and a validation loss plateauing after four epochs, as illustrated in Figure 3.2.

In response to the unsatisfactory results with the baseline configurations, several adjustments were implemented to enhance the model's robustness. Specifically, dropout and attention dropout were introduced, weight decay was increased, and the maximum number of updates was raised. Additionally, the learning rate was adjusted, and the scheduler was switched to a cosine strategy. These modifications resulted in significant improvements in model performance. The final hyper-parameters are represented in table 3.2.

Table 3.2 Comparison of Configuration Sets

Configuration	Original	Adapted
Learning rate scheduler	Inverse Square Root	Cosine
Learning rate	0.0005	1e-05
Minimum learning rate	1e-09	1e-07
Maximum learning rate	-	0.0003
Label smoothing	0.1	0.1
Weight decay	0.0	0.01
Max tokens	2000	3000
Max updates	200000	300000
Dropout	-	0.2
Attention dropout	-	0.2

Overall, these iterative adjustments resulted in a more effective model, characterised by reduced hallucination and improved training stability. Figure 3.2 shows that the final validation loss is significantly lower than the baseline. Additionally, the final model demonstrates a larger decrease in loss over the same number of epochs, which can be attributed to decreased overfitting.

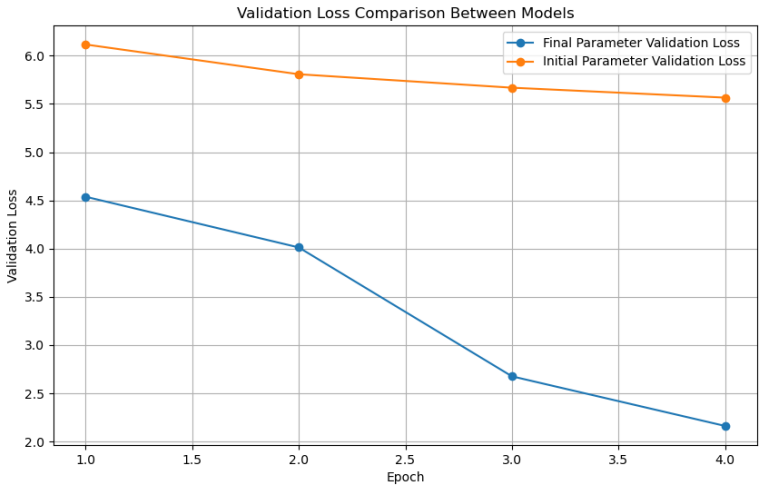


Figure 3.2 Loss for Initial and Final Fine-Tuning

3.3 Corpus Creation

Both previous approaches for GEC for the Maltese language, carried out by Mizzi [6] and Debattista [4], identified the need for a larger, more varied corpus for the Maltese language. Mizzi [6]'s approach utilised a series of sentences in Maltese, the source of which was not made clear. Debattista [4]'s approach utilised a corpus adapted from the 'Qari tal-Provi' proofreading diploma course, along with a synthetic corpus [77], which were made available through GitHub [3]. It can be argued that the proofreading corpora are not representative of authentic errors made during day-to-day typing. The proofreading corpora include mistakes made by students who already have an academically informed understanding of the language and are training for a proofreading profession, therefore, the text being corrected is far from day-to-day typing, as presented in table 3.3. As a result, the errors captured in these datasets may not accurately reflect the more diverse and spontaneous mistakes made by the majority of the population in everyday writing. Furthermore, the synthetic dataset created was not statistically representative of common Maltese errors. It was ultimately decided that a more varied and representative dataset was required for improved results.

[3] https://github.com/AaronDebattista09/ICS5200

Table 3.3 Comparison of Errors from the Qari tal-Provi Dataset and the Authentic Dataset

Source Text	Target Correction
Mit-testijiet li sarulu rrizulta li t-tifel kellu leukemia.	Mit-testijiet li sarulu rriżulta li t-tifel kellu leukemia.
Ghaldaqstant ma laqghatx it-talba ghall-kumpens.	Għaldaqstant ma laqgħatx it-talba għall-kumpens.

This section describes the process for the creation of an authentic source target dataset, and the creation of a synthetic dataset thereafter.

3.3.1 Data Collection Efforts

The main goal for the dataset was to collect as many authentic human errors as possible, from a diverse population [3]. It was decided that errors will be collected through an interface, these errors will then be used to create a statistically informed source-target synthetic dataset, figure 3.3 summarises the corpus creation process.

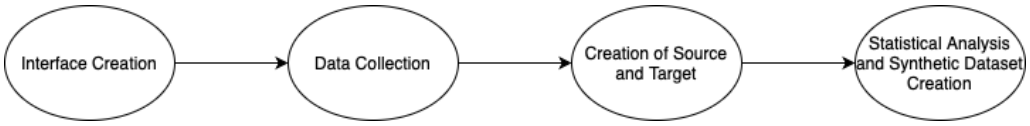


Figure 3.3 Proposed Authentic Error Dataset Creation Process.

3.3.2 Interface Creation and Data Collection

An interface was created and distributed [4], allowing users to listen to a sentence read aloud in Maltese and then type out the sentence in a provided text box, as shown in Figure 3.4.

[4] https://nlpgroup.research.um.edu.mt/simerr/index.html

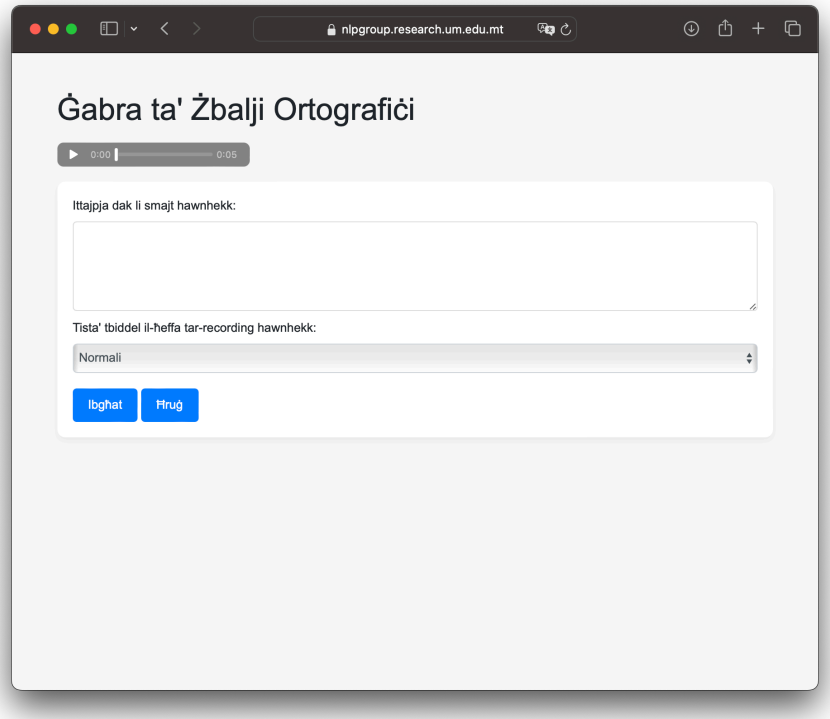


Figure 3.4 GEC Collection Interface, Main Page.

The audio files and their corresponding correctly spelled sentences were sourced from Mozilla's Common Voice, an open-source project. Common Voice was selected since its validation process involves the sentences and their corresponding audio files to be checked by 2 people. Common Voice allows for the downloading of .mp3 speech files, which were then matched to their corresponding transcriptions in a .csv file. The initial mapping system referenced the audio files by their original naming conventions. However, since Mozilla's Common Voice is a work in progress—with only 45% [5] of the audio files and corresponding text being validated, the numbering of audio files was not sequential, with many gaps in the sequence. To simplify and standardise the file names, they were renamed from the original format, such as common_voice_mt_20987403.mp3, to a more straightforward convention, with files renamed in ascending numerical order starting from 1. A mapping system was created, recording the new names alongside their corresponding old names in the .csv file, ensuring a clear and systematic method for tracking and accessing each audio file.

The interface was developed using HTML, CSS, and JavaScript, with PHP handling file naming and saving processes. It randomly selects an audio file, which the user listens to and transcribes. The transcribed text is saved in a .txt file, with each sentence stored in a separate text file named after the corresponding audio file. To

[5] https://pontoon.mozilla.org/mt/common-voice/project-info/

prevent overwriting duplicates, the file names also include a timestamp, which is later removed during the mapping process. This approach ensures that all versions of the transcribed sentences are uniquely identifiable and preserved for further analysis. The interface was hosted on a university server.

Users were instructed to type out the sentences as naturally as possible, without any knowledge of the correct versions. It was emphasised that they should type in a manner reflective of everyday writing, without the specific need to use a Maltese keyboard (see Figure 3.5 for a snippet containing the provided instructions). To capture authentic errors, the backspace key, spell-check, and auto-correct features were disabled. The dataset was distributed through official university channels, Facebook pages dedicated to the Maltese language [6], a talk at the ICT Faculty's FICT Exhibition [7], and the 'Sound Bites' snippet on the Sunday Times of Malta, dated July 28, 2024. This effort resulted in a collection of misspelled Maltese sentences over a data collection period of 2 months, yielding a total of 2,890 responses. The number of individual users was not recorded due to data privacy. The responses were manually vetted for noisy input, totaling to 2,814 valid responses. The erroneous text generated through the interface was used as the source text, while the correct sentences served as the target. The collected data was then used to create a statistically informed synthetic error dataset, enabling a series of analyses to produce similar results.

[6] https://www.facebook.com/groups/kelmetilmalti/permalink/7741571452585025/
[7] https://www.um.edu.mt/ict/expo/mainstage-auditorium/

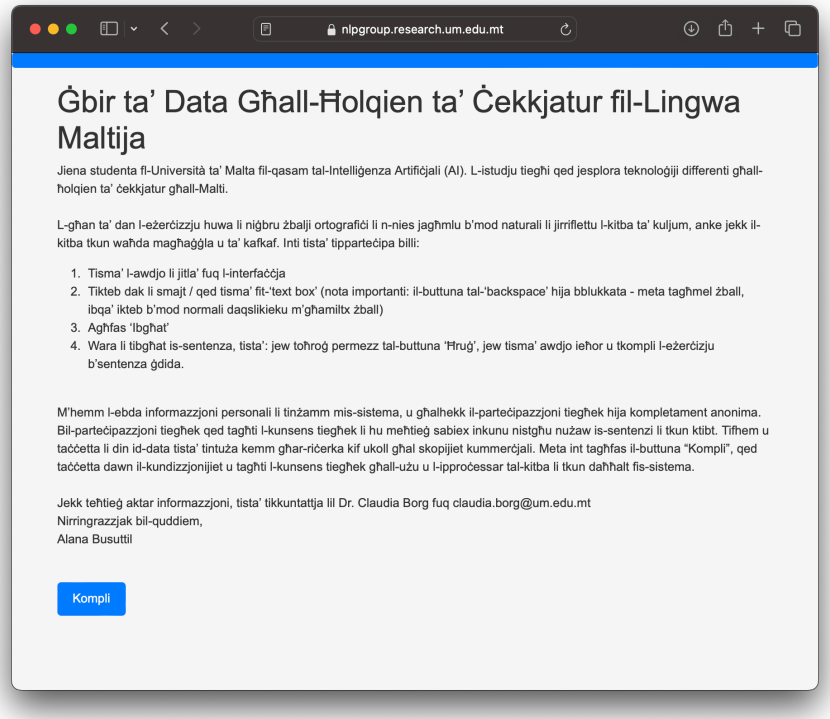


Figure 3.5 GEC Collection Interface, First Page.

3.3.3 Statistical Analysis

The baseline statistical analysis employed the minimum edit distance, also known as Levenshtein distance. This method calculates the fewest number of edits—insertions, deletions, and substitutions—needed to transform the source string into the target string, as shown in Equation 3.6.

$$\text{lev}_{a,b}(i,j) = \begin{cases} \max(i,j) & \text{if } \min(i,j) = 0 \\ \min \begin{cases} \text{lev}_{a,b}(i-1,j) + 1 \\ \text{lev}_{a,b}(i,j-1) + 1 \\ \text{lev}_{a,b}(i-1,j-1) + 1_{(a_i \neq b_j)} \end{cases} & \text{otherwise.} \end{cases} \tag{3.6}$$

In which a represents the source string, and b represents the target string. $a_i$is the i-th character of string a, b_j is the j-th character of string b. The minimum edit distance calculation was modified to output the number of insertions, deletions, and substitutions, as well as the specific positions where these edits occurred for each sentence pair. An example is as follows:

Misspelled Sentence: Dk kien it-tieni inċident f'inqas min gimgħatejn.
Corrected Sentence: Dan kien it-tieni incident f'inqas minn ġimagħtejn.
Edit Distance: 7
Insertions: 3
Deletions: 1
Substitutions: 3
Operations:
Replace 'k' at position 1 with 'a'
Insert 'n' at position 2
Replace 'ċ' at position 19 with 'c'
Insert 'n' at position 37
Replace 'g' at position 38 with 'ġ'
Insert 'a' at position 43
Delete 'a' from position 43

Next, the final counts for various types of errors were recorded, presented in table 3.4. The recorded errors were insertion errors, deletion errors, substitution errors, transposition errors caused by keyboard proximity, duplicate character errors, typographical errors unrelated to font issues, and 'random' errors — replacement errors that did not fit into any of the aforementioned categories.

Table 3.4 Error Types and Percentages

Error Type	**Count**	**Percentage (%)**
Total Insertions	5,714	21.82
Total Deletions	4,380	16.71
Total Substitutions	7,125	27.21
Proximity Errors	419	1.60
Random Errors	3,971	15.16
Same Letter Typed Twice	354	1.35
English to Maltese Errors	2,735	10.43

Keyboard proximity errors were identified through mapping, where a layout of keys and their neighboring keys was considered. The proximity mapping used is illustrated in Table 3.5.

Table 3.5 Proximity Map of Characters

Char	Proximity	Char	Proximity	Char	Proximity	Char	Proximity
a	qwsz	i	ujko	q	wa	y	tghu
b	vghn	j	uhmn	r	edft	z	asx
c	xdfv	k	ijlo	s	awedxz	ċ	1
d	serfcx	l	kop	t	rfgy	ġ	ħ
e	wsdr	m	njk	u	yhji	ż	asxz
f	rtgdc	n	bhjm	v	cfgb	Ċ	1
g	tyhfvb	o	iklp	w	qase	Ġ	Ħ
h	yujnbg	p	ol	x	zsdc	Ż	Z
						(space)	cvbnm,./

Similarly, typographical errors resulting from ‘no font errors’ were identified using a map of commonly misplaced characters, as displayed in table 3.6.

Table 3.6 Mapping of Non-Maltese Characters to Maltese Characters

Maltese	ħ	ġ	ċ	ż	għ	Ħ	Ġ	Ċ	Ż	Għ
Non-Maltese	h	g	c	z	gh	H	G	C	Z	Gh

The position of the errors was also analysed, categorising how many errors occurred at the beginning, middle, and end of sentences, as represented in table 3.7.

Table 3.7 Error Position Counts and Percentages

Error Position	Count	Percentage (%)
Beginning	3,381	19.64
Middle	9,382	54.49
End	4,456	25.88

The percentage of occurrences for each number of mistakes, ranging from a minimum of 0 to a maximum of 10, was calculated. Subsequently, the percentage of each type of mistake—insertions, deletions, and replacements—was determined, as illustrated in table 3.8.

Table 3.8 Probability of Occurrence for Insertion, Deletion and Replacement errors, for each Number of Errors per Sentence

Number of Errors per Sentence	Occurance %	Insertion %	Deletion %	Replacement %
0	7.6%	/%	/%	/%
1	12.8%	45.02%	16.01%	38.97%
2	12.8%	32.47%	16.46%	61.06%
3	11.6%	30.87%	16.22%	52.91%
4	9.9%	29.26%	18.89%	51.85%
5	7.9%	29.66%	18.31%	52.03%
6	6.8%	29.24%	20.38%	50.38%
7	5.9%	30.02%	19.03%	50.35%
8	4.3%	29.22%	19.73%	50.05%
9	3.3%	31.37%	22.11%	46.52%
10	3.1%	28.4%	26.23%	45.37%

Empirically, the different types of errors were mapped to insertion, deletion, and replacement mistakes by observing word pairs consisting of the incorrect word and its corrected version. This was done by analysing the statistics and source-target word pairs; table 3.9 shows a snippet of the source-target word pairs.

Table 3.9 Source and Target Word Pairs

Source	Target	Error Type
Jekk	Jekk	None
kienx	kienx	None
hemm	hemm	None
abbuż	abbuż	None
jew	jew	None
le	le,	Insertion of ‘,’
ma	ma	None
nafx.	nafx.	None
Dpmna	Domna	Replacement of ‘p’ with ‘o’
sena	sena	None
nistennew	nistennew	None
biex	biex	None
naraw	naraw	None
x	x’se	Insertion of “se’
jizvilluppaw	jiżviluppa.	Deletion of ‘w’ and insertion of ‘.’
hemm	Hemm	Replacement of ‘h’ with ‘H’
cans	ċans	Replacement of ‘c’ with ‘ċ’
kbitr	kbir	Deletion of ‘t’
li	li	None

As detailed in Table 3.10 and Figure 3.6, insertion errors were categorised into letter insertions and punctuation insertions, accounting for approximately 99.6% of all insertion errors. Deletion errors were divided into extra character deletions and keyboard proximity issues, comprising about 78.1% of all deletion errors. Replacement errors were classified into three types: replacing non-Maltese characters with Maltese characters (e.g., replacing ‘h’ with ‘ħ’), keyboard proximity errors, and capitalization errors, together making up around 66.3% of all replacement errors. The replacement of phonetically similar characters was also considered; however, the percentage was negligible. This could be due to either the rarity of such mistakes in the dataset or the test’s inability to capture errors involving phonetically similar characters, such as ‘s’ with ‘z’ and ‘b’ with ‘c’.

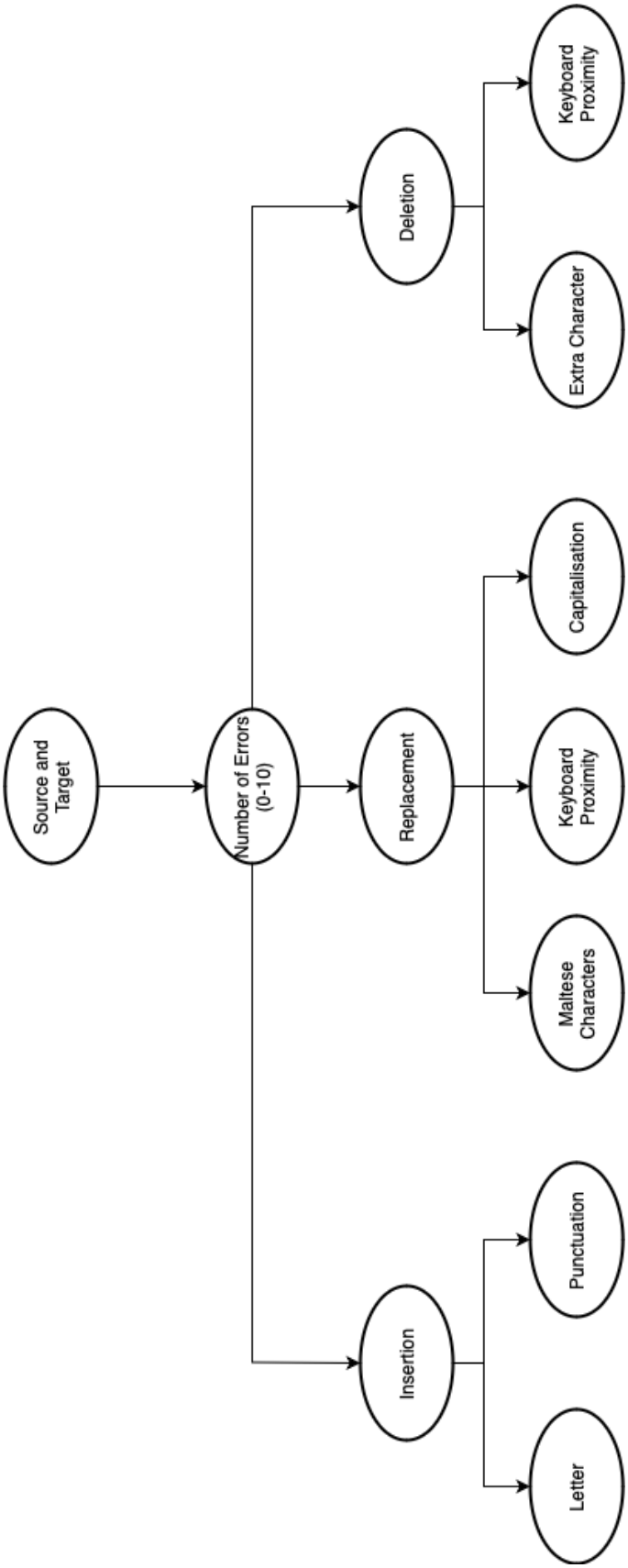


Figure 3.6 Error Creation Outline

Table 3.10 Error classification for insertion, deletion, and replacement errors with percentages of occurrence for each number of errors, ranging from 1-10 errors per sentence.

Type of Mistake, percentage	Type of error	1	2	3	4	5	6	7	8	9	10
Insertion	Letter	19%	26%	35%	47%	54%	63%	67%	65%	69%	74%
Insertion	Punctuation	81%	73%	64%	52%	46%	37%	33%	35%	30%	26%
Deletion	Extra Character	4%	13%	10%	8%	11%	8%	14%	15%	15%	14%
Deletion	Keyboard Proximity	60%	61%	65%	71%	63%	69%	71%	62%	72%	75%
Replacement	Maltese Character	35%	34%	43%	49%	49%	49%	47%	50%	42%	48%
Replacement	Keyboard Proximity	9%	5%	4%	4%	6%	6%	4%	4%	7%	5%
Replacement	Same Sounding	0%	0%	0%	0%	0.19%	0%	0%	0%	0%	0%
Replacement	Capitalisation	20%	34%	26%	20%	22%	18%	17%	17%	19%	15%

3.3.4 Creating the Synthetic Corpus

Based on the statistical analysis and the research presented in Section 2.9, a statistically informed, rule-based noising approach was chosen for synthetic corpus creation. This approach follows a methodology similar to that of Zhao *et al.* [53], where a series of deletion, insertion, replacement, and shuffling errors are introduced into each sentence. The noising techniques were guided by empirical and probabilistic observations from the authentic dataset, as detailed in section 3.3.3. The synthetic generation of source-target sentence pairs utilised sentences from the MLRS [8] corpus [74] that were not seen during the training of BERTu and mBERTu models [10].

The synthetic dataset was generated using Python. The percentages for letter insertions, punctuation insertions, extra character deletions, keyboard proximity deletions, replacements of non-Maltese characters with Maltese characters, keyboard proximity replacements, and capitalisation errors, as shown in Table 3.10, were organized into respective "buckets" for a streamlined approach, as detailed in Table 3.8.

For simplicity purposes, the error percentages were split into 3 buckets. Each bucket contains the average percentages used to represent each group: 0, 1-3 (Bucket 1), 4-6 (Bucket 2), and 7-10 (Bucket 3), the trends for each bucket is represented in figure 3.7. The code therefore selects a percentage of the dataset based on the number of errors (0-10) as shown in table 3.11. This percentage is then further divided into insertion, deletion, and replacement errors. Each of these categories is then split according to the buckets containing average values for letter insertions, punctuation insertions, extra character deletions, keyboard proximity deletions, replacements of non-Maltese characters with Maltese characters, keyboard proximity replacements, and capitalisation errors. A list of common misspellings for Maltese was compiled, this was done through searching the 'Għida bil-Malti'[9] Facebook page and listing the common spelling errors observed. The target corpus was then manually vetted for the collected misspellings, and the necessary corrections were made. The result was 515,369 source and target sentences, in which the source sentences contained statistically representative synthetic errors.

The final corpus comprises both the up-sampled authentic error set and the synthetically generated data. A basic up-sampling approach was taken in which the authentic errors were duplicated and distributed randomly throughout the dataset, totaling in 517,619 synthetic and authentic source-target sentence pairs, with 10,529,918 source tokens and 11,234,140 target tokens.

[8] https://mlrs.research.um.edu.mt/index.php?page=corpora
[9] https://www.facebook.com/ghidhabilmalti/

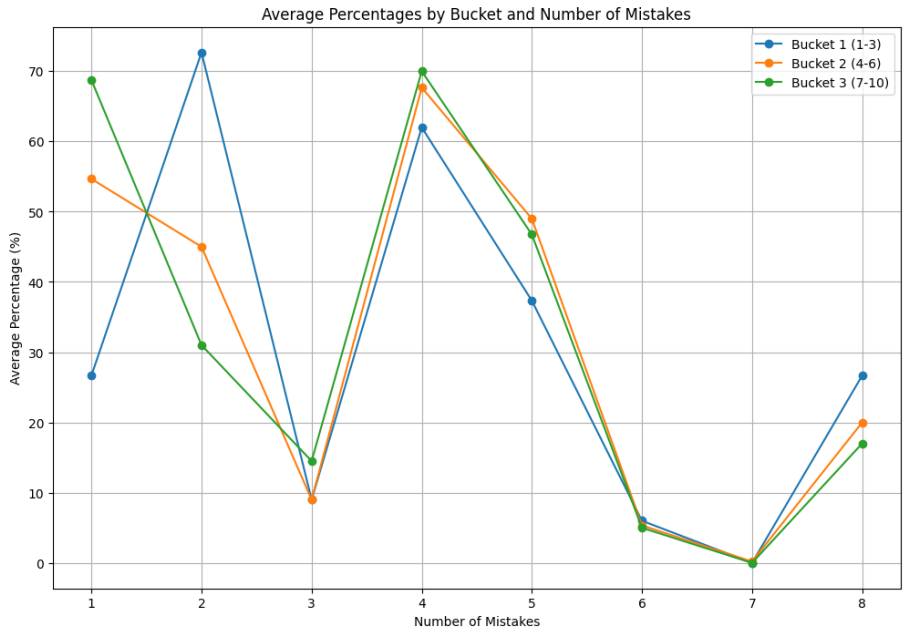


Figure 3.7 Similarity of Scores for Bucket Creation.

Table 3.11 Percentage of Errors and their Distribution by Type

Type of Error	Type of Sub-Error	Bucket 1	Bucket 2	Bucket 3
Insertion	Letter & Punctuation	26%	54%	69%
Deletion	Extra Character & Keyboard Proximity	32.47%	16.46%	61.06%
Replacement	Maltese Character & Keyboard Proximity & Capitalisation	37%	48%	48%

3.4 Final Datasets Used

The model was trained on different dataset versions, and its performance was observed. As represented in table 3.12, the different corpora used were as follows:

1. The authentic error corpus: The model was trained on the human-only corpus collected through the data collection efforts.

2. The synthetic error corpus: The model was trained on the synthetic corpus generated through a statistical observation of the authentic corpus.

3. The mixed corpus: The model was trained on the final corpus consisting of both authentic and synthetic errors.

4. Debattista [4]'s corpus: The model wad trained on the corpora used in the previous attempts at GEC for the Maltese language. The corpus included a

blend of synthetic data (57.32% of the corpus) and data sourced form the 'Qari tal-Provi' (42.68% of the corpus).

The final datasets were split into training, testing and validation sets. The training and validation sets were an 80:20 split of the datasets, whilst the test set was a 20% split of the original authentic dataset. In order to have accurate comparisons, the test set was kept constant throughout. Furthermore, the 'Qari tal-Provi' test set utilised by Debattista [4] was used as a second test set for comparison purposes.

The system was therefore trained on 4 different model configurations using 4 different datasets. The 4 encoder-decoder pairs are summarised as follows:

- BERTu and BERTu
- mBERTu and mBERTu
- BERT and BERT
- mBERT and mBERT

The above model pairs were trained on the 4 different datasets, which are summarised as follows:

- The authentic error corpus
- The synthetic error corpus
- The final, mixed corpus
- Debattista [4]'s corpus

Furthermore, each system was tested on the created authentic human error test set, and the 'Qari tal-Provi' [4] test set was used for comparison purposes.

Table 3.12 Different Corpora Used.

Dataset Version	Description	# of Sentences	Train/Val Split	# of Replacements	# of Deletions	# of Insertions
Authentic Error Corpus	Human-only corpus collected through data collection efforts.	2,251	80% / 20%	4,212	5,703	7,006
Synthetic Error Corpus	Synthetic corpus generated through statistical analysis of the authentic corpus.	513,118	80% / 20%	688,414	1,024,538	1,655,499
Mixed Corpus	Final corpus combining both authentic and synthetic errors.	517,619	80% / 20%	721,565	1,087,184	1,725,494
Debattista [4]'s Corpus	Previous GEC corpus for Maltese, blending synthetic data with Qari tal-Provi' proofreading course data.	9,373	80% / 20%	24,681	54,105	58,859

3.4.1 Tokenisation

The corpus was tokenised using the MTtokenizer [10], and word-piece tokenisation using the BERT tokeniser was applied over the word-level tokenised dataset. Three different tokenisation approaches were considered:

1. **WordPiece Tokenisation**: The original study by Guo *et al.* [59] utilised the WordPiece-tokenised IWSLT14 De-En dataset [10]. However, WordPiece tokenisation is not natively available for the Maltese language. As a result, the tokeniser struggled to effectively split Maltese words, leading to numerous unknown tokens and an incomplete vocabulary. A significant issue was the tokeniser's inability to correctly classify articles such as "il-" when attached to words.

2. **Standard Tokenisation for Maltese**: The MTWordTokenizer was considered over WordPiece tokenisation because it is specifically designed for the Maltese language. This tokenisation process involved removing punctuation and appropriately tokenising Maltese sentences.

3. **Combination of WordPiece and Standard Tokenization**: An approach combining both WordPiece and standard tokenisation was attempted. It was hypothesized that applying standard tokenisation before WordPiece tokenisation could resolve the issue of article recognition. This method proved successful, since the Maltese tokeniser appropriately splits the article and word, and consequently, a combination of both WordPiece and standard tokenisation was ultimately used.

3.5 Scoring Strategy

Debattista [4]'s approach to model evaluation included the adaptation of the ERRANT scorer [38]. As previously discussed in section 2.5, the ERRANT scorer is language specific. Debattista [4]'s scorer was therefore adapted to be able to evaluate the Maltese language. A brief outline of the scorer is as follows:

1. **Align Sentences:** The code aligns the target (reference) and predicted (translated) sentences. It returns a list of operations needed to transform the predicted sentence into the target sentence.

2. **Count Token Changes:** For each operation, the code counts how many tokens are involved:

[10] `https://sites.google.com/site/iwsltevaluation2014/data-provided`

Equal: Tokens that match between target and predicted (True Positives).

Replace: Tokens that differ between target and predicted (False Positives and False Negatives).

Insert: Tokens added in the predicted sentence but not in the target (False Positives).

Delete: Tokens missing from the predicted sentence but present in the target (False Negatives).

3. **Calculate Metrics:**

   Precision: Ratio of correctly predicted tokens to all predicted tokens.

   Recall: Ratio of correctly predicted tokens to all target tokens.

   $F_0.5$ Score: Harmonic mean of Precision and Recall, combining both metrics into one score.

In addition to the ERRANT scorer, BLEU scoring was also implemented. The BLEU scorer was selected due to its popular use in GEC evaluation [1], as well as its use in the original study performed by Guo *et al.* [59]. The metrics used to system evaluation are therefore as follows:

- **Adapted ERRANT Scorer:** this was selected in order to more accurately represent the $F_{0.5}$ scoring, and serve as a comparison metric to Debattista [4] approach, as well as the BEA-2019 shared task [29].
- **BLEU Scorer:** this was selected as a secondary metric to support the ERRANT scorer.

3.6 Technologies Used

A GPU HPC was accessed through the LT-Bridge Project (GA 952194) [11] and DFKI [12] through the Virtual Laboratory. The cluster offered a series of GPUs, the specifications of which are compiled in table 3.13. Different GPUs were used throughout the creation of the system, depending on availability.

[11] https://lt-bridge.eu
[12] https://www.dfki.de/web

Table 3.13 GPU Partition Details

Partition	Name	Arch	Mem (GB)	per Node	per Bus	per Bus (GB)	Time Limit
A100-40GB	A100-SXM4	Ampere	40	8	32	112	1 - 3 days
A100-80GB	A100-SXM4	Ampere	80	8	32	224	1 - 3 days
A100-PCI	A100-PCIE	Ampere	40	8	12	48	1 - 3 days
H100	H100 80GB HBM3	Hopper	80	8	28	224	1 - 1 day
RTX3090	RTX 3090	Ampere	24	8	12	64	1 - 3 days
RTXA6000	RTX A6000	Ampere	48	8	12	108	1 - 3 days
V100-16GB	V100-SXM2	Volta	16	8	10	64	1 - 3 days
V100-32GB	V100-SXM2	Volta	32	8	10	64	1 - 3 days
batch	RTX 6000	Turing	24	10	7	64	1 - 3 days

4 Results and Discussion

An analysis on the different model and dataset configurations set up is performed in order to determine the overall best approach. Following this, the system is compared to its predecessor [4], both in terms of data, as well as model performance. Section 4.1 provides a comparison between the different models and datasets, from which the highest performing system is selected as the final system and trained until convergence. Section 4.2 gives a quantitative and qualitative analysis of the final model; analysing its prediction capabilities and outputs. Section 4.3 provides an in-depth comparison of the system and dataset with regards to Debattista [4]'s proposed approach, including both a quantitative and qualitative comparison of the datasets and systems. An analysis of the limitations imposed by the dataset collection approach is also reviewed in section 4.4.

4.1 Model Comparison

Each of the final model systems was trained on each of the datasets over 10 epochs. The model systems surrounded the BERT, BERTu, mBERT, and mBERTu model pairs, and the datasets were the authentic, synthetic, mixed, and Debattista [4]'s. Each system was evaluated using the ERRANT scorer as described in section 3.5. The final results for span-based correction were analysed and compared, leading to the selection of the final, highest scoring model.

Table 4.1 Span-Based Correction $F_{0.5}$ Scores for Different Models across Various Datasets Over 10 Epochs

Dataset	BERT	BERTu	mBERT	mBERTu
Authentic	0.0023	0.0021	0.0022	0.0035
Synthetic	0.2153	0.0145	0.2030	0.2157
Mixed Authentic and Synthetic	0.3001	0.0139	*0.3035*	**0.3390**
Debattista [4]'s Dataset	0.0026	0.0171	0.0074	0.0590

Table 4.1 presents the $F_{0.5}$ score for each model trained on each dataset over 10 epochs. It is evident that the mBERTu model outperforms its mBERT, BERT and BERTu counterparts, producing the highest scores across all datasets, with the mBERT model being the second highest scoring, and the BERT the third. The BERTu model significantly underperforms, a result which was not expected.

Furthermore, it is evident that the mixed dataset is the overall best performing dataset, closely followed by the synthetic dataset, with the human and Debattista

[4]'s data performing poorly. The difference in performance suggests that the system performs better when trained on larger, more diverse datasets, this can be attributed to the increased variability and larger volumes of data better supporting the task of GEC. The highest performing system is therefore the mBERTu model trained on the mixed dataset, achieving a $F_{0.5}$ score of 0.3390. In order to better support this claim, the best performing model under each dataset was further trained until convergence in order to determine the best performing system, as represented in table 4.2, with the mBERTu model again outperforming its counterparts.

Table 4.2 Model Performance for Different Datasets on Converged Models

Model and Dataset	**Token-Based Detection F0.5**	**Span-Based Detection F0.5**	**Span-Based Correction F0.5**
mBERTu, Authentic Dataset	0.4951	0.3087	0.0774
mBERTu, Synth Dataset	0.6025	0.4747	0.2290
mBERTu, Mixed Dataset	**0.6638**	**0.5126**	**0.3466**
mBERTu, Debattista [4]	0.6232	0.4742	0.2523

One anomaly observed is the inconsistent behavior of the BERTu model, which scored significantly lower across datasets compared to the other models. This inconsistency can be highlighted in the loss curves presented in Figure 4.1, where the BERTu model experiences a plateau in loss early in the training process, suggesting under-fitting. Interestingly, it performed best on Debattista [4]'s dataset, achieving an $F_{0.5}$ score of 0.0171, which, while still low, was its highest. The reason behind this divergence is unclear, though it may indicate an underlying issue with the model's ability to generalise across different data types. The significant performance gap between the BERTu and mBERTu models can be attributed to mBERTu's exposure to larger and more diverse data, enhancing its capacity to extrapolate answers using cross-lingual knowledge. This broader exposure likely improves mBERTu's generalisation abilities, enabling it to identify and correct errors more effectively than BERTu, which is trained on less diverse and relatively smaller amounts of data.

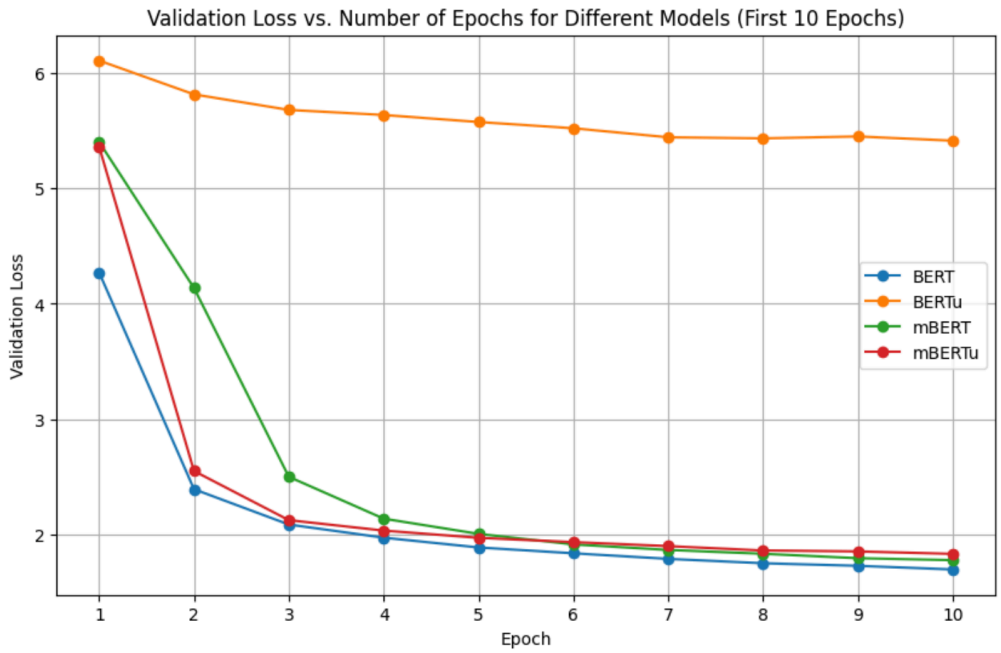


Figure 4.1 Loss per epoch for BERT, mBERT, BERTu, and mBERTu models during training

The BLEU score was also used as a secondary metric. Table 4.3 presents the BLEU score for each model trained on each dataset over 10 epochs. Upon analysis, it is evident that the BLEU scores support the same conclusions reached using the ERRANT scorer. Here, the mixed and synthetic datasets again consistently outperform the smaller datasets, furthermore, the mBERTu, mBERT and BERT models achieve the highest results, with the BERTu model significantly underperforming. Once again, the mBERTu model achieves the overall highest scores across the most datasets, and the mixed dataset is the overall highest performing dataset.

Table 4.3 BLEU Scores for Different Models across Various Datasets

Dataset	BERT	BERTu	mBERT	mBERTu
Authentic	0.0000	0.0000	0.0000	0.0000
Synthetic	0.2797	0.0000	0.2895	**0.3265**
Mixed Authentic and Synthetic	0.3278	0.0000	0.3020	*0.3074*
Debattista [4] Dataset	0.0000	0.0000	0.0000	0.0000

4.2 The Final Model

The mBERTu model was consistently the highest performing model, which can be attributed to its cross-lingual capabilities and a priori knowledge of the Maltese language. The model was trained until convergence, improving the final score.

Following this, the model was further analysed. This includes an more in-depth exploration of the final score, an analysis of the total number of errors corrected, which are split into the number of replacements, deletions and insertions, as well as an empirical analysis of the systems outputs.

4.2.1 Quantitative Analysis of the Final Model

The highest-performing model over 10 epochs was the mBERTu model trained on the mixed dataset, scoring a 0.3390 $F_{0.5}$ and 0.3265 BLEU score. The model was further trained until convergence. Table 4.4 records an improvement in the span-based correction $F_{0.5}$ score, reaching 0.3466, and a recorded BLEU score of 0.4141. The results highlight the models struggle with a large number of false negatives, referring to over-corrections, which are further analysed in section 4.2.2. Furthermore, a drop in precision and recall across the different tasks can be observed.

Table 4.4 Performance Metrics for Token-Based, Span-Based Detection, and Correction

Metric	**TP**	**FP**	**FN**	**Precision**	**Recall**	$F_{0.5}$ **Score**
Token-Based Detection	1510	364	2368	0.8058	0.3894	0.6638
Span-Based Detection	1004	475	2874	0.6788	0.2589	0.5126
Span-Based Correction	679	800	3199	0.4591	0.1751	0.3466

Furthermore, a character-level $F_{0.5}$ score was calculated. Table 4.5 shows that the model was able to accurately predict and correct errors on a character level, scoring an $F_{0.5}$ score of 0.8425, with high values for both precision and recall.

Table 4.5 Character-Level Prediction Metrics

Metric	$F_{0.5}$ **Score**
Precision	0.8372
Recall	0.8644
F0.5 Score	0.8425

To better understand and quantify the capabilities of the final model, the predicted sentences outputted were compared to their respective source and target sentences. The total number of replacement, deletion and insertion corrections, and the number of predicted corrections was calculated. As depicted in table 4.6, the model effectively identified all 3 types of errors, with 576 successfully predicted insertion errors, 717 deletion errors, and 226 replacement errors. The

model encountered the most difficulty with replacement errors, and performed best in predicting deletion errors.

Table 4.6 Total and Predicted Replacements, Deletions and Insertions on the authentic test set

	Replacements	**Deletions**	**Insertions**
Total Amount	966	1641	1507
Predicted Amount	226 (23.4%)	717 (43.7%)	576 (38.2%)

Whilst performing better than previous GEC attempts, the number of predicted corrections still remains relatively low, suggesting the need for further system refinements. Despite this, the results show the models ability to grasp context within the sentences, especially in terms of character deletions, correcting 43.7% of all deletion corrections.

4.2.2 Qualitative Analysis of the Final Model

An empirical exercise was conducted comparing the models predictions against the source and target sentences for correct and incorrect instances taken from the test set. Tables 4.7 and 4.8 display these comparisons.

Table 4.7 presents successful corrections made by the model. The first row illustrates an instance where no correction was necessary, which the model accurately recognised. The second and third rows show cases where the model successfully identified and corrected Maltese characters, substituting ‘gh’ with ‘għ’, ‘h’ with ‘ħ’, and ‘g’ with ‘ġ’. Notably, the model managed to recognise and apply multiple corrections in a single instance. The fourth row demonstrates a capitalisation correction, where the model correctly capitalised the first word of a sentence.

Table 4.7 Source, Target, and Correct Predictions

No.	Source	Target	Predictions	Correction Type
1	U le ma tasalx	U le ma tasalx	U le ma tasalx	No correction
2	Jista jghid min huma	Jista jgħid min huma	Jista j**għ**id min huma	Replacement, Maltese character
3	Dik hija xi haga kompletament differenti	Dik hija xi ħaġa kompletament differenti	Dik hija xi **ħ**a**ġ**a kompletament differenti	Replacement, Maltese character
4	fuq hekk messu jkellimna Andrew Borg Cardona	Fuq hekk messu jkellimna Andrew Borg Cardona	**F**uq hekk messu jkellimna Andrew Borg Cardona	Replacement, capitalisation

Despite these successes, the model faced significant challenges, particularly regarding hallucinations, where it either failed to complete a sentence or generated extra words or characters after completion. This issue could stem from underfitting during training. Table 4.8 illustrates several incorrect or incomplete predictions made by the model. The first row showcases an example of overcorrection, where unnecessary 'għ' were added to the sentence. Rows two and three reflect cases where correct corrections were made, but the model then struggled with hallucinations, leading to the unnecessary repetition of parts of the sentence after its completion. The fourth row presents an instance of an unnecessary correction, where a word was swapped for another that, while contextually sensible, was incorrect.

Table 4.8 Source, Target, and Predictions

No.	Source	Target	Predictions	Error Type
1	Allura kif tħares lejha	Allura kif tħares lejha	Allura kif tħares lej**għa tgħ**	Model hallucination
2	Minflok isir tigi jinghata	Minflok isir tiġi jingħata	Minflok isir tiġi jingħata **jing**	Model hallucination
3	imbaghad kont immur ghandu	Imbagħad kont immur għandu	Imbagħad kont immur għandu **għandugħad imbagħad kont immur għandu**	Model hallucination and overcorrection
4	Li hu miktub ma nistax immerih	Li hu miktub ma nistax immerih	Li hu miktub ma nistax **ngħid**	Overcorrection

Overall, the results suggest that the model was consistent at correcting Maltese characters and capitalisation errors. This supports the models contextual understanding of sentences, as well as a baseline understanding of the Maltese language. This could also be supported through the incorrect predictions, such as row four in table 4.8, where the substitution, although unnecessary, was contextually correct. Similar to its predecessor [4], however less frequently, it can be observed that the models largest struggle was hallucinations and over-corrections. Despite this, the model displays a baseline understanding of the language and task, and was able to correct a number of errors specific to Maltese.

4.2.3 Training mBERTu on a Combined Dataset

As described in section 4.1, the model performed best with larger datasets, and displayed reduced performance with smaller datasets. Building upon this, an experiment was carried out in which the mixed corpus was combined with Debattista [4]'s corpus, as highlighted in table 4.9. The final model, consisting of the mBERTu model system, was then trained on the combined dataset, and the results were analysed.

Table 4.9 Different Corpora Used, Including the Combined Corpus.

Dataset Version	**Description**	**# of Sentences**	**Train/Val Split**	**# of Replacements**	**# of Deletions**	**# of Insertions**
Authentic Error Corpus	Human-only corpus collected through data collection efforts.	2,251	80% / 20%	4,212	5,703	7,006
Synthetic Error Corpus	Synthetic corpus generated through statistical analysis of the authentic corpus.	513,118	80% / 20%	688,414	1,024,538	1,655,499
Mixed Corpus	Final corpus combining both authentic and synthetic errors.	517,619	80% / 20%	721,565	1,087,184	1,725,494
Debattista [4]'s Corpus	Previous GEC corpus for Maltese, blending synthetic data with Qari tal-Provi' proofreading course data.	9,373	80% / 20%	24,681	54,105	58,859
Combined Corpus	**Combined corpus containing the Mixed Corpus and Debattista [4]'s Corpus**	**526,992**	**80% / 20%**	**746,246**	**1,141,289**	**1,784,353**

Table 4.10 displays a decrease in overall performance when the final mBERTu system was trained on the combined dataset. The model scored an 0.2728 $F_{0.5}$, compared to the final 0.3466 $F_{0.5}$. The reason for the decrease in performance could be a result of the contrasting error types presented within the two datasets, which could potentially create inconsistencies when combined. This could have potentially caused the model to struggle with its predictions.

Table 4.10 Performance Metrics on Combined and Mixed Corpora

Metric	**Final Model on Mixed**			**Final Model on Combined**		
	Precision	**Recall**	$F_{0.5}$ **Score**	**Precision**	**Recall**	$F_{0.5}$ **Score**
Token-Based Detection	0.8058	0.3894	0.6638	0.8281	0.3840	0.6725
Span-Based Detection	0.6788	0.2589	0.5126	0.6235	0.2272	0.4622
Span-Based Correction	0.4591	0.1751	**0.3466**	0.3680	0.1341	0.2728

4.3 A Comparative Study

The developed system is contrasted alongside Debattista [4], through in-depth analyses of both the dataset as well as the final systems. The dataset comparison considers the size and quality difference of both the test sets, and final datasets, as well as the overall error analysis. Finally, the two systems are compared as stand alone systems.

4.3.1 Performance Evaluation on the Different Datasets

A replication effort in which Debattista [4]'s model was trained on the different created datasets in order to observe its performance was carried out. The model was trained over 500 iterations, and was tested on the authentic test set. Table 4.11 shows that the mixed dataset outperformed all synthetic, authentic and Debattista [4]'s datasets.

Table 4.11 Results on Debattista [4]'s Model using the authentic test set

Dataset	**Precision**	**Recall**	**F0.5 Score**
Authentic dataset	0.0428	0.0167	0.0326
Debattista [4] dataset	0.0503	0.0439	0.0489
Synthetic dataset	0.0470	0.0192	0.0365
Mixed dataset	0.0751	0.0373	**0.0625**

An analysis of the size difference between the Qari tal-Provi and mixed datasets

was performed. As depicted in table 4.12, the created dataset is significantly larger than the predecessor, including 502,395 more sentences, with 11,236,743 additional source tokens and 11,238,989 additional target tokens.

Table 4.12 Corpus Statistics: Source and Target Sentences and Tokens

Corpus	Number of Sentences	Tokens (Source/Target)
Mixed Corpus	517,619	10,529,918 / 11,234,140
Debattista [4]'s Corpus	16,485	288,760 / 289,094

We wanted to deduce whether the mixed dataset success was an attribute of its size, or quality, or both. In order to answer this, the mixed dataset's size was reduced in order to match the size of Debattista [4]'s dataset, totaling to 9,373 sentences. The idea surrounding this being that the comparison would be only on a basis of quality, rather than quality and size. All systems were again trained on both the reduced size dataset and Debattista [4]'s dataset. Tables 4.13 and 4.14 support the fact that the mixed corpus outperforms previous corpora both in terms of size as well as quality, showing it outperforming its counterpart over all systems.

Table 4.13 Span-Based Correction $F_{0.5}$ Score, Precision, and Recall on the Different Models trained on the Reduced Mixed Dataset and Debattista [4]'s Dataset

Model Type	Trained On	Precision	Recall	$F_{0.5}$ Score
mBERTu	Reduced Mixed Dataset	0.0893	0.0423	**0.0730**
BERTu	Reduced Mixed Dataset	0.0311	0.0185	0.0274
BERT	Reduced Mixed Dataset	0.0298	0.0146	0.0246
mBERT	Reduced Mixed Dataset	0.0186	0.0115	0.0166
mBERTu	Debattista [4]'s Dataset	0.0702	0.0360	**0.0590**
BERTu	Debattista [4]'s Dataset	0.0217	0.0093	0.0171
BERT	Debattista [4]'s Dataset	0.0044	0.0010	0.0026
mBERT	Debattista [4]'s Dataset	0.0129	0.0028	0.0074

Table 4.14 Span-Based Error Correction Precision, Recall, and $F_{0.5}$ Score on Debattista [4]'s Model

Dataset	Precision	Recall	$F_{0.5}$ Score
Debattista [4] dataset	0.0503	0.0439	0.0489
Reduced Mixed dataset	0.0888	0.0516	**0.0776**

The mixed dataset is therefore an improvement from previous datasets in terms

of both increased size and improved quality. This is further supported through the increased precision, recall and $F_{0.5}$ scores achieved by the reduced size dataset.

4.3.2 Performance Evaluation on Different Test Sets

The created systems and Debattista [4]'s system were tested on both the authentic and 'Qari tal-Provi' [4] test sets. The results recorded in table 4.15 for the created models and for Debattista [4]'s models, differ from the ones initially presented in table 4.1 and 4.11 respectively, with an overall drop in performance, and the BERT model being the highest performing model. Despite this, the mBERTu model maintains overall strong performance, and the mixed dataset remains the overall best performing dataset. The difference in scores can be attributed to the different contexts the two test sets exist within. As represented in table 4.16, Debattista [4]'s test set was sourced from a proof reading course exercise aimed at students receiving training in spotting difficult and complex spelling mistakes, implying that the errors are more deliberate and therefore likely more difficult to recognise. Contrastingly, the errors collected through the data collection campaign are more akin to errors often corrected by 'autocorrect' systems in day-to-day typing. Therefore, in the context of more formalized errors, the system significantly under-performs.

Table 4.15 Span-Based Correction F0.5 Scores for Different Models across Various Datasets on the 'Qari tal-Provi' [4] Test Set

Dataset	BERT	BERTu	mBERT	mBERTu	Debattista [4]
Authentic	0.0009	0.0009	0.0009	0.0038	0.0164
Synthetic	0.1207	0.0044	0.1156	0.0718	0.0254
Mixed Authentic and Synthetic	**0.1442**	0.0076	*0.1341*	0.1082	0.0231
Debattista [4]'s Dataset	0.0012	0.0105	0.0033	0.0399	0.0072

Table 4.16 Comparison of Errors from the Qari tal-Provi Dataset and the Authentic Dataset

Qari tal-Provi Corpus		Authentic Corpus	
Source Text	**Target Correction**	**Source Text**	**Target Correction**
Id-Demanju Pubbliku jimxi wkoll id f'id mal-access għall-pubbliku.	Id-demanju pubbliku jimxi wkoll id f'id mal-aċċess għall-pubbliku.	iun nies ivvutaw fuq hekk	In-nies ivvutaw fuq hekk.
"Jien kont infurmat b'dawn il-pjani gimgha ilu", sostna Mario Calleja.	"Jien kont infurmat b'dawn il-pjani ġimgħa ilu", sostna Mario Calleja.	Diċembru Żahra ġġġħa jkollu konfront ieġħor ma kickboxer mill-Irlanda ta' Fuq.	F'Diċembru Zahra se jkollu konfront ieħor ma' kickboxer mill-Irlanda ta' Fuq.

In order to more closely observe the test sets, an exercise was carried out in which the source and target sentences of each test set was compared. The total number of insertion, deletion and replacement errors present in each was counted, as displayed in table 4.17. It is evident that Debattista [4]'s set consists of less sentences yet contains a larger amount of overall insertion, and deletion errors, and significantly less replacements errors. This can be attributed to the nature of the proofreading exercise carried out. The difference in insertion, deletion and replacement errors further explains the models' reduced performance observed in table 4.15, in which the model likely struggled with a higher ratio of errors per sentence.

Table 4.17 Overall Counts for Insertions, Deletions, and Replacements

Category	Insertions	Deletions	Replacements	Size in Sentences	Source Characters	Target Characters
'Qari tal-Provi' [4] Test Set	5393	4041	230	495	51038	52390
Authentic Test Set	966	1641	1507	563	25639	25505

4.3.3 Performance Evaluation on the Different Models

The final system, utilising the mBERTu models, fine-tuned on the mixed dataset, achieved a final $F_{0.5}$ score of 34.66%, outperforming its predecessor [4]. To investigate whether this improvement is solely a result of the improved dataset, or

rather a combination of an improved dataset and improved system architecture, an experiment was conducted. Specifically, Debattista [4]'s model was trained for 20,000 epochs on the reduced size mixed dataset, previously identified as the highest-performing dataset in Sections 4.3.1 and 4.3.2 as well as trained on Debattista [4]'s dataset. The model was trained under the same conditions as the best-performing model described in Debattista [4]'s study. The aim of this experiment was to determine if Debattista [4]'s model could achieve competitive performance when trained on more representative data. The results are summarised in Table 4.18, which displays the final performance metrics for each model on the authentic test set.

Table 4.18 Span-Based Error Correction Precision, Recall, and F0.5 Score on Debattista [4]'s Model and the Final Model on the Authentic Test Set

Model	Trained On	Precision	Recall	F0.5 Score
mBERTu Model	**Mixed Dataset**	0.4591	0.1751	0.3466
Debattista [4]'s Model	**Debattista [4]'s Dataset**	0.2674	0.1637	0.2374
Debattista [4]'s Model	**Reduced Mixed Dataset**	0.2621	0.1606	0.2327

The results in Table 4.18 show that Debattista [4]'s model performs comparably on both its original dataset and the reduced mixed dataset when evaluated on the authentic test set. Notably, Debattista [4]'s model trained on the reduced mixed dataset does not outperform the model trained on Debattista [4]'s dataset. To further examine these findings, Table 4.19 presents the models' performances when evaluated on the 'Qari tal-Provi' [4] test set.

Table 4.19 Span-Based Error Correction Precision, Recall, and F0.5 Score on Debattista [4]'s Model and the Final Model on the 'Qari tal-Provi' [4] Test Set

Model	Trained On	Precision	Recall	F0.5 Score
Final mBERTu Model	**Mixed Dataset**	0.1014	0.1486	0.1082
Debattista [4]'s Model	**Debattista [4]'s Dataset**	0.1747	0.0913	0.1477
Debattista [4]'s Model	**Reduced Dataset**	0.0807	0.0339	0.0632

Table 4.19 demonstrates that Debattista [4]'s model trained on the reduced mixed dataset performs worse than when trained on Debattista [4]'s dataset.

Overall, the presented results suggest that the improved dataset alone does not account for the performance gains observed in the final mBERTu model. The mBERTu model fine-tuned on the mixed dataset significantly outperforms Debattista [4]'s model on the authentic test set, and produces competitive results on the 'Qari tal-Provi' test set [4], indicating that the improvement is likely due to the

model architecture or training strategies, rather than the dataset used for training alone.

4.3.4 Error Analysis

The mixed synthetic and authentic corpus incorporates insertion, deletion and replacement errors, as represented in table 3.8. These errors can empirically be extended into more specific error types, including those reported by Debattista [4]:

Table 4.20 Types of Errors and Examples Captured

Type of Error	Source	Prediction
Omission errors	Filwaqt li rrikonoxxiet li **bis-saha** tas-social workers u tal-probation officers fil-habs sar titjib kbir favur il-benessri tal-priġunieri, l-avukat qalet ukoll li gad fadal afna xi jsir biex il-ħabsin – l-iżjed dawk illi jiddis-pjaihom ta' għemilhom – tassew ikunu jistgħu jirrijabilitaw ruħhom bis-sħiħ.	Filwaqt li rrikonoxxiet li **bis-saħħa** tas-social workers u tal-probation officers fil-ħabs sar titjib kbir favur il-benessri tal-priġunieri, l-avukat qalet ukoll li għad fadal ħafna xi jsir biex il-ħabsin – l-iżjed dawk illi jiddisp-jaċihom ta' għemilhom – tassew ikunu jistgħu jirrijabilitaw ruħhom bis-sħiħ.
Transposition errors	**Tnat** kemm waqaw bic-ciet mis-saqaf li nkixfet ix-xibka	**Tant** kemm waqgħu biċċiet mis-saqaf li nkixfet ix-xibka.
Insertion errors	**Il-vizzjju** tadd-droga rikeb lis-sojeta taghna	**Il-vizzju** tad-droga rikeb lis-soċjetà tagħna.
Keyboard Proximity er-rors	**iun nies** ivvutaw fuq hekk	**In-nies** ivvutaw fuq hekk.
Duplicate Character er-rors	B'hekk tinżamm **il-privatezzza** tal-persuna	B'hekk tinżamm **il-privatezza** tal-persuna.
Silent Letter errors	Kien jigri li minflok konna **nerqu** fl-ilma konna nerqu fl-gharawq	Kien jiġri li minflok konna **negħrqu** fl-ilma konna negħrqu fl-għaraq.
No Font errors	Jekk intom itxtiequ tarawha ghandi kopja **taghha** hawnhekk.	Jekk intom tixtiequ tarawha għandi kopji **tagħha** hawnhekk.
Common Article errors	Nafu li l-akbar sdifa **hi** dik ta' l-AIrMalta	Nafu li l-akbar sfida **hija** dik tal-Air Malta.
Corruption Noising	**lgg** lġġifieri naħseb li hija ċara.	Jiġifieri naħseb li hija ċara!

In addition to the errors reported by Debattista [4] in table 4.20, the dataset scope was extended through the data collection carried out to represent new error types, as displayed in table 4.21.

Table 4.21 Further Types of Errors and Examples Captured

Type of Error	**Source**	**Target**
Confusion of similar words	**Kolox sew Sur**	**Kieku sew!**
Phonetic errors, including phonetic transfer	Qed **nghajdu** bejn stati ġirien .	Qed **ngħidu** bejn ” Stati ġirien ”.
Influence of other languages	Tmien sewwieqa oħra kienu qed jużaw il-**mobile** waqt is-sewqan	Tmien sewwieqa oħra kienu qed jużaw il-**mowbajl** waqt is-sewqan.
Punctuation related errors	Fil-fatt **iż-=Żaqq** li jdoqq kienet ta’ gidi maqtul minnu stess.	Fil-fatt, **iż-żaqq** li jdoqq kienet ta’ gidi maqtul minnu stess.
Slang	Hu mifhum ukoll li sal-Ħadd li **adda** sar kuntatt ma’ David abela, għalhekk kollox jindika li huwa miet nhar it-Tnejn.	Hu mifhum ukoll li sal-Ħadd li **għadda** sar kuntatt ma’ David Abela, għalhekk kollox jindika li huwa miet nhar it-Tnejn.
Spelling influence errors	Jien u żewġi stajna **noqghodu** filt-taqsima talkura intensiva.	Jien u żewġi stajna **noqogħdu** fit-Taqsima tal-Kura Intensiva.

4.4 Limitations Imposed by the Data Collection

The data collection resulted in 2,814 sentences containing authentic human errors. These errors were collected through typing out sentences in a text box, without using the backspace button or any supporting spell checkers or auto correct. The final result was a number of varied spelling errors similar to those exposed to ‘autocorrect’ and spell checking systems. Despite this, the collection imposed a set of limitations. This section gives an account of the errors that were captured successfully and those which were not captured due to the nature of the collection.

Errors Successfully Captured

The data collection and synthesis were able to capture several types of errors [80], including:

- **Phonetic Errors**: Errors involving homophones and words that sound similar.
- **Confusion of Similar Words**: Misuse of words that are easily confused due to their similarity in form or meaning.
- **Misapplication of Spelling Rules**: Errors arising from incorrect application of spelling rules, including:
  - Omission errors
  - Commission errors
  - Transposition errors
  - Rule-based errors
- **Typographical Errors**: Mistakes caused by keyboard proximity or accidental key presses.
- **Influence of Other Languages**: Errors resulting from interference or influence from other languages.
- **Technological Influences**: Errors due to auto-correct and predictive text features.
- **Punctuation-related Errors**: Mistakes in punctuation, including issues with hyphenation and accent marks.
- **Slang**: Informal spellings commonly used in casual communication, such as replacing the maltese 'għ' with an 'a' (e.g. "adna" instead of "għadna").

Errors Not Captured

Due to the nature of the exercise, certain morphological errors were not accounted for. This is mainly attributed to the fact that all sentences were spoken in correct Maltese. A few examples of errors that were unaccounted for are:

- **Morphological Errors**
  - **Inflectional Errors**: Incorrect use of tense, number, or other grammatical forms (e.g.,"Huma ghandu" instead of "Huma għandhom" (they have)).

  - **Derivational Errors**: Incorrect use of prefixes or suffixes (e.g. “Kiser” and “Kisser” being used interchangeably, where one is in the first form and the other is in second form respectively).

- **Non-standard Sentences**

  - **Text Speak**: non-standard sentence structures used in day-to-day texting (e.g. “pls ċemplilni meta tasal.” instead of “Jekk jogħġbok ċemplili meta tasal.”).

Furthermore, certain errors specific to bilingualism were not captured. Some errors found in the Maltese language can be attributed to bilingual tendencies, such as code-switching and code-mixing. Code-switching and code-mixing refer to the mixed use of language, which in this case is Maltese and English. Due to the nature of the collection, code-switching errors, grammar transfer errors, direct translation errors and invented forms errors were unable to be accounted for.

- **Transfer Errors**:
  - **Phonetic Transfer**: Errors due to the influence of the phonetic patterns of another language.
  - **Orthographic Transfer**: Errors arising from the influence of another language’s writing system.
- **Code-Switching Errors**: Errors resulting from the mixing of languages within a sentence or text.
- **Spelling Influence**: Errors caused by the spelling conventions of another language.
- **Grammar Transfer**: Errors due to the influence of grammatical rules from another language.
- **Direct Translation**: Errors from translating phrases directly from one language to another.
- **Invented Forms**: Creation of new forms that do not adhere to the standard rules of either language.

4.5 Main Takeaways

In summary, the mBERTu model for both encoder and decoder components trained on a mixed dataset showed the most promising results. The final system outperforms its predecessor [4], with a final $F_{0.5}$ of 34.66% compared to the previous

$F_{0.5}$ of 31.84%. The improvements can be attributed to both the overall system architecture, as well as the introduction of a more representative, and larger corpus. It is worth noting that the created corpora and Debattista [4]'s corpus differ in context, with the created mixed corpus being more representative of day-to-day errors, whilst Debattista [4]'s corpus being more formal, since it was modeled after the 'Qari tal-Provi' proofreading course. The final system was consistent in recognising and correcting a series of error types, including non-Maltese to Maltese characters and capitalisation. Despite this, there is still room for improvement, with a significant number of errors going unnoticed, as calculated in table 4.6.

5 Conclusion

The final system and dataset offer a step forward in spell checking for the Maltese language. The final $F_{0.5}$ score of 34.66% suggests that there is more work to be done to reach standards akin to higher resourced languages. The proposed conclusion firstly revisits the proposed aims and objectives in section 5.1, following this, the next steps to be taken are identified in section 5.2, and finally, a few concluding remarks are given in section 5.3.

5.1 Revisiting the Aims and Objectives

The aims and objectives were initially outlined in section 1.2. These objectives are revisited, and an analysis on how each one was fulfilled is provided.

1. **The First Objective: Collect a diverse set of authentic human errors to construct a more representative corpus that accurately reflects the range of errors found in real-world typing. Develop a statistically informed synthetic dataset, based on the authentic dataset.**

   This objective was achieved. The data collection campaign was successful, over which a total of 2,890 sentences were collected, of which, 2,814 were considered valid through manual vetting. Following this, a statistical analysis of the dataset was conducted, informing the creation of a synthetic corpus, totaling to 513,118 sentences. The final mixed corpus is the combination of the synthetic and authentic datasets, which totals to 515,369 sentences without up-sampling. Therefore, this study produced two main datasets, the authentic dataset, and the synthetic dataset, which were then combined to create the mixed dataset, consisting of both authentic and synthetic data. The mixed dataset was consistently the highest performing overall, due to its larger size, as well as its quality. The mixed dataset is the largest, most diverse and representative corpus available for GEC for the Maltese language, consistently outperforming its predecessor [4] in both size and overall quality.

2. **The Second Objective: Replicate existing work to serve as a baseline for Maltese GEC.**

   This objective was achieved. Previous work on GEC for the Maltese language conducted by Debattista [4] was replicated as a baseline approach. The final approach replicated and adapted Guo *et al.* [59]'s system, in which BERT models are incorporated in parallel sequence decoding through the use of adapters. Before landing on the final system, Kaneko *et al.* [12]'s approach for pipe-lining encoder-decoder models and pre-trained MLMs was attempted, as well as Vázquez *et al.* [60]'s approach for addressing the differences between BERT and MT encoder spaces for translation tasks. The final approach was selected due to: successful replication efforts, the systems lightweight and adaptable structure, through which each component can be considered as a plug-in unit, and the fact that it leverages BERT for both the encoder and decoder components, allowing better leveraging of pre-trained Maltese models.

3. **The Third Objective: Develop an approach for the Maltese language, leveraging the capabilities of pre-trained Maltese models. Train and fine-tune the created GEC system for the Maltese language using the newly created corpora and employ effective techniques tailored for low-resource environments.**

   This objective was achieved. A series of different encoder-decoder model ensembles were compared in order to reach the final, optimal system, consisting of the mBERTu model for both encoder and decoder components. The ensembles incorporated BERT, BERTu, mBERT and mBERTu models. The different model ensembles were trained on a series of different datasets, namely the authentic, synthetic and mixed corpora, created within objective two, as well as the 'Qari tal-Provi' and synthetic dataset utilised in previous work on GEC for the Maltese language by Debattista [4]. The mBERTu model trained on the mixed dataset was selected as the optimal model. Alongside training, a series of hyper-parameter tuning exercises were performed in order to boost the model performance. This included fine-tuning of the BERT models, fine-tuning the adapter modules, and fine-tuning the overall system. The most effective techniques employed included reducing the overall model size due to underfitting. This significantly reduced loss plateau which in turn reduced model hallucination. Furthermore, the incorporation of hyper-parameters such as drop-out and weight decay [4], as well as the implementation of the cosine learning rate scheduler all had positive contributions to the models overall performance. The final model scored an $F_{0.5}$

score of 34.66%.

4. **The Fourth Objective: Evaluate previous systems for Maltese GEC. Conduct a comprehensive comparison of the final approach with previous systems.**

   This objective was achieved. A comprehensive comparison between both systems was conducted in order to determine the best performing dataset, and the best performing system. In order to compare the final mixed dataset and Debattista [4]'s dataset in terms of quality, an exercise was carried out in which the mixed dataset was reduced to match Debattista [4]'s dataset size. All models were then trained on both the reduced dataset and Debattista [4]'s dataset, in which the reduced dataset consistently outperformed its predecessor. The final dataset therefore improves on previous available datasets in both size as well as quality. Furthermore, a qualitative analysis of both datasets was performed, concluding that the final mixed dataset was able to capture a larger amount of errors, therefore making it more detailed, and better representative of errors within the Maltese language. Furthermore, the two models were compared, Debattista [4]'s final system recorded an $F_{0.5}$ score of 31.84%, whilst the final system proposed recorded an $F_{0.5}$ score of 34.66%. In order to understand whether the increased score is attributed to the dataset or the overall system architecture, Debattista [4]'s model was further trained until convergence on the reduced dataset, in which the final mBERTu model again outperformed its predecessor. It can therefore be concluded that the final system outperforms its predecessor in all domains.

5.2 Future Work

The efforts made within this study build upon the gaps identified in previous research in GEC for the Maltese language [4] [6]. This study establishes a clear path for future developments and improvements for Maltese GEC. The current system can be built upon both in terms of data, as well as system architecture.

5.2.1 Data Improvements

The corpus created can be further improved, both in terms of size, as well as quality. The data collection campaign could potentially be better established in order to collect more data from more diverse sources. This includes improving the interface front-end, making it more appealing, as well as improving the back-end systems in order to ensure the seamless collection of larger amounts of data. Furthermore, the data collection can be extended to include more helpful data points such as

demographic information, in order to ensure that the data collected is well representative of real-life errors. This could be extended further to include a mobile application that supports model testing and data collection.

Furthermore, the nature of the collection has room for improvement, since certain common Maltese errors are outside of the capabilities of the current system, as defined in section 4.4. Therefore, new methods in data collection, such as web scraping, or the inclusion of text message or chat-form data could lead to a more well-rounded corpus. A more extensive integration of the 'Qari tal-Provi' dataset could also be attempted.

Finally, the creation of the synthetic dataset could benefit from more experimentation with different noising techniques as defined in section 2.9, this could include back-propagation, neural noising, and other techniques for comparison and optimization.

5.2.2 System Architecture Improvements

The current system makes use of the following model pairs:

- BERTu and BERTu
- mBERTu and mBERTu
- BERT and BERT
- mBERT and mBERT

In order to observe whether mixing the models has any significant improvements, as well as to determine if certain models perform better on the encoder or decoder side of the system, we propose extending the model combinations to include different pairings such as:

- BERTu and mBERT
- mBERTu and BERTu
- mBERT and BERT
- BERT and mBERT

Since the system is lightweight, it allows for seamless customisation; therefore, different models outside of the BERT architecture can also be experimented with. Furthermore, attempting to pre-train the models on Maltese GEC data could potentially improve the system's performance [48]. This approach may enhance the models' ability to detect and correct errors by exposing them to a wider variety of linguistic structures and error patterns.

5.3 Final Remarks

The final system and corpus presented represent a significant advancement in spell-checking for the Maltese language, introducing a state-of-the-art, adaptable framework for Maltese GEC along with the first comprehensive annotated error corpus for Maltese.

The corpus developed is, to date, the largest annotated error corpus for Maltese. It incorporates authentic errors collected through a data collection campaign, supplemented by synthetic data generated through statistical modeling of these errors. The corpus comprises 517,619 sentence pairs of incorrect and corrected sentences. Additionally, an authentic error test set containing 563 sentence pairs was created from the collected authentic errors. The system outperformed previous models [4][6], achieving a final $F_{0.5}$ score of 34.66%.

The system design relies on pre-trained Maltese models, BERTu and mBERTu, configured in an encoder-decoder structure based on the approach proposed by Guo *et al.* [59]. The top-performing configuration utilised mBERTu for both the encoder and decoder components. The system underwent extensive tuning, including optimising hyperparameters such as dropout and weight decay to adapt to low-resource environments [4]. The lightweight nature of the system also allows for future experimentation with a variety of model combinations.

The model demonstrated strong performance in error correction, accurately resolving approximately 23.4% of replacement errors, 43.7% of deletion errors, and 38.2% of insertion errors. It effectively identified and corrected fundamental Maltese language errors, including capitalisation errors and substitutions of English characters with their Maltese equivalents. It also occasionally handled more complex corrections, such as replacing ‘qhallura’ with ‘allura’, suggesting a foundational understanding of Maltese linguistic rules.

However, the model showed limitations, particularly with hallucinations, where sentences were sometimes left incomplete, or words were unnecessarily repeated within a sentence. These issues may be attributed to the low-resource nature of Maltese language data, which, despite the improvements made, remains limited compared to higher-resourced languages like English [12]. Ultimately, the work carried out provides a foundation for future work, in which the collection campaign can be further expanded, and the system can be adapted to involve new model pipelines and more extensive training.

References

[1] C. Bryant, Z. Yuan, M. R. Qorib, H. Cao, H. T. Ng, and T. Briscoe, "Grammatical error correction: A survey of the state of the art," *Computational Linguistics*, pp. 643–701, Sep. 2023. DOI: 10.1162/coli_a_00478. [Online]. Available: https://aclanthology.org/2023.cl-3.4.

[2] J. C. Park, M. Palmer, and C. Washburn, "An English grammar checker as a writing aid for students of English as a second language," in *Fifth Conference on Applied Natural Language Processing: Descriptions of System Demonstrations and Videos*, Washington, DC, USA: Association for Computational Linguistics, Mar. 1997, pp. 24–24. DOI: 10.3115/974281.974296. [Online]. Available: https://aclanthology.org/A97-2014.

[3] J. Náplava, M. Straka, J. Straková, and A. Rosen, "Czech grammar error correction with a large and diverse corpus," *Transactions of the Association for Computational Linguistics*, vol. 10, B. Roark and A. Nenkova, Eds., pp. 452–467, 2022. DOI: 10.1162/tacl_a_00470. [Online]. Available: https://aclanthology.org/2022.tacl-1.26.

[4] A. Debattista, "Error-Checking for Maltese," M.S. thesis, University of Malta, Msida, Malta, Feb. 2023.

[5] K. Omelianchuk, V. Atrasevych, A. Chernodub, and O. Skurzhanskyi, "GECToR – grammatical error correction: Tag, not rewrite," in *Proceedings of the Fifteenth Workshop on Innovative Use of NLP for Building Educational Applications*, J. Burstein *et al.*, Eds., Seattle, WA, USA, Online: Association for Computational Linguistics, Jul. 2020, pp. 163–170. DOI: 10.18653/v1/2020.bea-1.16. [Online]. Available: https://aclanthology.org/2020.bea-1.16.

[6] R. Mizzi, "The development of a statistical spell checker for Maltese," M.S. thesis, Msida, Malta, 2000. [Online]. Available: https://www.um.edu.mt/library/oar/handle/123456789/92160.

[7] C. Bryant and T. Briscoe, "Language model based grammatical error correction without annotated training data," in *Proceedings of the Thirteenth Workshop on Innovative Use of NLP for Building Educational Applications*, J. Tetreault, J. Burstein, E. Kochmar, C. Leacock, and H. Yannakoudakis, Eds., New Orleans, Louisiana: Association for Computational Linguistics, Jun. 2018, pp. 247–253. DOI: 10.18653/v1/W18-0529. [Online]. Available: https://aclanthology.org/W18-0529.

[8] M. Rosner and C. Borg, "Language Report Maltese," in *European Language Equality: A Strategic Agenda for Digital Language Equality*, G. Rehm and A. Way, Eds. Cham: Springer International Publishing, 2023, pp. 183–186, ISBN: 978-3-031-28819-7. DOI: 10.1007/978-3-031-28819-7_27. [Online]. Available: https://doi.org/10.1007/978-3-031-28819-7_27.

[9] V. Marmara', "Stħarrig Dwar L-Istat Tal-Ilsien Malti," Feb. 2024.

[10] K. Micallef, A. Gatt, M. Tanti, L. van der Plas, and C. Borg, "Pre-training data quality and quantity for a low-resource language: New corpus and BERT models for Maltese," in *Proceedings of the Third Workshop on Deep Learning for Low-Resource Natural Language Processing*, C. Cherry *et al.*, Eds., Hybrid: Association for Computational Linguistics, Jul. 2022, pp. 90–101. DOI: 10.18653/v1/2022.deeplo-1.10. [Online]. Available: https://aclanthology.org/2022.deeplo-1.10.

[11] Y. Wang, Y. Wang, J. Liu, and Z. Liu, *A comprehensive survey of grammar error correction*, 2020. eprint: 2005.06600 (cs.CL). [Online]. Available: https://arxiv.org/abs/2005.06600.

[12] M. Kaneko, M. Mita, S. Kiyono, J. Suzuki, and K. Inui, "Encoder-decoder models can benefit from pre-trained masked language models in grammatical error correction," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, D. Jurafsky, J. Chai, N. Schluter, and J. Tetreault, Eds., Online: Association for Computational Linguistics, Jul. 2020, pp. 4248–4254. DOI: 10.18653/v1/2020.acl-main.391. [Online]. Available: https://aclanthology.org/2020.acl-main.391.

[13] C. E. Shannon, "A mathematical theory of communication," *The Bell System Technical Journal*, vol. 27, no. 3, pp. 379–423, 1948. DOI: 10.1002/j.1538-7305.1948.tb01338.x.

[14] M. R. Costa-jussà, "An overview of the phrase-based statistical machine translation techniques," *The Knowledge Engineering Review*, vol. 27, no. 4, pp. 413–431, 2012. DOI: 10.1017/S026988891200029X.

[15] Z. Yuan, "Grammatical error correction in non-native English," University of Cambridge, Computer Laboratory, Tech. Rep. UCAM-CL-TR-904, Mar. 2017. DOI: 10.48456/tr-904. [Online]. Available: https://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-904.pdf.

[16] D. Bahdanau, K. Cho, and Y. Bengio, *Neural machine translation by jointly learning to align and translate*, 2016. eprint: 1409.0473 (cs.CL). [Online]. Available: https://arxiv.org/abs/1409.0473.

[17] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997. DOI: 10.1162/neco.1997.9.8.1735.

[18] A. Vaswani *et al.*, "Attention is all you need," *CoRR*, vol. abs/1706.03762, 2017. eprint: 1706.03762. [Online]. Available: http://arxiv.org/abs/1706.03762.

[19] C. Olah, *Understanding LSTM networks*, https://colah.github.io/posts/2015-08-Understanding-LSTMs/, 2015.

[20] M. E. Peters *et al.*, "Deep contextualized word representations," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, New Orleans, Louisiana: Association for Computational Linguistics, Jun. 2018. [Online]. Available: https://aclanthology.org/N18-1202.

[21] Y. Bengio, R. Ducharme, and P. Vincent, "A neural probabilistic language model," in *Advances in Neural Information Processing Systems*, T. Leen, T. Dietterich, and V. Tresp, Eds., vol. 13, MIT Press, 2000. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2000/file/728f206c2a01bf572b5940d7d9a8fa4c-Paper.pdf.

[22] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, J. Burstein, C. Doran, and T. Solorio, Eds., Minneapolis, Minnesota: Association for Computational Linguistics, Jun. 2019, pp. 4171–4186. DOI: 10.18653/v1/N19-1423. [Online]. Available: https://aclanthology.org/N19-1423.

[23] B. Muller, A. Anastasopoulos, B. Sagot, and D. Seddah, "When being unseen from mBERT is just the beginning: Handling new languages with multilingual language models," in *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Online: Association for Computational Linguistics, Jun. 2021. [Online]. Available: https://aclanthology.org/2021.naacl-main.38.

[24] A. Busuttil, *Grammatical analysis of Maltese text*, 2023. [Online]. Available: https://www.um.edu.mt/library/oar/handle/123456789/115279.

[25] R. Dale and A. Kilgarriff, "Helping our own: The HOO 2011 pilot shared task," in *Proceedings of the 13th European Workshop on Natural Language Generation*, C. Gardent and K. Striegnitz, Eds., Nancy, France: Association for Computational Linguistics, Sep. 2011, pp. 242–249. [Online]. Available: https://aclanthology.org/W11-2838.

[26] R. Dale, I. Anisimoff, and G. Narroway, "HOO 2012: A report on the preposition and determiner error correction shared task," in *Proceedings of the Seventh Workshop on Building Educational Applications Using NLP*, J. Tetreault, J. Burstein, and C. Leacock, Eds., Montréal, Canada: Association for Computational Linguistics, Jun. 2012, pp. 54–62. [Online]. Available: https://aclanthology.org/W12-2006.

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

[28] H. T. Ng, S. M. Wu, T. Briscoe, C. Hadiwinoto, R. H. Susanto, and C. Bryant, "The CoNLL-2014 shared task on grammatical error correction," in *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task*, H. T. Ng, S. M. Wu, T. Briscoe, C. Hadiwinoto, R. H. Susanto, and C. Bryant, Eds., Baltimore, Maryland: Association for Computational Linguistics, Jun. 2014, pp. 1–14. DOI: 10.3115/v1/W14-1701. [Online]. Available: https://aclanthology.org/W14-1701.

[29] C. Bryant, M. Felice, Ø. E. Andersen, and T. Briscoe, "The BEA-2019 shared task on grammatical error correction," in *Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications*, H. Yannakoudakis, E. Kochmar, C. Leacock, N. Madnani, I. Pilán, and T. Zesch, Eds., Florence, Italy: Association for Computational Linguistics, Aug. 2019, pp. 52–75. DOI: 10.18653/v1/W19-4406. [Online]. Available: https://aclanthology.org/W19-4406.

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

[31] C. Napoles, K. Sakaguchi, and J. Tetreault, "JFLEG: A fluency corpus and benchmark for grammatical error correction," in *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, M. Lapata, P. Blunsom, and A. Koller, Eds., Valencia, Spain: Association for Computational Linguistics, Apr. 2017, pp. 229–234. [Online]. Available: https://aclanthology.org/E17-2037.

[32] A. Koyama, T. Kiyuna, K. Kobayashi, M. Arai, and M. Komachi, "Construction of an evaluation corpus for grammatical error correction for learners of Japanese as a second language," English, in *Proceedings of the Twelfth Language Resources and Evaluation Conference*, N. Calzolari *et al.*, Eds., Marseille, France: European Language Resources Association, May 2020, pp. 204–211, ISBN: 979-10-95546-34-4. [Online]. Available: https://aclanthology.org/2020.lrec-1.26.

[33] A. Rozovskaya, K.-W. Chang, M. Sammons, and D. Roth, "The University of Illinois system in the CoNLL-2013 shared task," in *Proceedings of the Seventeenth Conference on Computational Natural Language Learning: Shared Task*, H. T. Ng, J. Tetreault, S. M. Wu, Y. Wu, and C. Hadiwinoto, Eds., Sofia, Bulgaria: Association for Computational Linguistics, Aug. 2013, pp. 13–19. [Online]. Available: https://aclanthology.org/W13-3602.

[34] A. Rozovskaya, K.-W. Chang, M. Sammons, D. Roth, and N. Habash, "The Illinois-Columbia system in the CoNLL-2014 shared task," in *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task*, H. T. Ng, S. M. Wu, T. Briscoe, C. Hadiwinoto, R. H. Susanto, and C. Bryant, Eds., Baltimore, Maryland: Association for Computational Linguistics, Jun. 2014, pp. 34–42. DOI: 10.3115/v1/W14-1704. [Online]. Available: https://aclanthology.org/W14-1704.

[35] Z. Yuan, F. Stahlberg, M. Rei, B. Byrne, and H. Yannakoudakis, "Neural and FST-based approaches to grammatical error correction," in *Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications*, H. Yannakoudakis, E. Kochmar, C. Leacock, N. Madnani, I. Pilán, and T. Zesch, Eds., Florence, Italy: Association for Computational Linguistics, Aug. 2019, pp. 228–239. DOI: 10.18653/v1/W19-4424. [Online]. Available: https://aclanthology.org/W19-4424.

[36] R. Li *et al.*, "The LAIX systems in the BEA-2019 GEC shared task," in *Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications*, H. Yannakoudakis, E. Kochmar, C. Leacock, N. Madnani, I. Pilán, and T. Zesch, Eds., Florence, Italy: Association for Computational Lin-

guistics, Aug. 2019, pp. 159–167. DOI: 10.18653/v1/W19-4416. [Online]. Available: https://aclanthology.org/W19-4416.

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

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

[39] C. Napoles, K. Sakaguchi, and J. Tetreault, "There's no comparison: Reference-less evaluation metrics in grammatical error correction," in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, J. Su, K. Duh, and X. Carreras, Eds., Austin, Texas: Association for Computational Linguistics, Nov. 2016, pp. 2109–2115. DOI: 10.18653/v1/D16-1228. [Online]. Available: https://aclanthology.org/D16-1228.

[40] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "Bleu: A method for automatic evaluation of machine translation," in *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ser. ACL '02, Philadelphia, Pennsylvania: Association for Computational Linguistics, 2002, pp. 311–318. DOI: 10.3115/1073083.1073135. [Online]. Available: https://doi.org/10.3115/1073083.1073135.

[41] M. Felice and Z. Yuan, "To err is human, to correct is divine," *XRDS*, vol. 21, no. 1, pp. 22–27, Oct. 2014, ISSN: 1528-4972. DOI: 10.1145/2659833. [Online]. Available: https://doi.org/10.1145/2659833.

[42] J. Lee, "Automatic article restoration," in *Proceedings of the Student Research Workshop at HLT-NAACL 2004*, Boston, Massachusetts, USA: Association for Computational Linguistics, May 2004, pp. 31–36. [Online]. Available: https://aclanthology.org/N04-2006.

[43] M. Chodorow, J. Tetreault, and N.-R. Han, "Detection of grammatical errors involving prepositions," in *Proceedings of the Fourth ACL-SIGSEM Workshop on Prepositions*, F. Costello, J. Kelleher, and M. Volk, Eds., Prague, Czech Repub-

lic: Association for Computational Linguistics, Jun. 2007, pp. 25–30. [Online]. Available: https://aclanthology.org/W07-1604.

[44] A. Kunchukuttan, S. Chaudhury, and P. Bhattacharyya, "Tuning a grammar correction system for increased precision," in *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task*, H. T. Ng, S. M. Wu, T. Briscoe, C. Hadiwinoto, R. H. Susanto, and C. Bryant, Eds., Baltimore, Maryland: Association for Computational Linguistics, Jun. 2014, pp. 60–64. DOI: 10.3115/v1/W14-1708. [Online]. Available: https://aclanthology.org/W14-1708.

[45] J. Lee and S. Seneff, "Correcting misuse of verb forms," in *Proceedings of ACL-08: HLT*, J. D. Moore, S. Teufel, J. Allan, and S. Furui, Eds., Columbus, Ohio: Association for Computational Linguistics, Jun. 2008, pp. 174–182. [Online]. Available: https://aclanthology.org/P08-1021.

[46] C. Wang, R. Li, and H.-C. Lin, "Deep context model for grammatical error correction," in *Slate*, 2017. [Online]. Available: https://api.semanticscholar.org/CorpusID:19114279.

[47] C. Brockett, W. B. Dolan, and M. Gamon, "Correcting ESL errors using phrasal SMT techniques," in *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, N. Calzolari, C. Cardie, and P. Isabelle, Eds., Sydney, Australia: Association for Computational Linguistics, Jul. 2006, pp. 249–256. DOI: 10.3115/1220175.1220207. [Online]. Available: https://aclanthology.org/P06-1032.

[48] M. Felice, Z. Yuan, Ø. E. Andersen, H. Yannakoudakis, and E. Kochmar, "Grammatical error correction using hybrid systems and type filtering," in *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task*, H. T. Ng, S. M. Wu, T. Briscoe, C. Hadiwinoto, R. H. Susanto, and C. Bryant, Eds., Baltimore, Maryland: Association for Computational Linguistics, Jun. 2014, pp. 15–24. DOI: 10.3115/v1/W14-1702. [Online]. Available: https://aclanthology.org/W14-1702.

[49] M. Junczys-Dowmunt and R. Grundkiewicz, "Phrase-based machine translation is state-of-the-art for automatic grammatical error correction," in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, J. Su, K. Duh, and X. Carreras, Eds., Austin, Texas: Association for Computational Linguistics, Nov. 2016, pp. 1546–1556. DOI: 10.18653/v1/D16-1161. [Online]. Available: https://aclanthology.org/D16-1161.

[50] Z. Yuan and T. Briscoe, "Grammatical error correction using neural machine translation," in *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, K. Knight, A. Nenkova, and O. Rambow, Eds., San Diego, California: Association for Computational Linguistics, Jun. 2016, pp. 380–386. DOI: 10.18653/v1/N16-1042. [Online]. Available: https://aclanthology.org/N16-1042.

[51] Z. Xie, A. Avati, N. Arivazhagan, D. Jurafsky, and A. Y. Ng, "Neural language correction with character-based attention," vol. abs/1603.09727, 2016. eprint: 1603.09727. [Online]. Available: http://arxiv.org/abs/1603.09727.

[52] J. Ji, Q. Wang, K. Toutanova, Y. Gong, S. Truong, and J. Gao, "A nested attention neural hybrid model for grammatical error correction," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, R. Barzilay and M.-Y. Kan, Eds., Vancouver, Canada: Association for Computational Linguistics, Jul. 2017, pp. 753–762. DOI: 10.18653/v1/P17-1070. [Online]. Available: https://aclanthology.org/P17-1070.

[53] W. Zhao, L. Wang, K. Shen, R. Jia, and J. 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)*, J. Burstein, C. Doran, and T. Solorio, Eds., Minneapolis, Minnesota: Association for Computational Linguistics, Jun. 2019, pp. 156–165. DOI: 10.18653/v1/N19-1014. [Online]. Available: https://aclanthology.org/N19-1014.

[54] Y. Zhang, B. Zhang, Z. Li, Z. Bao, C. Li, and M. Zhang, "SynGEC: Syntax-enhanced grammatical error correction with a tailored GEC-oriented parser," in *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, Y. Goldberg, Z. Kozareva, and Y. Zhang, Eds., Abu Dhabi, United Arab Emirates: Association for Computational Linguistics, Dec. 2022, pp. 2518–2531. DOI: 10.18653/v1/2022.emnlp-main.162. [Online]. Available: https://aclanthology.org/2022.emnlp-main.162.

[55] Y. J. Choe, J. Ham, K. Park, and Y. Yoon, "A neural grammatical error correction system built on better pre-training and sequential transfer learning," in *Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications*, H. Yannakoudakis, E. Kochmar, C. Leacock, N. Madnani, I. Pilán, and T. Zesch, Eds., Florence, Italy: Association for Compu-

tational Linguistics, Aug. 2019, pp. 213–227. DOI: 10.18653/v1/W19-4423. [Online]. Available: https://aclanthology.org/W19-4423.

[56] S. Katsumata and M. Komachi, "Stronger baselines for grammatical error correction using a pretrained encoder-decoder model," in *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing*, K.-F. Wong, K. Knight, and H. Wu, Eds., Suzhou, China: Association for Computational Linguistics, Dec. 2020, pp. 827–832. [Online]. Available: https://aclanthology.org/2020.aacl-main.83.

[57] A. Luhtaru, E. Korotkova, and M. Fishel, "No error left behind: Multilingual grammatical error correction with pre-trained translation models," in *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, Y. Graham and M. Purver, Eds., St. Julian's, Malta: Association for Computational Linguistics, Mar. 2024, pp. 1209–1222. [Online]. Available: https://aclanthology.org/2024.eacl-long.73.

[58] S. Chollampatt, W. Wang, and H. T. Ng, "Cross-sentence grammatical error correction," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, A. Korhonen, D. Traum, and L. Màrquez, Eds., Florence, Italy: Association for Computational Linguistics, Jul. 2019, pp. 435–445. DOI: 10.18653/v1/P19-1042. [Online]. Available: https://aclanthology.org/P19-1042.

[59] J. Guo, Z. Zhang, L. Xu, H.-R. Wei, B. Chen, and E. Chen, *Incorporating bert into parallel sequence decoding with adapters*, 2020. eprint: 2010.06138 (cs.CL). [Online]. Available: https://arxiv.org/abs/2010.06138.

[60] R. Vázquez, H. Celikkanat, M. Creutz, and J. Tiedemann, "On the differences between BERT and MT encoder spaces and how to address them in translation tasks," in *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing: Student Research Workshop*, J. Kabbara, H. Lin, A. Paullada, and J. Vamvas, Eds., Online: Association for Computational Linguistics, Aug. 2021, pp. 337–347. DOI: 10.18653/v1/2021.acl-srw.35. [Online]. Available: https://aclanthology.org/2021.acl-srw.35.

[61] S. Cao, N. Kitaev, and D. Klein, "Multilingual alignment of contextual word representations," *CoRR*, vol. abs/2002.03518, 2020. eprint: 2002.03518. [Online]. Available: https://arxiv.org/abs/2002.03518.

[62] S. Chollampatt and H. T. Ng, "Connecting the dots: Towards human-level grammatical error correction," in *Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications*, J. Tetreault, J. Burstein, C. Leacock, and H. Yannakoudakis, Eds., Copenhagen, Denmark: Association for Computational Linguistics, Sep. 2017, pp. 327–333. DOI: 10.18653/v1/W17-5037. [Online]. Available: https://aclanthology.org/W17-5037.

[63] R. Grundkiewicz and M. Junczys-Dowmunt, "Near human-level performance in grammatical error correction with hybrid machine translation," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, M. Walker, H. Ji, and A. Stent, Eds., New Orleans, Louisiana: Association for Computational Linguistics, Jun. 2018, pp. 284–290. DOI: 10.18653/v1/N18-2046. [Online]. Available: https://aclanthology.org/N18-2046.

[64] Z. Yuan, S. Taslimipoor, C. Davis, and C. Bryant, "Multi-class grammatical error detection for correction: A tale of two systems," in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, M.-F. Moens, X. Huang, L. Specia, and S. W.-t. Yih, Eds., Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, Nov. 2021, pp. 8722–8736. DOI: 10.18653/v1/2021.emnlp-main.687. [Online]. Available: https://aclanthology.org/2021.emnlp-main.687.

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

[66] Z. Yuan, T. Briscoe, and M. Felice, "Candidate re-ranking for SMT-based grammatical error correction," in *Proceedings of the 11th Workshop on Innovative Use of NLP for Building Educational Applications*, J. Tetreault, J. Burstein, C. Leacock, and H. Yannakoudakis, Eds., San Diego, CA: Association for Computational Linguistics, Jun. 2016, pp. 256–266. DOI: 10.18653/v1/W16-0530. [Online]. Available: https://aclanthology.org/W16-0530.

[67] D. T. Hoang, S. Chollampatt, and H. T. Ng, "Exploiting n-best hypotheses to improve an SMT approach to grammatical error correction," *CoRR*, vol. abs/1606.00210, 2016. eprint: 1606.00210. [Online]. Available: http://arxiv.org/abs/1606.00210.

[68] R. Grundkiewicz, M. Junczys-Dowmunt, and K. 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*, H. Yannakoudakis, E. Kochmar, C. Leacock, N. Madnani, I. Pilán, and T. Zesch, Eds., Florence, Italy: Association for Computational Linguistics, Aug. 2019, pp. 252–263. DOI: 10.18653/v1/W19-4427. [Online]. Available: https://aclanthology.org/W19-4427.

[69] D. Opitz and R. Maclin, "Popular ensemble methods: An empirical study," *Journal of Artificial Intelligence Research*, vol. 11, pp. 169–198, Aug. 1999, ISSN: 1076-9757. DOI: 10.1613/jair.614. [Online]. Available: http://dx.doi.org/10.1613/jair.614.

[70] Z. Yuan and C. Bryant, "Document-level grammatical error correction," in *Proceedings of the 16th Workshop on Innovative Use of NLP for Building Educational Applications*, J. Burstein *et al.*, Eds., Online: Association for Computational Linguistics, Apr. 2021, pp. 75–84. [Online]. Available: https://aclanthology.org/2021.bea-1.8.

[71] F. Palma Gomez, A. Rozovskaya, and D. Roth, "A low-resource approach to the grammatical error correction of Ukrainian," in *Proceedings of the Second Ukrainian Natural Language Processing Workshop (UNLP)*, M. Romanyshyn, Ed., Dubrovnik, Croatia: Association for Computational Linguistics, May 2023, pp. 114–120. DOI: 10.18653/v1/2023.unlp-1.14. [Online]. Available: https://aclanthology.org/2023.unlp-1.14.

[72] Z. Xie, G. Genthial, S. Xie, A. Ng, and D. 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)*, M. Walker, H. Ji, and A. Stent, Eds., New Orleans, Louisiana: Association for Computational Linguistics, Jun. 2018, pp. 619–628. DOI: 10.18653/v1/N18-1057. [Online]. Available: https://aclanthology.org/N18-1057.

[73] S. Kiyono, J. Suzuki, M. Mita, T. Mizumoto, and K. 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)*, K. Inui, J. Jiang, V. Ng, and X. Wan, Eds., Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 1236–1242. DOI: 10.18653/v1/D19-1119. [Online]. Available: https://aclanthology.org/D19-1119.

[74] A. Gatt and S. Čéplö, "Digital corpora and other electronic resources for maltese," in *Corpus linguistics*, UCREL Lancaster, 2013, pp. 96–97.

[75] O. Vella, *Spellchecking exercises for diploma students*, Private communication, 2015.

[76] O. Vella, *Spellchecking exercises for diploma students*, Private communication, 2022.

[77] M. Junczys-Dowmunt, R. Grundkiewicz, S. Guha, and K. Heafield, "Approaching neural grammatical error correction as a low-resource machine translation task," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, M. Walker, H. Ji, and A. Stent, Eds., New Orleans, Louisiana: Association for Computational Linguistics, Jun. 2018, pp. 595–606. DOI: 10.18653/v1/N18-1055. [Online]. Available: https://aclanthology.org/N18-1055.

[78] A. Golding, "A Bayesian hybrid method for context-sensitive spelling correction," in *Third Workshop on Very Large Corpora*, 1995. [Online]. Available: https://aclanthology.org/W95-0104.

[79] M. Ghazvininejad, O. Levy, Y. Liu, and L. Zettlemoyer, "Constant-time machine translation with conditional masked language models," *CoRR*, vol. abs/1904.09324, 2019. eprint: 1904.09324. [Online]. Available: http://arxiv.org/abs/1904.09324.

[80] G. Elliott and N. Johnson, "All the right letters - just not necessarily in the right order. Spelling errors in a sample of GCSE English scripts," 2009. DOI: 10.17863/CAM.100481. [Online]. Available: https://www.repository.cam.ac.uk/handle/1810/354703.