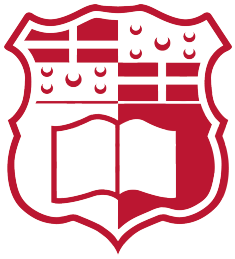


Advancing Automated Maltese Spell Checking Using Deep Learning

Owen Fava

Supervised by Prof. Alexiei Dingli

Department of Artificial Intelligence

Faculty of ICT

University of Malta

February, 2026

A dissertation submitted in partial fulfilment of the requirements for the degree of M.Sc. in Artificial Intelligence.

L-Università ta' Malta

WWW.UM.EDU.MT

First edition, Tuesday 24th February, 2026

Dedication

For my family and my girlfriend.
Your patience, love and support meant a lot to me throughout this work.

Acknowledgements

I would like to express my sincere gratitude to my supervisor, Prof. Alexiei Dingli, for his guidance, support and help throughout this work.

I am also grateful to Prof. Michael Spagnol, Mr. Thomas Pace, and Ms. Amy Borġ for their professional help, for serving as points of reference for linguistic accuracy, and for providing the document *L-Ilsien Malti Għal Qalbi*, which allowed me to extract words and sentences.

My thanks also go to Mr. Reuben Degiorgio for authorising the scraping of the *verb.mt* website to collect additional linguistic data, and to Mr. Christopher Giordano for providing the *Sagħtar* documents, which were highly valuable for data extraction. I further thank Mr. Robert Camenzuli, Mr. Mark Axiaq and Mr. Joseph Izzo Clarke for sending publicly available, proofread documents related to *Kawżi mill-Qorti tal-Ġustizzja*, which helped me expand and strengthen my dataset.

I would also like to thank Mr. Benjamin Joseph Spiteri, Mr. Timothy Paul Mercieca, and Mr. Jeremy Farruġia for their support in accessing and using the University of Malta server.

I am also deeply thankful to my family and my girlfriend for their constant support throughout this work.

Abstract

Research in Natural Language Processing (NLP) for Maltese has advanced in recent years, yet reliable and contextually aware spell checking support remains limited. Existing tools, such as spelling.mt, provide dictionary-based correction but cannot handle grammatical or context-sensitive errors, leaving a clear gap compared with high-resource languages. This dissertation investigates whether a Large Language Model (LLM) can be fine-tuned to develop an effective and resource-efficient spell checker for Maltese.

The study adapts Meta's Large Language Model Meta AI (LLaMA)-3-8B-Instruct model using pairs of incorrect and corrected Maltese words and sentences. A custom fine-tuning corpus was created by introducing linguistically motivated synthetic errors into correctly written text and incorporating smaller sets of authentic incorrect-correct pairs from prior research. Two complementary models were trained: a word-level model that corrects individual tokens and a sentence-level model that corrects grammatical and orthographic errors within full sentences. Their performance was evaluated across synthetic, real-world, and correct-input datasets.

The main contribution of this dissertation is the development of the first LLM-based spell checking models, specifically fine-tuned for Maltese, supported by a curated error-correction dataset and a systematic multi-setting evaluation. To our knowledge, this study is the first to explore the use of a modern generative LLM to build a Maltese spell checker and assess its potential in a low-resource setting.

The developed models demonstrated strong performance on incorrect-correct pairs with synthetically generated errors, with the word-level model achieving 84.64% accuracy and the sentence-level model reaching 95.3%. For real-world evaluation, only the sentence-level model was tested because no suitable word-level datasets are available. In this context, the accuracy of the sentence-level model dropped to 29.6%, reflecting the increased variability of naturally occurring mistakes. However, metrics such as Error Annotation Toolkit (ERRANT), Bilingual Evaluation Understudy (BLEU), and Grammar Language Evaluation Understudy (GLEU) returned promising values, indicating that the model often produced corrections close to the target.

Overall, the findings demonstrate that fine-tuning an LLM offers a promising pathway towards effective Maltese spell checking, while revealing key challenges related to generalisation and data scarcity. The study provides an initial benchmark and outlines concrete directions for developing more robust, deployable spell checking solutions for low-resource languages.

Contents

List of Figures

List of Tables

List of Abbreviations

1

Introduction

Natural Language Processing (NLP) is a branch of Artificial Intelligence (AI) that enables computers to understand and process human language. It employs Machine Learning (ML) algorithms to analyse written and spoken data, allowing systems to interpret linguistic patterns and meanings (Tolegenova, 2024). However, progress in NLP has not been evenly distributed across languages, with English dominating both research attention and data resources (Joshi et al., 2020). This dominance stems from English's global use and significant online presence, which has resulted in the largest collections of digitised texts, including books, patents, and digital resources, attracting substantial attention from the research community. As a result, English-based models benefit from abundant, high-quality data for training and evaluation, accelerating the development of high-performing NLP tools, a benefit that low-resource languages do not share (Nicholas & Bhatia, 2023).

Low-resource languages such as Maltese have fewer digitised corpora, limited representation in large-scale datasets, and minimal integration into major NLP frameworks. These constraints make the development of many NLP applications considerably more challenging, including automatic spelling correction (Magueresse et al., 2020), which this dissertation investigates in the context of Maltese.

Over the years, various techniques have been used to develop spelling correction systems. These approaches range from simple dictionary look-up and rule-based methods to probabilistic algorithms and advanced neural architectures. The progression of these systems mirrors the broader shift within NLP from handcrafted rules to data-driven models that capture linguistic context and variation (Bijoy et al., 2025). Although spell checkers exist for many languages, their design must account for language-specific characteristics, resulting in substantial variation in performance depending on linguistic complexity and the availability of suitable data (S. Singh & Singh, 2020).

In the context of Maltese, the limited availability of data creates significant challenges for developing an effective spell checker. Modern methods typically depend on extensive, varied training datasets, which means they must be adapted to function ef-

fectively in environments with limited data (Blasi et al., 2022; Magueresse et al., 2020).

The following sections will provide a detailed discussion of the problem area addressed in this study and the motivation behind it.

1.1 | The Problem Area

Developing a spell checker that can correct both spelling and grammatical errors involves several technical and linguistic challenges. A system must be able to recognise when a word is misspelled, suggest a likely correction, and assess whether a sentence remains grammatically valid after the change. This process requires more than just dictionary lookup because it depends on context-aware models capable of interpreting relationships between words, such as agreement in number, gender, or tense. Designing such systems typically relies on access to large, annotated datasets that include real examples of both typographical and grammatical errors. Without these resources, it is difficult to build models that generalise well to real-world usage.

These challenges are further amplified when working with low-resource languages such as Maltese. Due to its limited digital footprint, Maltese lacks large-scale corpora and annotated datasets, especially those containing examples of naturally occurring errors. This scarcity limits the effectiveness of data-driven approaches that have proven successful for high-resource languages like English. In many cases, researchers rely on synthetic error generation to expand training data (Bryant et al., 2023).

The structure of the Maltese language adds further complexity. Maltese is a Semitic-derived language influenced by Romance and English. It shares key characteristics with other Semitic languages, making use of root-and-template morphology, in which different forms of the same word are created by inserting vowels between a fixed sequence of consonants. Maltese is distinguished by free word order, mixed morphology, an aspect-based temporal system, and the absence of a morphological infinitive (Fabri, 2010; Rosner & Borg, 2023). Additionally, Maltese combines non-linear morphological patterns with affixation typical of Romance languages. This interaction between systems results in rich but often irregular word forms (Azzopardi-Alexander & Borg, 2013). Even minor internal changes can alter a word's grammatical role or meaning, and grammatical agreement such as number, gender, or tense is frequently expressed within the word itself and across sentence boundaries (Fabri, 2010). Consequently, effective grammatical error correction in Maltese requires models that can capture sentence-level context rather than relying solely on word-level processing.

Together, these factors define the problem area addressed in this study. Develop-

ing an effective Maltese spell checker requires overcoming both linguistic and resource-based challenges. It also calls for methods that can model grammatical relationships and sentence-level structure in contexts where annotated training data is scarce. Addressing these issues is essential for advancing Maltese language technology and contributing to broader efforts in equitable low-resource NLP.

1.2 | Motivation

Research in NLP for Maltese has progressed steadily in recent years and has contributed significantly to strengthen the language's presence in digital contexts. Important developments include a Part-of-Speech (POS) and dependency parser, and the construction of a large corpus of approximately 470 million tokens spanning multiple genres (Korpus Malti[1]). Alongside these contributions, two Maltese language models have been introduced: a monolingual Bidirectional Encoder Representations from Transformers (BERT)-based model and a multilingual Maltese model trained using multilingual BERT as its initial checkpoint.

Spell checking has likewise been the focus of several research efforts. A range of approaches including rule-based systems, statistical methods, and deep learning models have been explored. Yet despite this work, no publicly accessible tool has achieved the level of accuracy, robustness, and contextual awareness required for reliable use in real-world settings. The only publicly available system remains spelling.mt[2], introduced by Casha (2004). This tool uses a generated word list and an algorithm similar to Unix Aspell. While useful as a dictionary-based spell checker, it cannot identify grammatical errors or handle context-sensitive corrections. More recent research has applied Transformer-based architectures to the task, including work that made use of the Maltese BERT model, though its effectiveness remains limited.

The landscape of NLP has changed considerably with the introduction of Large Language Models (LLMs). These models have achieved state-of-the-art performance across diverse applications, including those involving low-resource languages. Their success suggests that LLMs may offer a promising new direction for Maltese spell checking.

The need for a reliable Maltese spell checker is growing. Maltese is used widely across education, government, media, and everyday digital communication. On social media in particular, writing in Maltese frequently exhibits inconsistent or incorrect word forms, which may be influenced by the absence of accessible correction tools. In con-

[1] https://mlrs.research.um.edu.mt/index.php?page=corpora
[2] https://spelling.mt/

trast, languages such as English benefit from integrated spell checkers and grammatical error correction systems that help users produce clear and accurate text. The lack of equivalent support for Maltese affects writing quality and may contribute to the spread of incorrect forms in digital spaces.

A robust spell checker for Maltese would offer several advantages. It would support everyday writing, assist language learners, and improve academic and professional communication. It would also help preserve high-quality Maltese in the digital era and contribute to the improvement of language resources used in downstream NLP applications.

In light of these needs and opportunities, this dissertation aims to investigate whether LLMs can be leveraged to develop an effective and contextually aware spell checker for Maltese. The study explores the use of LLM-based approaches for detecting and correcting errors, evaluates their performance, and compares them with existing methods.

1.3 | Aims and Objectives

The aim of this study is to create a reliable Maltese spell checker by finetuning an open-source LLM, specifically Meta's Large Language Model Meta AI (LLaMA) model, using a custom Maltese dataset.

To offer a clear overview of how the research aim is pursued, Table 1.1 provides a consolidated mapping that connects the aim to its objectives, the related research questions, the methodological approach, and the chapter in which each element is examined.

The following Sections 1.3.1 - 1.3.4 then discuss each objective in detail, outlining the steps taken to achieve them.

1.3.1 | First Objective

The first objective of this study is:

> *Data Collection and Pre-processing - Gather and prepare a comprehensive dataset of Maltese text, including common spelling mistakes and their corrections.*

This objective is designed to address the following research question:

> *What techniques can be employed to design a resource-efficient spell checker for a low-resourced language, considering limited linguistic resources?*

Table 1.1: Overview of aim, objectives, research questions, methods, and chapters. Source: Author.

Aim: This study aims to create a reliable Maltese spell checker by fine-tuning an open-source LLM, specifically Meta's LLaMA model, using a custom Maltese dataset.			
Objective	**Research Question**	**Method**	**Chapter**
Data Collection and Pre-processing - Gather and prepare a comprehensive dataset of Maltese text, including common spelling mistakes and their corrections.	What techniques can be employed to design a resource efficient spell checker for a low-resourced language, considering limited linguistic resources?	Methodology	4
Fine-tuning of the LLM - Perform fine-tuning on the selected LLM using the custom Maltese dataset.	What techniques can be employed to design a resource efficient spell checker for a low-resourced language, considering limited linguistic resources?	Methodology	4
Evaluation - Evaluate the fine-tuned model using proper evaluation metrics.	How well does the spell checker capture and correct spelling mistakes?	Evaluation and Results	5
Comparison with existing spell checkers - Evaluating the fine-tuned model against existing spell checkers to assess whether it outperforms or falls short of their performance.	Does the proposed methodology improve on results of previous research on Maltese spell checkers? Is the implemented tool more robust, when compared to previous implementations?	Discussion	6

This objective is pursued by creating a reliable, proof-read corpus that, to the best of our knowledge, contains no spelling errors. This error-free corpus serves as the target, or gold-standard, Maltese text for the models. Synthetic errors are then introduced into these correct texts to produce both word-level and sentence-level datasets, with incorrect forms generated based on common spelling patterns identified in studies examining frequent mistakes in Maltese writing. In addition, real-world spelling errors collected from previous research were incorporated into both the training and testing stages of the study.

1.3.2 | Second Objective

The second objective of this study is stated as follows:

> *Fine-tuning of the LLM - Perform fine-tuning on the selected LLM using the custom Maltese dataset.*

This objective is designed to address the same research question as the first objective, namely:

> *What techniques can be employed to design a resource-efficient spell checker for a low-resourced language, considering limited linguistic resources?*

To fulfil this objective, the study fine-tunes the selected LLM using the constructed Maltese dataset. This process configures suitable hyperparameters and trains the model at both the word and sentence levels, enabling it to learn effective correction patterns under data-constrained conditions.

Specifically, the word-level model is designed to correct typos and other spelling errors, transforming non-word errors into the appropriate corrected forms. The sentence-level model, in addition to handling non-word errors in sentences, is intended to provide appropriate grammar corrections. For example, if an input sentence contains errors such as incorrect tense, gender, or number agreement, the model should be capable of correcting these mistakes to produce contextually and grammatically correct output.

1.3.3 | Third Objective

The third objective of this study is stated as follows:

> *Evaluation - Evaluate the fine-tuned model using proper evaluation metrics.*

This objective is designed to address the following research question:

> *How well does the spell checker capture and correct spelling mistakes?*

This objective is met by evaluating the system on Maltese text corpora using appropriate metrics tailored to the spell checking task. The word-level models are assessed on their ability to correct typographical and orthographic errors, while the sentence-level models are evaluated on both spelling mistakes and broader grammatical mistakes.

1.3.4 | Fourth Objective

The fourth objective of this study is stated as follows:

> *Comparison with existing spell checkers - Evaluate the fine-tuned model against existing spell checkers to assess whether it outperforms or falls short of their performance.*

This objective is designed to address the following research questions:

> *Does the proposed methodology improve on results of previous research on Maltese spell checkers?*
>
> *Is the implemented tool more robust when compared to previous implementations?*

To fulfil this objective, the results obtained in this study were compared with those reported in earlier research on Maltese spell checkers. This comparison assesses whether the proposed methodology offers performance improvements and whether the developed tool demonstrates increased robustness relative to existing implementations.

1.4 | Approach

This study investigates how an LLM can be adapted and fine-tuned to develop a resource-efficient Maltese spell checker. The approach centres on training Meta's LLaMA-3-8B-Instruct model on pairs of incorrect and corrected Maltese words and sentences, enabling it to learn how to transform spelling mistakes into its correct form. In this framework, the model performs a single-step correction process, generating the corrected output directly from the input sequence.

A custom fine-tuning dataset was created by collecting correctly written Maltese text from a range of sources and introducing synthetic spelling and grammatical errors into the collected corpora. The synthetic errors were designed on the basis of linguistic literature examining frequent orthographic and grammatical mistakes in Maltese writing, ensuring that the generated examples reflected commonly observed error patterns. Real-world errors and their corresponding corrections, taken from previous studies, were also incorporated. These real-world datasets were used both during training and during testing to ensure that the model was evaluated on authentic Maltese error patterns as well as systematically generated ones.

Following fine-tuning, the model is evaluated using different evaluation frameworks to assess how accurately it corrects spelling errors and whether its performance shows

improvements relative to existing Maltese spell checkers. This enables the study to assess the model's performance as a spell checker and to evaluate whether it is appropriate for a low-resourced language such as Maltese.

1.5 | Document Structure

The dissertation is organised into seven chapters, each addressing a different stage of the research process. The *Introduction* (Chapter 1) outlines the research problem, motivation, aim, objectives and research questions, and provides an overview of the approach adopted in this study. The *Background* (Chapter 2) reviews the landscape of spell checking techniques, from traditional rule-based and statistical methods to modern deep learning and LLM-based approaches, establishing the conceptual foundations for the study. The *Literature Review* (Chapter 3) synthesises key findings from prior work, with a focus on error correction, low-resource language processing and the development of spelling and grammatical correction systems. The *Methodology* (Chapter 4) details the collection and creation of the dataset, the generation of synthetic errors, the model configurations and the experimental procedures used during fine-tuning, and identifies the relevant methodological limitations. *Results and Evaluation* (Chapter 5) presents the experimental workflow and reports the performance of the fine-tuned models across the different evaluation settings. The *Discussion* (Chapter 6) offers a critical interpretation of these findings, examining the strengths and weaknesses of the approach and explaining the models' behaviour. Finally, the *Conclusion* (Chapter 7) summarises the main contributions of the dissertation, reflects on the extent to which the research aim and objectives have been achieved, and proposes directions for future work.

2

Background

This chapter provides a structured overview of the foundational concepts and techniques for creating a spell checker for Maltese. It begins with an introduction to the Maltese language, describing its distinctive linguistic features, including its Semitic origins, the influence of other languages, and its complex morphological patterns. Building on this linguistic background, the chapter explores the types and classifications of spelling errors, illustrating how they can affect written Maltese. It then examines traditional and modern spell checking approaches, explaining how each technique works and reviewing their application to other low-resource languages. The discussion also considers each method's potential benefits and relevance for Maltese, considering its specific linguistic challenges. Through this progression, the chapter lays the foundation for adapting established techniques to develop practical and effective tools for Maltese.

2.1 | Maltese Language

The Maltese language has a rich linguistic history spanning over a thousand years. It is the Maltese Islands' national language and is an official language alongside English (Fabri, 2010; Gauci, 2024).

The linguistic development of Maltese has been primarily shaped by Arabic and Romance influences, with English contributing from the nineteenth century onwards. Notably, Maltese is the only European language with a strong Arabic foundation and the only Arabic-derived language written in the Latin script. As a result, the Latin alphabet was adapted for Maltese through the introduction of additional letters, including letters with diacritics (ċ, ġ, ħ, ż) and digraphs (għ, ie) to represent the language's distinctive phonological characteristics. These adaptations have resulted in a Maltese alphabet comprising twenty-four consonants and six vowels (Rosner & Borg, 2023; Zammit, 2017).

Two illustrative examples demonstrating how the Maltese alphabet captures distinctive phonological features are the letter **ħ** and the digraph **għ**. The letter **ħ**, which

includes a diacritic, represents the voiceless pharyngeal fricative, transcribed in the International Phonetic Alphabet (IPA) as /ħ/. This phoneme continues the Arabic ḥ and merges related sounds like h and ḫ. For example, the word **ħajt** (wall or thread) is derived from the Arabic word ḥāyiṭ (wall) or ḫayṭ (thread).

In contrast, the digraph **għ** comes from historical Arabic pharyngeal consonants. It is part of the Maltese alphabet and plays a significant role in the language's orthography and morphology. However, in contemporary Maltese, it is not pronounced. Instead, it typically influences the pronunciation of adjacent vowels, often lengthening them. For instance, in the word **għamel** (to do), it is transcribed in the IPA as /ˈɑːmɛl/. It may also be silent, as in **għajn** (eye), transcribed as /ɑɪn/ (Lucas & Čéplö, 2020).

Maltese has a rich linguistic heritage and is widely spoken throughout the Maltese Islands; however, it is classified as a low-resource language within the Natural Language Processing (NLP) field. A language is classified as low-resource when high-quality datasets and linguistic resources are scarce, in addition to limited computational tools. In the case of Maltese, existing corpora are often limited in size, inconsistently annotated, and frequently characterised by inaccuracies or missing data. These constraints present significant challenges to developing reliable and effective NLP models (Nicholas & Bhatia, 2023; Żammit, 2023).

Despite these limitations, this study aims to make a meaningful contribution by using the available resources to create a reliable and robust spell checker tool. The resulting system is expected to improve the quality of everyday written Maltese and help develop linguistic resources.

The following sections of this chapter outline key background information and techniques that support this work. Before exploring the different methods for building a spell checker, it is essential to understand the various spelling errors because each type presents its own challenges for detection and correction. Section 2.2 explains the different spelling errors and how they are classified based on specific characteristics.

2.2 | Spelling Errors

Spelling errors are common in writing and typically arise from two main causes: cognitive and typographical factors. Cognitive errors occur when a writer does not know the correct spelling of a word, confuses homophones, or experiences learning difficulties such as dyslexia, which affects spelling ability (Hládek et al., 2020).

In contrast, typographical errors do not happen due to knowledge gaps. They occur because of physical limitations when typing. These limitations can include using a

keyboard with a different language layout, which limits access to certain characters, or typing quickly, which increases the likelihood of accidental keystrokes. Typographical errors can be grouped into four types: insertion, omission, substitution, and transposition. Insertion errors occur when two keys are pressed simultaneously or when a neighbouring finger accidentally hits an extra key, adding an unintended letter. Omission errors occur when a key is only partially pressed or completely missed, resulting in a missing letter. Substitution errors involve pressing a neighbouring key instead of the correct one, while transposition errors occur when adjacent letters in a word are reversed (Mitton, 1996).

While typographical and cognitive errors differ in their origins, both ultimately lead to mistakes that affect the form and validity of written words. To systematically describe such errors, researchers have proposed a higher-level classification based on their outcomes. Specifically, spelling errors are often divided into two broad categories: non-word and real-word errors (Ahmed et al., 2009).

Non-word errors involve words that do not exist in the language's dictionary. For example, in Maltese, writing the word "*filodu*" instead of "*filgħodu*" would be a non-word error because the former does not exist in the Maltese dictionary. On the other hand, real-word errors occur when a valid word from the language's dictionary is incorrectly used in a given context. For instance, in Maltese, writing "*Xtrajt **flok** abjad.*" instead of "*Xtrajt **flokk** abjad.*", by using the word "*flok*" instead of "*flokk*", classifies it as a real-word error because both "*flok*" and "*flokk*" exist in the Maltese dictionary but have different meanings (Sooraj et al., 2018).

A more comprehensive overview of the different types of spelling errors in Maltese is provided in Table A.1, located in Appendix A.

Researchers have dedicated significant effort to developing spell checkers that correct non-word and real-word spelling errors in text. This work has led to extensive research across various languages worldwide (Abdulrahman & Hassani, 2022; Kaalep et al., 2022). Building an effective spell checker involves overcoming two main challenges: detecting errors and generating appropriate corrections (Kukich, 1992; Pirinen & Lindén, 2014). A diverse array of techniques, ranging from traditional rule-based methods to modern deep learning approaches, have been proposed to tackle these challenges.

The next sections provide detailed explanations of the techniques used over time to detect and correct spelling errors. Section 2.3 focuses on traditional approaches, while Section 2.4 examines modern techniques developed in recent years.

2.3 | Traditional Techniques

In the early development of spell checkers, researchers relied on traditional techniques such as dictionary look-up, rule-based methods, edit distance algorithms, and n-gram models to detect and correct spelling errors. These methods are more straightforward to implement and understand than more recent deep learning approaches. Nevertheless, they are less effective at handling complex linguistic patterns and detecting real-word errors. Despite their limitations, traditional techniques laid the groundwork for modern spell checkers and played a pivotal role in shaping the field. Many of these methods remain in use today, particularly in hybrid systems or in applications that prioritise simplicity and minimal computational cost (Hládek et al., 2020).

This section explores these traditional approaches in detail. It examines their main components, advantages, and limitations. It also considers their contributions to the evolution of more advanced spell checkers.

2.3.1 | Dictionary Look-up

Dictionary look-up is a foundational technique for detecting spelling errors. It involves comparing each word in the input against a predefined lexicon. Words not found in the dictionary are flagged as potential errors, making this method particularly effective for identifying non-word errors (Mishra & Kaur, 2013; S. P. Singh et al., 2016). Due to its simplicity, it is still employed in modern spell checkers. Its continued use is particularly evident in hybrid systems that combine dictionary-based methods with Machine Learning (ML) models. These systems aim to leverage the strengths of both approaches. ML handles complex error modelling, while dictionary look-up ensures that suggestions conform to basic linguistic rules (Ransing et al., 2024).

While dictionary look-up remains a valuable component in such systems, it is not without limitations. It often fails to recognise proper nouns, domain-specific terms, and real-word errors (Bassil, 2012). Furthermore, although described as straightforward to implement (Kukich, 1992), its computational efficiency declines as dictionary size increases. This becomes especially problematic in large-scale or multilingual applications, where extensive lexicons can hinder performance (Tolentino et al., 2007).

These challenges are discussed in more detail in Section 2.3.1.1 and Section 2.3.1.2. These sections explore methods to improve dictionary coverage and efficiency. Section 2.3.1.3 looks at applications in low-resource settings, while Section 2.3.1.4 focuses on how the technique can be adapted and its relevance for Maltese spell checking.

2.3.1.1 | Dictionary Size and Accuracy

One significant challenge in the dictionary look-up technique is the creation and maintenance of a comprehensive dictionary. A dictionary that is too small increases the likelihood of valid words being incorrectly flagged as errors, resulting in higher False Positive (FP) rates. On the other hand, while expanding the dictionary improves lexical coverage, it may also increase the risk of False Negatives (FNs). This occurs when spelling errors resemble valid words, particularly low-frequency or domain-specific terms, causing genuine errors to remain undetected (Kukich, 1992).

This issue is further exacerbated in morphologically rich languages, such as Maltese, where words are formed through inflection, derivation, and compounding. For instance, a single root such as **k-t-b** (write) can generate a wide range of surface forms. These include both inflectional variants (e.g., kiteb) and derivational forms (e.g., kitba) that differ in word class or grammatical function. Table 2.1 illustrates some of the different word forms derived from **k-t-b**, reflecting grammatical features such as tense, person, and word class (e.g., noun, verb). Therefore, effective spell checking in such contexts must rely on dictionaries that are capable of recognising this morphological variation (Mangion, 1999).

Table 2.1: Examples of Maltese word forms derived from the root *k-t-b*. Source: Author.

Word Form	**Translation**	**Tense**	**Gender**	**Word Class**
kiteb	he wrote	Past	Masculine	Verb
jikteb	he writes	Present	Masculine	Verb
kitbet	she wrote	Past	Feminine	Verb
tikteb	she writes	Present	Feminine	Verb
kitbu	they wrote	Past	Not Applicable (N/A)	Verb
jiktbu	they write	Present	N/A	Verb
kitba	writing	N/A	N/A	Noun
kittieb	writer (m.)	N/A	Masculine	Noun
kittieba	writer (f.) / writers	N/A	Feminine / Plural (context-dependent)	Noun

Earlier work by Peterson (1980) suggested managing dictionary size by using a fixed word count with topic-specific dictionaries. These dictionaries could be merged as needed to reflect domain-specific usage. Although this idea has had an impact, it struggles to scale and adjust to the complexities of morphologically rich and low-resource languages.

More recent methods address these issues through corpus-based and morphology-based techniques. A corpus-based approach looks at word frequency in various textual corpora to ensure that commonly used and relevant terms are included, which improves the accuracy and speed of dictionary look-up (Mati et al., 2021). Morphology-based methods create surface forms from root morphemes, reducing the need for large, predefined word lists (Gamu & Woldeyohannis, 2023).

These methods were selected because of their effectiveness in improving dictionary coverage and adaptability in morphologically complex settings, where surface variation poses a challenge to predefined dictionaries. Their relevance to Maltese lies in the language's rich morphological structure and limited digital resources, making these approaches particularly suitable for the language.

2.3.1.2 | Computational Efficiency

Computational efficiency poses another significant challenge in dictionary look-up techniques, particularly as dictionary size increases. In unoptimised implementations, each input word must be compared sequentially against all dictionary entries. This leads to degraded performance in systems with large vocabularies (Kukich, 1992).

To address this limitation, various data structures have been explored to improve search efficiency. These often involve trade-offs among processing speed, memory usage, and adaptability. Examples include hash tables, Binary Search Trees (BSTs), Frequency-Ordered Binary Search Trees (FOBSTs), and tries, all of which have demonstrated strong performance in terms of access time and retrieval efficiency. These data structures have been widely applied in spell checkers and may offer comparable benefits in the context of Maltese (Chaabi & Ataa Allah, 2022; Cissé & Sadat, 2023; Salifou & Naroua, 2014).

For instance, hash tables allow fast exact-match retrieval, but they may be less effective in handling the morphological richness of Maltese unless a comprehensive lexicon is available (Bento et al., 2015). BSTs, with their support for lexically ordered searches, can facilitate efficient prefix matching, although they require careful balancing to avoid performance degradation (Lin, 2019). FOBSTs offer improved efficiency for frequently accessed words, which could be beneficial if frequency data for Maltese are available (Sheil, 1978). Lastly, tries provide flexible prefix and approximate matching, making

them particularly well-suited to morphologically complex languages such as Maltese (Fredkin, 1960; Reznik, 2005).

2.3.1.3 | Dictionary Look-up Technique in Low-Resource Languages

Section 2.3.1 introduced the dictionary look-up as a foundational technique for spelling error detection. Building on the earlier discussion, this section explores the use of dictionary look-up in spell checkers for low-resource languages, with a particular focus on its role in non-word error detection, dictionary coverage, and overall detection performance.

To evaluate the effectiveness of the dictionary look-up in these systems, it is useful to consider the most commonly reported detection metrics:

- **Detection accuracy:** The proportion of all words (both correct and incorrect) that are correctly classified by the system.
- **True Positive (TP) rate:** The proportion of valid words (correct spellings) correctly identified as valid.
- **True Negative (TN) rate:** The proportion of invalid words (spelling errors) correctly identified as incorrect.

Several studies demonstrate how the dictionary look-up has been adapted in low-resource language contexts, each illustrating different trade-offs between dictionary size, linguistic coverage, and performance.

For Sinhala, Liyanapathirana et al. (2021) developed a rule-based system supported by a dictionary compiled from multiple public sources. It achieved strong detection performance, with a TP rate of 96.6% and a TN rate of 98.2%.

A spell checker for Dinka was proposed by Chol David and Balagadde Ssali (2022), using a 6,976-word dictionary derived from the Dinka New Testament. The system targeted character-level errors such as substitutions, deletions, and transpositions, and achieved a detection accuracy of 98.1%, with a TP rate of 99.1% and a TN rate of 91.36%.

Gamu and Woldeyohannis (2023) introduced a linguistically informed approach for Dawurootsuwa, a morphologically complex Omotic language. Their hybrid system combined a dictionary of 5,123 root words with 2,631 morphological rules to generate surface forms covering inflection, derivation, and compounding. The system achieved a detection accuracy of 90.4%. In this case, the addition of morphological rules was crucial to capture the range of word forms not directly present in the dictionary.

To facilitate direct comparison of the above mentioned studies, Table 2.2 summarises the reported detection performance.

Table 2.2: Detection performance metrics for dictionary look-up technique in low-resource languages. Source: Author.

Language	Detection Accuracy	TP Rate	TN Rate
Sinhala	Not reported	96.6%	98.2%
Dinka	98.1%	99.1%	91.36%
Dawurootsuwa	90.4%	Not reported	Not reported

These studies demonstrate that the dictionary look-up technique can be a practical approach for error detection in low-resource languages. The strong performance observed in Dawurootsuwa, a morphologically rich language, further indicates that such a technique can function effectively in linguistically complex environments.

Building on these findings, Section 2.3.1.4 explores how the approach can be adapted for Maltese and assesses its relevance for developing a Maltese spell checker.

2.3.1.4 | Dictionary Look-up: Relevance and Applicability for a Maltese Spell Checker

In the case of Maltese, a low-resource and morphologically rich language, dictionary-based spell checking depends heavily on the quality and coverage of the underlying dictionary.

Evidence from other low-resource languages supports this. Mati et al. (2021) showed the importance of usage-based lexical coverage in Albanian. Rajan et al. (2025) reported strong performance in a Konkani spell checker using a dictionary with 1,510,514 unique entries. These studies highlight the central role of extensive lexical coverage in overall system performance.

To apply the dictionary look-up technique effectively to Maltese, an extensive or usage-based dictionary is ideal if resources are available. If data is limited, a more practical approach uses a root-word dictionary with morphological rules to generate surface forms. A root-word dictionary improves coverage while limiting resource demands. For efficient word retrieval, various data structures offer trade-offs, but trie data structures are especially well-suited for Maltese. A trie data structure scales with word length and stores common prefixes only once, allowing for efficient retrieval of words. Although

optimised for prefix sharing, tries also represent related word forms efficiently, supporting both broad coverage and fast retrieval in dictionary-based systems (Reznik, 2005).

In conclusion, if dictionary coverage and computational efficiency are correctly handled, dictionary look-up can be a reliable method for finding non-word errors in a Maltese spell checker.

However, detection alone is not enough for a complete spell checker; correction is necessary too. Since dictionary look-up does not provide this function, the next section (Section 2.3.2) looks at the rule-based technique, a fundamental method commonly used for spelling corrections.

2.3.2 | Rule-Based Technique

The rule-based technique is among the earliest and most foundational approaches to error correction in spell checkers. They involve using handcrafted rules developed by linguistic experts tailored to the specific characteristics of a language to rectify spelling errors (Hládek et al., 2020; Martynov et al., 2024).

For example, the rule-based technique is often combined with the dictionary look-up technique. Spelling errors are first detected by identifying words not found in a predefined dictionary. Predefined linguistic rules are then applied to transform the erroneous word into valid dictionary entries (Kukich, 1992).

Rule-based methods played a central role in the development of early spell checkers, particularly before the introduction of ML in NLP. While generally effective for correcting non-word errors, rule-based systems struggle with real-word errors, which require contextual understanding. These limitations stem from the substantial manual effort needed to construct such rules and their limited ability to generalise to unseen or complex errors, particularly those involving grammar or usage (Bijoy et al., 2025; Debattista, 2022).

Section 2.3.2.1 will examine how the rule-based technique has been applied in low-resource language spell checkers.

2.3.2.1 | Rule-Based Technique in Low-Resource Languages

Rule-based techniques, which rely on handcrafted linguistic rules addressing morphology, orthography, syntax, and grammar, have been widely applied in the development of spell checkers for low-resource languages. Their effectiveness and adaptability have been explored in various low-resource contexts. For example, Dixit et al. (2005) proposed a morphology-based system for Marathi that corrects spelling errors using linguistically informed morphological and orthographic rules.

Similarly, Rao et al. (2012) developed a rule-based spell checker for Telugu, employing morphological analysis and Sandhi splitting to handle the complexities of this agglutinative language. In more recent work, Mengliev et al. (2023) designed a syntactic rule-based analysis system for Uzbek, while Gamu and Woldeyohannis (2023) constructed a morphology-based checker for Dawurootsuwa using over 2,500 rules to generate inflected and derived forms.

A rule-based grammar checker was also developed for Afan Oromo by Tesfaye (2011), consisting of 123 rules applied across five stages: tokenisation, Part-of-Speech (POS) tagging, stemming, grammatical relation analysis, and correction suggestion.

Although rule-based systems rely on manually constructed rules, they are valued for their precision and adaptability to the specific linguistic structures of low-resource languages. Studies such as Gondaliya et al. (2022) highlight their effectiveness in improving spelling and grammatical accuracy. However, as noted by S. Singh and Singh (2020), these systems tend to perform less effectively in detecting real-word errors, which typically require contextual understanding beyond the capacity of fixed rule sets.

Given their effectiveness in morphologically rich and syntactically diverse low-resource languages, rule-based techniques are also of particular interest for Maltese. Section 2.3.2.2 examines how this technique can be applied and assesses its relevance for the development of a Maltese spell checker.

2.3.2.2 | Rule-Based Technique: Relevance and Applicability for a Maltese Spell Checker

Rule-based techniques offer a practical approach to error correction in Maltese. Their application, however, presents both opportunities and challenges. As a morphologically rich language, Maltese exhibits complex inflectional patterns, compounding, and orthographic variation that are difficult to capture using static rules entirely. While rule-based systems offer precision and transparency, their success depends on the rule sets' quality, coverage, and linguistic accuracy. Creating such rules for Maltese would demand significant manual effort and linguistic expertise. Nevertheless, as shown in the Dawurootsuwa and Telugu systems, rule-based systems can be more scalable by focusing on the root-word derivation and morphological generation. For Maltese, a component-based rule-based framework that begins with affix removal and pluralisation and extends to compound formation and syntactic heuristics may provide a practical and extensible solution. This layered approach could mitigate resource limitations while ensuring coverage of the language's most productive word formation processes.

While the rule-based technique is language-dependent and has demonstrated effec-

tiveness, alternative methods that require less manual rule development have also been explored. One such approach involves using edit distance algorithms, which estimate word similarity algorithmically and provide a flexible basis for automatic error correction. A more detailed overview of these algorithms is provided in Section 2.3.3.

2.3.3 | Edit Distance Algorithms

Edit distance algorithms are computational algorithms that measure the similarity between two strings by calculating the minimum number of operations required to transform one string into another. These operations typically include insertion, deletion, and substitution of characters, with some variants also considering transposition, while others use only a subset of these four operations (C. Zhao & Sahni, 2020).

Levenshtein Distance, also known as the Minimum Edit Distance (MED) algorithm, is a widely recognised method for calculating edit distance, introduced by Levenshtein et al. (1966). This algorithm computes the minimum number of single-character operations: insertions, deletions, or substitutions required to transform one string into another. For example, changing the word "kitten" into "sitting" requires three edit operations, resulting in a Levenshtein distance of three (Iqbal et al., 2013; Khalid et al., 2022).

A notable variant of the Levenshtein distance is the Damerau–Levenshtein distance, first proposed by Damerau (1964). This extension includes the transposition of adjacent characters in addition to insertions, deletions, and substitutions, thereby improving its capacity to capture a broader range of character-level transformations (Kokong et al., 2024).

Beyond Levenshtein-based models, alternative edit distance algorithms have been developed. For instance, Hamming distance only permits substitutions and requires strings of equal length, making it suitable for fixed-length code comparisons. In contrast, the Jaro distance and its refinement, the Jaro-Winkler distance, focuses on character transpositions and partial matches, particularly in short strings such as names (Friendly, 2019; Sarkar, 2016; C. Zhao & Sahni, 2019). Other variants, such as Optimised Edit Distance and Reverse Edit Distance, have been proposed to improve runtime efficiency or reduce the number of comparisons when searching large lexicons (Iqbal et al., 2013).

Depending on the intended application, these algorithms offer different trade-offs between accuracy, operation flexibility, and computational cost.

In the context of spell checkers, edit distance algorithms can serve two primary purposes. The first involves error detection, which identifies significant differences between

two strings. The second is error correction, which suggests the closest valid matches based on minimal edit operations (Boroujeni et al., 2020).

To illustrate their practical application, Section 2.3.3.1 reviews how edit distance algorithms have been employed in spell checkers for different low-resource languages.

2.3.3.1 | Edit Distance Algorithms in Low-Resource Languages

This section presents studies demonstrating how edit distance algorithms have been used to develop spell checkers for different low-resource languages.

Khairul Islam et al. (2019) developed a Bangla spell checker using the Levenshtein Distance algorithm, employing a dictionary-based approach to detect spelling errors and spelling errors were corrected through minimal edit operations.

Similarly, Patil et al. (2021) applied the Levenshtein Distance algorithm to create a Marathi spell checker based on a dictionary of over 929,000 entries. The study highlights challenges in candidate ranking when multiple suggestions share the same edit distance and proposes using n-gram models to improve contextual disambiguation.

For Indonesian, Santoso et al. (2019) compared the Levenshtein and Damerau–Levenshtein algorithms. It was concluded that the Damerau–Levenshtein algorithm outperforms the Levenshtein Distance algorithm, particularly in handling common typographical errors. This improvement is attributed to the inclusion of the transposition operation, which allows the system to address adjacent character swaps better.

In a recent study, Cissé and Sadat (2023) proposed a hybrid spell checking system for Wolof that integrates dictionary look-up, rule-based error detection, and a Weighted Levenshtein Distance algorithm. The system includes pre-processing steps such as normalisation, punctuation removal, and tokenisation. Following pre-processing, the system detects errors using orthographic rules and a manually constructed dictionary of 1,410 words. For out-of-dictionary words, correction candidates are generated by calculating the edit distance with language-specific weighting to account for Wolof-specific errors, particularly those influenced by French orthography. Candidates are then ranked by edit distance, and the closest match is selected. The study also highlights key limitations, including the restricted dictionary size and lack of semantic context, suggesting that future work could incorporate context-aware or deep learning approaches.

Another approach is presented by Mukazhanov et al. (2023), who developed an augmented Damerau–Levenshtein algorithm tailored for Kazakh. This method incorporates language-specific phonetic and orthographic substitution rules into the edit distance metric, with particular attention to confusion between visually or phonetically

similar Cyrillic characters. The authors suggest that future refinements could include probabilistic or neural techniques to improve contextual sensitivity.

These studies indicate that edit distance algorithms effectively correct non-word errors in various low-resource languages. However, these algorithms are limited in handling context-sensitive corrections. This limitation reduces their effectiveness in addressing real-word errors, prompting the need to use more advanced techniques, such as ML models, to overcome these challenges.

An overview of the correction performance reported in the aforementioned studies is presented in Table 2.3.

Table 2.3: Detection performance metrics for edit distance algorithms in low-resource languages. Source: Author.

Language	**Algorithm**	**Correction Accuracy**	**Suggestion Rate**
Marathi	Levenshtein Distance	85.56%	Not reported
Indonesian	Damerau–Levenshtein Distance	75%	Not reported
Indonesian	Levenshtein Distance	73%	Not reported
Wolof	Weighted Levenshtein Distance	98.31%	93.33%
Kazakh	Augmented Damerau–Levenshtein	97.2% (Kazakh-specific); 92.8% (general errors)	Not reported

These results demonstrate the adaptability of edit distance algorithms across a range of low-resource languages. The following section, Section 2.3.3.2 considers how these algorithms can be applied to a Maltese spell checker and evaluates their relevance for this specific application.

2.3.3.2 | Edit Distance Algorithms: Relevance and Applicability for a Maltese Spell Checker

Given the success of edit distance algorithms in correcting non-word errors across diverse low-resource languages, as discussed in Section 2.3.3.1, these algorithms hold strong potential for use in a Maltese spell checker. Maltese, a Semitic language with influences from Romance and English, presents unique orthographic and morphological features. Given these characteristics, a Maltese spell checker could benefit from

adaptations similar to those developed for Wolof (Cissé & Sadat, 2023) and Kazakh (Mukazhanov et al., 2023). Specifically, incorporating edit weights, phoneme-aware rules, and hybrid strategies, such as combining rule-based and dictionary look-up methods, may provide a strong foundation for improving accuracy. Furthermore, edit distance methods can serve as an efficient first-pass filter before applying more context-aware techniques, such as those based on ML models. They may also function as a refinement step to enhance the output of such models by enforcing orthographic consistency or ranking candidates based on string similarity.

While edit distance algorithms are practical for basic error correction, they are limited in handling real-word errors, which typically require contextual understanding. To address these limitations, researchers have proposed techniques such as n-gram models and ML approaches (Cissé & Sadat, 2023; Mukazhanov et al., 2023; Patil et al., 2021).

Section 2.3.4 examines the n-gram model. Although statistical, it is often considered a traditional technique due to its simplicity, interpretability, and low reliance on large-scale data. For this reason, it is grouped with the previously discussed techniques as part of the traditional approach to developing a spell checker.

2.3.4 | N-gram Model

N-gram models are based on the idea that the likelihood of an element in a sequence depends only on a limited number of preceding elements. This concept was formalised in early probability theory by Markov (1913) and later applied to language modelling by Shannon (1951) (Chaabi & Ataa Allah, 2022).

An n-gram refers to a sequence of n elements of a given text and can be classified as character-based or word-based. Character-based n-grams consist of n consecutive characters, while word-based n-grams contain n consecutive words. N-grams can vary in size, and some common sizes have specific names for easy identification. For example, an n-gram of size 1 is termed a unigram ($n = 1$), of size 2 a bigram ($n = 2$), and of size 3 a trigram ($n = 3$). N-grams larger than three are simply referred to as n-grams (Mishra & Kaur, 2013; Mohapatra et al., 2013).

In spell checking, n-grams can be used for both error detection and correction. For error detection, the text is divided into n-grams and compared with the most frequent n-grams found in correctly written text. Instead of relying on a dictionary, the system identifies n-grams that are infrequent or statistically improbable. Words containing these unusual or improbable n-grams are marked as potential spelling errors (Randhawa & Saroa, 2014; Sakuntharaj, 2022).

For error correction, n-grams can be used with or without a dictionary. When a dictionary is available, n-grams help assess the similarity between the spelling error and potential corrections by comparing shared sequences of letters. The word with the highest number of matching n-grams is selected as the best candidate for correction. This candidate is then validated by checking it against the dictionary to confirm its correctness. On the other hand, when used without a dictionary, n-grams are used to identify unusual or statistically infrequent letter combinations in the spelling error. These n-grams are classified as invalid if they occur infrequently or are absent in a reference corpus of correctly spelled text. A valid correction is then generated by replacing these invalid n-grams with more common ones. For example, the word "recieve" may be corrected to "receive," as the trigram "cei" is more statistically frequent in English than the trigram "cie". This approach relies on statistical patterns observed in a large corpus of correctly spelled words, making it possible to generate corrections even without a dictionary (Allkivi-Metsoja & Kippar, 2023; El Atawy & Abd ElGhany, 2018).

N-gram-based techniques have shown strong potential in low-resource settings due to their algorithmic simplicity, language independence, and reliance on statistical patterns present in raw text rather than on extensive lexical resources (Flores & Radev, 2022; Wojcicki & Zientarski, 2024).

To explore this further, Section 2.3.4.1 reviews how n-gram models have been employed in spell checkers for low-resource languages.

2.3.4.1 | N-gram Model in Low-Resource Languages

Previous studies have shown that n-gram models can effectively detect and correct spelling errors in low-resource languages, as they capture contextual patterns often missed by rule-based techniques and edit distance algorithms.

For example, Ambili et al. (2016) developed a hybrid system for Malayalam that combines dictionary look-up with n-gram analysis. Spelling error detection is performed using dictionary look-up, while correction involves comparing the n-grams of spelling errors with those of dictionary entries, with corrections ranked by a similarity coefficient. This approach performed well, particularly in handling single-character errors.

Other hybrid approaches using n-grams have also been explored. Sharma and Jain (2013) integrated an n-gram model with the Damerau-Levenshtein Distance algorithm for Hindi. Candidate corrections were filtered by edit distance and further ranked using word frequency and contextual n-gram information, improving the selection of the most appropriate correction.

For Tamil, Sakuntharaj and Mahesan (2018) proposed a bigram probabilistic model targeting real-word errors. Integrated with the MED algorithm, their approach achieved 98% accuracy in generating suitable corrections.

In another study, Sakuntharaj and Mahesan (2021) applied unigrams, bigrams, and trigrams to address spelling and grammatical issues in Tamil. By leveraging contextual patterns, the system could also predict missing words, an essential feature for morphologically rich languages. The system demonstrated an accuracy of 99%, which a language expert validated.

In a more recent study, Oluwaseyi et al. (2024) developed a spell checker for Yorùbá, combining the Levenshtein Distance algorithm with n-gram-based context evaluation. Candidate corrections were ranked by both structural similarity and contextual fit, resulting in a correction accuracy of 71.50% on a test set of 2,000 sentences.

These studies illustrate the potential of n-gram models to address the linguistic complexities of low-resource languages. When used alongside dictionary look-up and edit distance algorithms, they enhance error correction by incorporating contextual information (J. Zhao et al., 2017). Moreover, they offer a valuable basis for assessing the applicability of n-gram models to Maltese, as discussed in Section 2.3.4.2.

2.3.4.2 | N-gram Model: Relevance and Applicability for a Maltese Spell Checker

N-gram models are particularly relevant for a Maltese spell checker, as they provide a lightweight, language-independent approach capable of addressing both non-word and real-word errors. Unlike rule-based systems or dictionary-only methods, n-gram models use statistical patterns in text to identify improbable character or word sequences and to rank correction candidates based on contextual fit. This is especially important for Maltese, which exhibits considerable orthographic variation.

To adapt n-gram models for Maltese, character- or word-based models can be trained on available raw text, even if unannotated. These models can flag statistically unlikely sequences as potential spelling errors. Correction candidates may then be ranked using n-gram frequency data, either independently or in conjunction with edit distance or dictionary-based techniques. Coverage can also be enhanced by incorporating common morphological features in Maltese, such as affixation and compounding. This combined use of statistical modelling and linguistic knowledge makes n-gram models a practical and adaptable solution to develop a Maltese spell checker.

Despite these advantages, n-gram models face limitations in handling complex errors and dynamic grammatical structures. Their effectiveness also depends on the coverage of n-grams, as sparse or limited sequences can reduce accuracy. To address these

challenges, integration with syntactic rules or ML models may be necessary for a more comprehensive coverage (Patil et al., 2021).

2.3.5 | Conclusion

In conclusion, Section 2.3 has explored the key traditional techniques used in developing spell checkers. These include dictionary look-up (Section 2.3.1), rule-based methods (Section 2.3.2), edit distance algorithms (Section 2.3.3), and the n-gram model (Section 2.3.4). Building on this foundation, the next section, Section 2.4, introduces modern approaches. It focuses on statistical models, deep learning, and transformer-based methods.

2.4 | Modern Techniques

With advances in ML and NLP, modern spell checkers have increasingly adopted models based on statistical learning and deep learning. These techniques go beyond surface-level corrections by learning contextual patterns from large corpora, enabling them to handle non-word and real-word errors more accurately.

Statistical models laid the foundations for this shift, while deep neural networks and Transformer architectures have further improved the ability to understand language nuances. Although these approaches require more computational resources and data, they significantly improve adaptability and precision (Chai et al., 2024; Cissé & Sadat, 2024; Zhu et al., 2019).

This section will explore these modern approaches, highlighting their methodologies, advantages, and impact in the field of spell checking.

2.4.1 | Statistical Learning

Statistical learning introduced probabilistic, data-driven approaches that enabled models to learn language patterns directly from annotated corpora, improving a wide range of NLP tasks, including spell checking. Unlike rule-based systems that rely on predefined dictionaries or handcrafted rules, these models learn from measurable input properties known as features. These features may include letter sequences (n-grams), word co-occurrence patterns, or keyboard proximity. The selection and quality of these features are crucial, as they guide what the system prioritises and learns from during training. This process, known as feature engineering, is central in building effective models (Cheekatimalla, 2025; X. Zhang, Mao, & Cambria, 2023). Widely used statistical models

for spell checking include the noisy channel model, Hidden Markov Models (HMMs), logistic regression, and Support Vector Machines (SVMs) (Hládek et al., 2020).

These models offer advantages such as interpretability and lower computational requirements compared to deep learning architectures, making them a practical alternative for low-resource languages where large annotated datasets or computational resources may be limited. However, they often struggle to capture abstract or context-sensitive language patterns. These limitations are addressed by deep learning approaches (further discussed in Subsection 2.4.2), which leverage neural network architectures to automatically learn complex and hierarchical representations from large-scale data (Rai et al., 2024).

The following sections (Sections 2.4.1.1–2.4.1.4) provide a detailed overview of these statistical models and examine their application to various low-resource languages, highlighting both their strengths and limitations in such contexts.

2.4.1.1 | Noisy Channel Model

The noisy channel model is a probabilistic framework for spelling correction, which views spelling errors as correct words distorted during their transmission through a noisy communication channel. The fundamental principle underpinning this model is that typographical errors arise from distortions such as insertions, deletions, substitutions, and transpositions. To recover the intended word w given an observed error x, the model applies Bayesian inference and selects the word that maximises the posterior probability $P(w|x)$. Using Bayes' Rule, this can be reformulated using Equation 2.1.

$$\hat{w} = \arg\max_{w \in V} P(x \mid w) \cdot P(w) \tag{2.1}$$

In Equation 2.1, $P(w)$ represents the prior probability of a word, typically derived from a language model, and $P(x|w)$ is the likelihood or channel model, reflecting the probability that the correct word w would be transformed into the observed error x.

This model was first applied to spelling correction by Kernighan et al. (1990), widely recognised for a foundational implementation. Their approach involved using confusion matrices to capture probabilities of different character-level errors based on empirical data (Jurafsky & Martin, 2025).

To improve the error model, these matrices can be refined through an iterative process: the system initially assigns uniform probabilities, applies the correction algorithm to a corpus of known spelling errors, and updates the matrices based on the predicted corrections. Repeating this process constitutes an instance of the Expectation-Maximization (EM) algorithm, allowing the system to better align with real-world error distributions.

Although the model was initially developed for non-word spelling errors, it can be extended to handle real-word errors by incorporating more sophisticated language modelling techniques (Jurafsky & Martin, 2025).

2.4.1.2 | Hidden Markov Model

HMMs provide a probabilistic framework for spelling correction by modelling the relationship between a user's intended input and the actual text produced.

In this framework, the correct sequence of words or characters is represented as a series of hidden states, while the typed sequence, which may contain spelling errors, is treated as the observed input.

The model uses two key types of probabilities to link these components: transition probabilities and emission probabilities. Transition probabilities capture how likely one correct word is to follow another, based on patterns in natural language usage. Emission probabilities, by contrast, quantify the likelihood of an observed input being generated from a specific intended word. These emission values are typically derived from common typing errors such as substitutions, omissions, or transpositions. When the emission probability of an observed input is unusually low for a given intended word, the model treats the input as a potential spelling error.

By combining transition and emission probabilities, the HMM estimates the most likely sequence of intended words or characters that could have generated the observed, noisy input. The Viterbi algorithm is typically employed to efficiently compute this most probable sequence, enabling accurate decoding. As a result, the model can effectively correct both non-word and real-word spelling errors (Jurafsky & Martin, 2025; Tarniceriu et al., 2015).

2.4.1.3 | Logistic Regression

Logistic regression, originally developed in statistics and widely used in the 1960s, entered the field of NLP more prominently in the 1990s, often under the name Maximum Entropy Modelling (MaxEnt). Though conceptually identical, it was introduced independently of its statistical origins and applied to core NLP tasks such as POS tagging, parsing, language modelling, and text classification, where it established itself as a foundational probabilistic method (Jurafsky & Martin, 2025).

The development of spell checkers is another specific application of logistic regression in NLP, particularly for spelling error detection and candidate ranking. As a probabilistic classifier, it estimates the likelihood that a given word, character, or phrase is incorrect based on a rich set of contextual and linguistic features. These may include

the surrounding word context, POS tags, character-level patterns, and word frequency statistics (D. Han & Chang, 2013).

During training, the model learns weights for each feature by maximising the likelihood of the observed data, allowing it to assign calibrated probabilities to candidate tokens. In practical systems, this enables the model to either flag potentially incorrect tokens or rank correction suggestions by their likelihood, supporting both detection and correction tasks (Remse et al., 2016; P. Wang et al., 2014).

2.4.1.4 | Support Vector Machines

SVMs are supervised ML models widely used in NLP for classification and ranking tasks. At their core, SVMs operate by identifying an optimal hyperplane—a decision boundary that maximises the margin between different classes in a high-dimensional feature space. When data are not linearly separable, SVMs apply kernel functions to project inputs into a higher-dimensional space where a separating hyperplane can be constructed. This geometric property allows SVMs to generalise effectively to unseen data. Figure 2.1 illustrates this concept, showing how the hyperplane separates two classes with the widest possible margin.

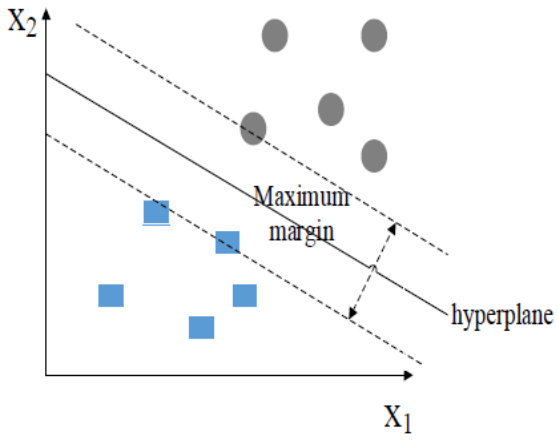


Figure 2.1: Illustration of a Support Vector Machine hyperplane separating two classes with the maximum margin. Source: Guo and Jin (2020).

In the context of spelling correction, SVMs are employed for both error detection and correction candidate ranking. For detection, each word or character is classified as correct or incorrect using features such as character n-grams and dictionary presence. For ranking, SVMs reorder correction candidates generated by baseline models, based on features like edit distance, language model scores, and phonetic resemblance. This

can be achieved either through confidence scores from standard classification SVMs or via dedicated ranking SVM frameworks. Ranking is particularly useful when multiple possible corrections exist, as it helps prioritise the most likely options.

In hybrid systems combining Statistical Machine Translation (SMT) or language models with ML, SVMs refine candidate selection and improve overall correction precision. Numerous studies have demonstrated that SVM-based systems outperform traditional techniques such as rule-based or n-gram-only approaches, particularly in feature-driven or resource-constrained settings, due to their ability to integrate diverse linguistic and contextual features in a principled way (Guo & Jin, 2020; Liu et al., 2015; Yang et al., 2006).

2.4.1.5 | Statistical Learning in Low-Resource Languages

Among statistical learning methods applied to spelling correction in low-resource languages, the noisy channel model and HMM are the most widely adopted. Their ability to function effectively with small, unlabeled corpora makes them well-suited to resource-scarce linguistic settings, especially when compared to methods such as logistic regression and SVMs, which typically require larger volumes of annotated data.

One application of the noisy channel model is demonstrated by Luitel et al. (2024) in their work on Nepali, a language with limited annotated resources. To address this, they train a small word-based Transformer language model for contextual understanding and extract probabilistic error rules from unprocessed text in an unsupervised manner. These components are combined in a noisy channel framework to estimate the most probable intended word given a noisy input. Using a 1.2 Gigabyte (GB) corpus, the system successfully corrects both non-word and real-word errors, demonstrating the effectiveness of this approach in low-resource settings.

Similarly, Gezmu et al. (2018) employ a noisy channel approach to develop a portable spelling correction system for Amharic. Their method leverages corpus-derived knowledge by compiling a term list from frequent words in curated, proofread corpora. The model applies a smoothed trigram language model and a generalised error model to detect and correct non-word errors. Additionally, it accounts for Amharic's syllabic script using transliteration. By normalising script inconsistencies and ranking candidate corrections using trigram probabilities, the system outperforms other spell checkers, demonstrating how the noisy channel model can be effectively adapted for morphologically rich and structurally complex languages.

In another study, Sheykholeslam et al. (2012) apply the noisy channel model to build a spell checker for Persian, using incorrect–correct word pairs extracted from digitised

books. Their method ranks possible corrections by likelihood, achieving notable improvements in accuracy despite the absence of large-scale training data.

Transitioning to HMM-based approaches, several works demonstrate the efficacy of HMMs in capturing contextual dependencies for spelling correction. Hladek et al. (2013) propose an HMM-based model for Slovak that applies the Viterbi algorithm to identify the most probable sequence of correct words from noisy input. The model integrates statistical language modelling with contextual word probabilities, handling both non-word and real-word errors effectively in an unsupervised setting, especially useful for processing large corpora without manual annotation.

In the context of Indonesian, Kamayani et al. (2011) present a document-level spelling checker that integrates HMMs for ranking correction candidates. The system begins with error detection using binary search and hashing, applies similarity scoring through a forward-reverse dictionary, and ranks candidates using HMM.

Further refinement of the HMM approach is seen in the work of Soleh and Purwarianti (2011), which combines morphological analysis with probabilistic modeling to improve non-word error correction. A root-word list supported by a morphological analyser reduces reliance on full lexicons, while the error correction relies on a similarity-based metric. The HMM is then used to contextually rank correction candidates. The model proves particularly effective for affixed and composite word forms, and significantly improves upon bigram-based ranking methods used in earlier Indonesian spell checkers.

Together, these studies affirm the adaptability and effectiveness of both noisy channel models and HMMs in spell checkers for low-resource languages. Noisy channel methods excel in integrating error modelling with language fluency, while HMMs offer robust contextual decision-making.

Based on these observations, statistical learning techniques emerge as a promising approach for application to Maltese. Section 2.4.1.6 outlines the relevance of statistical learning to Maltese and discusses its applicability in the development of a spell checker for the language.

2.4.1.6 | Statistical Learning: Relevance and Applicability for a Maltese Spell Checker

Statistical learning offers a practical pathway for developing spell checkers for Maltese. By leveraging features such as character n-grams, word co-occurrence patterns, and keyboard proximity, models like the noisy channel model or HMM can learn from unlabelled or minimally labelled data. Additionally, Their lower computational requirements make them especially suitable for languages where linguistic resources and in-

frastructure are constrained.

In conclusion, statistical learning approaches have proven to be more effective than traditional techniques. However, they still show limitations, especially with complex language patterns. To address these challenges, recent research has shifted its focus toward deep learning, which utilises neural architectures to automatically learn linguistic features from large datasets and achieve stronger generalisation. Deep learning techniques will be further discussed in Subsection 2.4.2.

2.4.2 | Deep Learning

Traditional techniques and statistical learning laid the foundation for NLP applications such as spell checkers, but have struggled with complex, context-dependent errors. The advent of deep learning marked a significant advancement, addressing the limitations of earlier methods and achieving state-of-the-art performance. By processing large datasets and capturing contextual nuances, deep learning models have significantly improved spell checkers.

Deep learning is a branch within the broader fields of ML and Artificial Intelligence (AI), as illustrated in Figure 2.2. These models use multiple processing layers that enable them to autonomously generate accurate outputs, making them particularly effective at handling complex data, such as text. The depth of these layers improves their ability to understand context, which is crucial for tasks such as spell checking. As large datasets have grown, deep learning models have been able to identify complex patterns, significantly improving spell checking, particularly for real-world errors that require contextual awareness (Bijoy et al., 2025; Lubis et al., 2023).

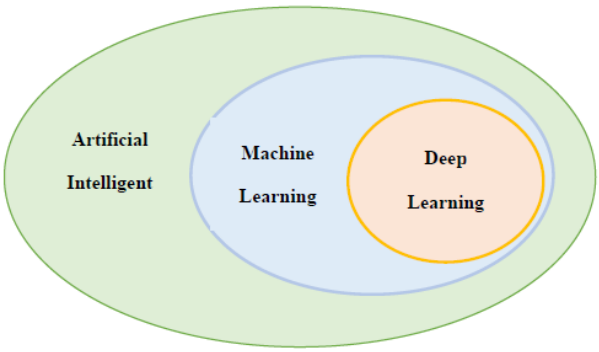


Figure 2.2: Deep learning in relation to AI and ML. Source: Perumal et al. (2024).

A common deep learning approach for spell checking involves the use of sequence-

to-sequence (Seq2Seq) models, which take a sequence, typically a word or sentence, as input and generate a corrected output sequence. When entire sentences are used as input, these models can leverage broader contextual information, leading to more accurate corrections. Seq2Seq architectures are frequently implemented using Recurrent Neural Networks (RNNs), particularly with variants such as Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) networks (Samsudin & Hassen, 2023).

The following sections provide a detailed overview of the LSTM and GRU network architectures, as well as their application in low-resource language settings. Specifically, Section 2.4.2.1 examines the LSTM architecture, while Section 2.4.2.2 focuses on the GRU model. Section 2.4.2.3 then explores the use of these models in the context of spell checkers for low-resource languages, while Section 2.4.2.4 discusses the relevance and applicability of deep learning to the development of a Maltese spell checker.

2.4.2.1 | Long Short-Term Memory

An LSTM is a specialised type of RNN, designed by Hochreiter and Schmidhuber (1997) to address the vanishing gradient problem commonly encountered by traditional RNNs. One of the key advantages of an LSTM over a traditional RNN is its ability to retain information over longer periods of time (Arumugam & Shanmugamani, 2018), which makes it particularly effective for handling sequential data (S. Wang & Jiang, 2015).

The original LSTM architecture proposed by Hochreiter and Schmidhuber (1997) includes a memory cell and two gates, the input and output gates. The main idea behind the LSTM architecture is to have a memory cell that stores information over time, with gates to regularise the flow of information in and out of the memory cell (Greff et al., 2017).

However, Graves and Schmidhuber (2005) highlight that the most commonly used LSTM architecture in the literature includes several enhancements originally proposed by Gers and Schmidhuber (2000) and Gers et al. (1999). This modified LSTM architecture consists of a memory cell, peephole connections, and three gates: input, output, and forget (Greff et al., 2017). Figure 2.3 presents a schematic representation of this architecture.

In an LSTM architecture, as shown in Figure 2.3, the gates operate together to control the flow of information into, out of, and within the memory cell. While all gates (forget, input, and output) are computed simultaneously at each time step, they follow a logical processing order that begins with the forget gate.

The forget gate decides which information from the previous memory cell state should be discarded. It receives the current input, the previous hidden state, and the

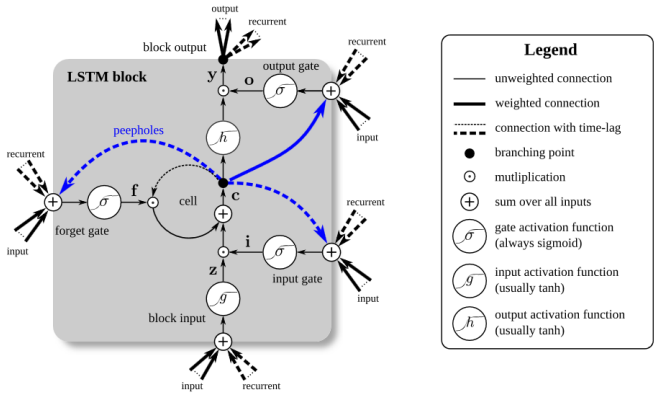


Figure 2.3: Diagram of the commonly used LSTM architecture in the literature. This is the LSTM architecture that contains additional improvements from the original architecture. Source: Greff et al. (2017).

previous memory cell state, along with a bias vector. A sigmoid activation function is applied to these inputs to produce an output vector, which is then used to perform element-wise multiplication with the previous memory cell state. This operation determines how much of the old memory should be retained or discarded. Specifically, the forget gate multiplies the memory cell's previous state by a factor between 0 (for complete removal) and 1 (for full retention), where this factor is influenced by the current input and the previous hidden state (Sooraj et al., 2018).

Following the forget gate, the input gate controls the addition of new information to the memory cell over multiple time steps, ensuring that important data is written while preventing irrelevant data from being stored when the gate is closed. Once this data is written, the output gate manages what information is passed from the memory cell to become the hidden state. It uses a factor between 0 and 1 to regulate how much of the memory cell's internal state is exposed as the hidden state. The hidden state is a vector that captures relevant information learned from previous time steps, allowing the LSTM to retain a memory of what has been processed so far, which is essential for handling sequential data (Hochreiter & Schmidhuber, 1997).

2.4.2.2 | Gated Recurrent Unit

The GRU architecture, introduced by Cho et al. (2014), is a simplified variant of the RNN that addresses the vanishing gradient problem, much like the LSTM. While both use gating mechanisms to regulate information flow, the GRU achieves this with a more streamlined structure by combining the input and forget gates into a single update gate and omitting separate memory cells. This makes the GRU computationally efficient

while retaining the capacity to model long-term dependencies (Chung et al., 2014). A visual representation of the GRU architecture is shown in Figure 2.4.

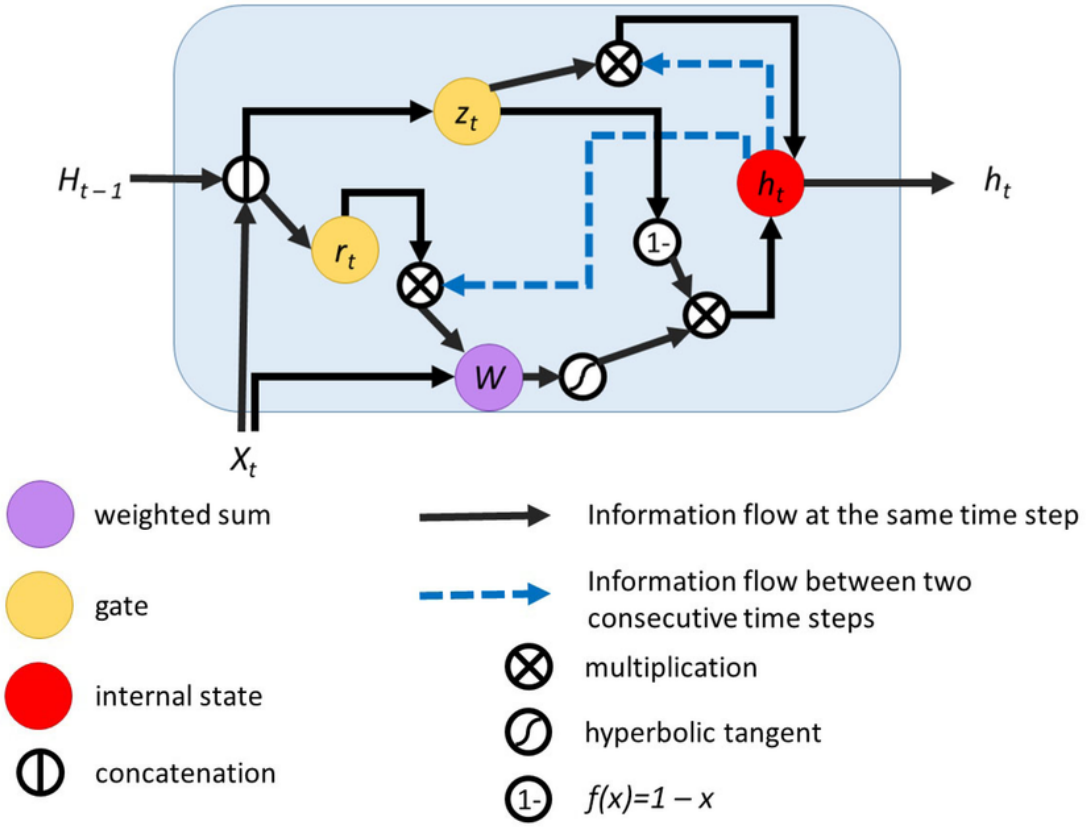


Figure 2.4: Detailed schematic of a GRU cell illustrating the internal operations and information flow. The update gate z_t controls how much of the previous hidden state h_{t-1} is carried forward, while the reset gate r_t determines how much of the past information to forget when computing the candidate activation. Source: Nistor et al. (2021).

The GRU consists of three primary components, which are the update gate, the reset gate, and the current memory content. These components enable the GRU to selectively update and retain important information, thereby effectively capturing long-term dependencies in sequential data (Shiri et al., 2023).

The update gate regulates how much of the current input should be ignored and determines the extent to which past information should be retained and integrated with the new input. To compute the update gate, the previous hidden state is concatenated with the current input, followed by a linear transformation and a sigmoid activation function. When activated, the update gate creates a shortcut connection, allowing the

previous memory state to directly influence the final memory state. This mechanism helps mitigate the vanishing gradient problem.

On the other hand, the reset gate, regulates how much of the past information should be forgotten. It controls the influence of the previous time step's memory on the current input and is computed similarly to the update gate.

Finally, the current memory content is determined by combining the reset gate's influence with the transformed previous hidden state and the current input. This combination is passed through a hyperbolic tangent activation function, producing the candidate activation (Shen et al., 2018; Shiri et al., 2023).

2.4.2.3 | Deep Learning in Low-Resource Languages

Numerous studies have demonstrated the effectiveness of deep learning models in spell checkers. This section examines how such models have been applied to correct spelling errors in low-resource languages.

Gamo et al. (2024) evaluated several neural network architectures, including LSTM, GRU, RNN, and Bidirectional Long Short-Term Memory (BiLSTM) for Wolaita, finding BiLSTMs to be most effective, achieving an accuracy of 98.12%.

In another study, Ahmadzade and Malekzadeh (2021) proposed a deep learning-based spell checker for Azerbaijani using an LSTM-based encoder–decoder architecture with attention. Designed to handle the language's complex morphology, the model was trained on synthetic spelling errors and evaluated on real-world errors. It demonstrated strong generalisation across different edit distances, achieving an F1-score of 75% for exact matches, 90% for one-character edits, and 96% for two-character edits.

Malayalam, a low-resource language, was the focus of a deep learning-based spell checker developed by Sooraj et al. (2018), who used a character-level LSTM model for spelling error detection. The model was trained on a Malayalam news corpus to learn character sequence probabilities, enabling it to flag unlikely word forms as errors. A custom character-splitting method, which preserved compound characters, significantly improved detection accuracy over standard Unicode splitting. For error correction, the system estimated the probability of valid characters at the detected error position and generated candidate words, which were ranked based on their overall likelihood. The model achieved an F1-score of 0.63 and accuracy of 55.2% for error detection, with the correct correction appearing in the top 5 suggestions 100% of the time.

Further contributions were made by S. Singh and Singh (2021), who proposed a deep learning approach for Hindi, employing a Bidirectional Recurrent Neural Network (BiRNN) with LSTM cells within an attention-based encoder-decoder framework.

The model was designed to automatically detect spelling errors and substitute them with the most likely corrections. A publicly available monolingual corpus was used for training and testing, with evaluation performed in two scenarios. When tested on a dataset generated via a split function, the model achieved a precision of 86%, recall of 72%, F-measure of 78%, and accuracy of 80%. Testing on a manually generated dataset produced a precision of 81%, recall of 72%, F-measure of 76%, and accuracy of 74%.

The effectiveness of deep learning models for spelling correction in low-resource languages was further demonstrated by Kasmaiee et al. (2023). An encoder-decoder architecture based on the LSTM model was developed to construct a spell checker for Persian, which outperformed traditional rule-based methods by achieving an accuracy of 87%, precision of 70%, recall of 89%, and F-measure of 78%.

Additionally, Lubis et al. (2023) employed a Convolutional Neural Network (CNN)-based approach to develop a spell checker for Indonesian. This model was trained on 10,000 social media entries and evaluated under different text pre-processing scenarios, including stemming, case folding, and stop word removal. It was found that the combination of deep learning with comprehensive pre-processing significantly improved performance, with an accuracy of up to 89% achieved when all pre-processing steps were applied.

To facilitate direct comparison of the deep learning approaches discussed above, Table 2.4 summarises the reported detection and correction performance across different low-resource languages.

These studies demonstrate the versatility and effectiveness of deep learning approaches for spelling correction across a range of low-resource languages. Despite variations in linguistic structures and model architectures, significant improvements in accuracy and performance metrics have consistently been achieved. Additionally, these studies highlight the promising potential of deep learning for low-resource languages. Section 2.4.2.4 examines the relevance and applicability of deep learning for a Maltese spell checker, highlighting its potential to exceed the capabilities of traditional approaches.

2.4.2.4 | Deep Learning: Relevance and Applicability for a Maltese Spell Checker

Deep learning offers substantial potential for Maltese spell checking by learning complex morphological patterns and context-sensitive errors without relying on manually crafted rules. Its adaptability makes it particularly suitable for low-resource and morphologically rich languages. However, it relies on significant computational resources and large annotated datasets, which can pose challenges for Maltese. Nevertheless,

Table 2.4: Deep learning results in low-resource languages. Source: Author.

Study	Language	Key Features	Model	Results
Gamo et al. (2024)	Wolaita	Compared BiLSTM with LSTM, GRU, RNN	BiLSTM	Accuracy: 98.12%
Ahmadzade and Malekzadeh (2021)	Azerbaijani	Synthetic training data, tested on real-world misspellings, edit-distance evaluation	LSTM encoder-decoder + attention	F1-score: 75% (dist. 0), 90% (1 edit), 96% (2 edits)
Sooraj et al. (2018)	Malayalam	Custom character-splitting; error location prediction; candidate ranking	Character-level LSTM for error detection	F1-score: 63%, Error detection accuracy: 55.2%
S. Singh and Singh (2021)	Hindi	Two test sets: synthetic split vs. manually curated	BiRNN + LSTM + attention	Synthetic split: F1-score: 78%, Accuracy: 80%; Manual split: F1-score: 76%, Accuracy: 74%
Kasmaiee et al. (2023)	Persian	Deep learning vs rule-based system	LSTM encoder–decoder	Accuracy: 87%, Precision: 70%, Recall: 89%, F1-score: 78%
Lubis et al. (2023)	Indonesian	Trained on social media; tested under various preprocessing scenarios	CNN	Accuracy: 89%

with carefully curated datasets, deep learning can deliver scalable and accurate solutions that outperform traditional and statistical methods, as demonstrated by the studies reviewed in Section 2.4.2.3.

While deep learning approaches have shown strong performance, recent advances in Transformer-based models offer even greater potential for NLP tasks such as spell checking. The following section (Section 2.4.3) presents an overview of Transformer-based architectures, examines their application to spell checkers, and explores their effectiveness in low-resource language settings.

2.4.3 | Transformer-Based Learning

The Transformer architecture, introduced by Vaswani et al. (2017), has revolutionised modern NLP, setting new benchmarks for performance across a range of tasks, including translation, summarisation, and spell checkers (de Araújo et al., 2024; Martynov et al., 2024). Prior to its development, deep learning models such as CNN, RNN, and variants like LSTM and GRU were considered state-of-the-art. However, the Transformer's introduction of self-attention and its scalability enabled a significant leap forward, setting a new standard in both language understanding and generation (Samsudin & Hassen, 2023; Wolf et al., 2020).

2.4.3.1 | Self-Attention Mechanism

At the core of the Transformer architecture is the self-attention mechanism, which enables the model to evaluate the importance of each token in a sequence relative to all others. Unlike recurrent models that process tokens sequentially, self-attention allows the Transformer to capture long-range dependencies in parallel, significantly improving computational efficiency and performance on long sequences. This mechanism is applied throughout both the encoder and decoder, forming the backbone of the architecture (Vaswani et al., 2017).

2.4.3.2 | Encoder-Decoder Structure

The Transformer follows an encoder-decoder design. The encoder consists of n identical layers, each made up of two components, multi-head self-attention and a position-wise feed-forward network. These are connected by residual connections and followed by layer normalisation, which together transform the input sequence into a continuous representation.

The decoder also consists of n identical layers and includes an additional masked multi-head self-attention layer. This masking ensures that, during training, the model cannot "look ahead" at future tokens. Instead, each prediction is based only on previously generated tokens, allowing the decoder to generate output step by step in an auto-regressive manner (Vaswani et al., 2017).

Figure 2.5 provides a visual representation of the original Transformer architecture introduced by Vaswani et al. (2017). It depicts the encoder-decoder structure, illustrating how self-attention mechanisms and feed-forward layers are organised within each block. This diagram serves as a useful reference for understanding the flow of data through the model during both encoding and decoding phases.

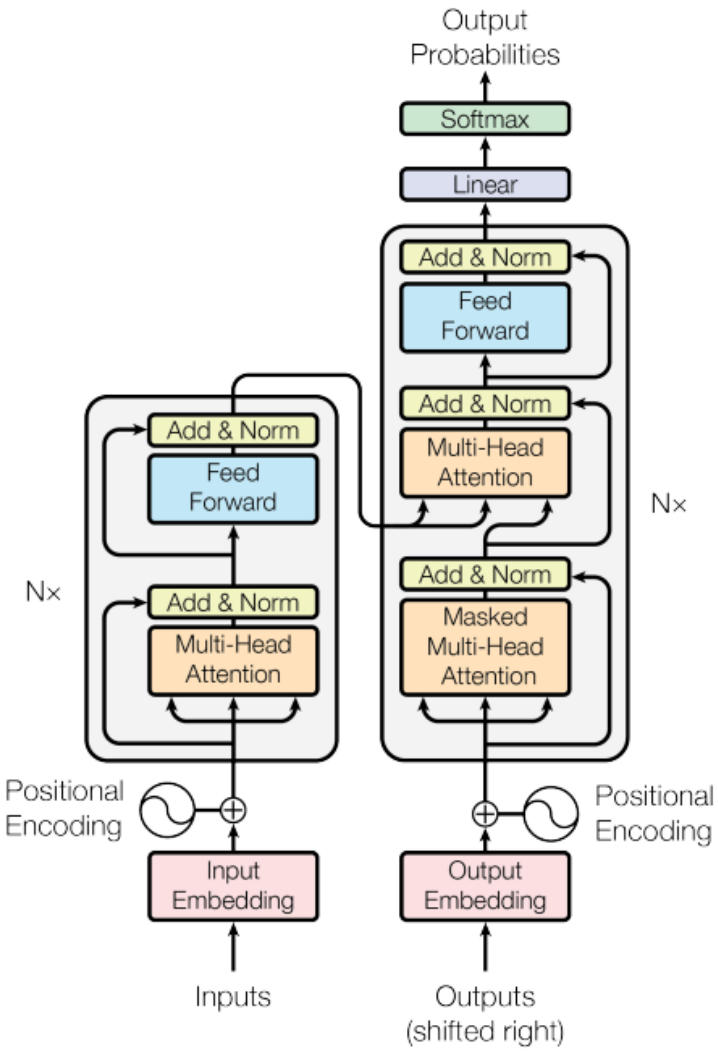


Figure 2.5: Transformer architecture. Source: Vaswani et al. (2017).

2.4.3.3 | Encoder-Only vs Decoder-Only Structures

Modern Transformer models often adopt specialised structures depending on their target tasks. Encoder-only models, such as Bidirectional Encoder Representations from Transformers (BERT), are designed to process and understand input sequences by generating rich contextual representations. These are typically used for classification, question answering, and named entity recognition (Devlin et al., 2019).

Decoder-only models, like Generative Pre-trained Transformers (GPTs), are optimised for language generation. They generate output sequences based on an initial prompt, using masked self-attention to maintain an auto-regressive behaviour (Radford & Narasimhan, 2018).

Although both structures are useful, recent advancements have largely focused on decoder-only architectures, especially in Large Language Models (LLMs) such as OpenAI's GPT-4[1], and Meta's Large Language Model Meta AI (LLaMA)[2], which have shown exceptional capabilities in generating human-like text (Suresh & P, 2024).

2.4.3.4 | Large Language Models

LLMs, which are built on the Transformer architecture, have significantly transformed the field of NLP. They are capable of performing complex tasks such as translation, summarisation, text generation, and spelling correction (de Araújo et al., 2024; Martynov et al., 2024). Notable examples of LLMs include OpenAI's GPT-4[1], Meta's LLaMA[2], and Google's Gemini[3]. These models outperform earlier approaches by effectively capturing relationships between words in large datasets, an area traditional techniques often fell short (Chang et al., 2023; Nicholas & Bhatia, 2023).

A key strength of LLMs lies in their ability to process large amounts of data simultaneously, due to their billions of parameters. This capacity for handling data in parallel overcomes the limitations of older models, enabling more efficient training and data analysis. However, these advantages come with significant trade-offs.

One of the main challenges faced by LLMs is their heavy reliance on large, high-quality datasets for effective training. While this is manageable for well-resourced languages, it becomes a more difficult task for low-resource languages, where such data may be scarce or even non-existent. Additionally, training and inference require substantial computational resources, resulting in high hardware demands and operational costs (Naveed et al., 2023).

To address the computational demands of LLMs, these models are typically deployed on specialised hardware such as Graphics Processing Units (GPUs) and Tensor Processing Units (TPUs), which are optimised for large-scale parallel processing. Such hardware advancements have been instrumental in supporting the growing complexity and scale of deep learning models (Telenti et al., 2024). In addition to hardware solutions, an increasingly important strategy for mitigating data and cost-related challenges involves model optimisation techniques (Xia et al., 2024).

[1] https://openai.com/index/gpt-4/
[2] https://www.llama.com/
[3] https://gemini.google.com/

Given the substantial size and resource requirements of LLMs, both training and inference can be time-consuming and costly. One widely adopted optimisation strategy is fine-tuning, which adapts a pre-trained model to smaller, task-specific datasets, improving efficiency (Xia et al., 2024). Beyond fine-tuning, Naveed et al. (2023) identifies several lightweight alternatives that reduce computational and operational demands. These methods include Parameter-Efficient Fine-Tuning (PEFT), pruning, quantisation, knowledge distillation, and context length interpolation. PEFT allows targeted model adaptation by updating only a subset of parameters, offering similar benefits to fine-tuning with far lower resource demands. Pruning removes redundant components to reduce model size and improve latency. Quantisation improves efficiency by lowering numerical precision. Knowledge distillation compresses a model by transferring knowledge from a larger model to a smaller one, thereby reducing its size and improving efficiency. Context length interpolation adjusts the amount of input context the model uses, balancing accuracy with computational cost.

These optimisation strategies improve the performance of LLMs while improving efficiency and cost-effectiveness, thereby enabling their practical use in real-world applications.

2.4.3.5 | Transformer-Based Learning in Low-Resource Languages

Transformer-based models have been explored for developing spell checkers in low-resource languages, as evidenced by numerous studies. For example, Phyu et al. (2024) created a Transformer-based spelling correction model for Myanmar, making use of a corpus derived from social media text. They categorised spelling mistakes into ten categories and incorporated these categories as error-type features into the model, leading to improved performance. The F1-score improved from 75.91% to 86.71%, the Grammar Language Evaluation Understudy (GLEU) score increased from 96.4% to 97.67%, and the word error rate dropped from 3.3% to 2.0%.

Pal and Mustafi (2020) developed a context-aware spelling correction system for Hindi, aimed at improving Optical Character Reader (OCR) accuracy, particularly for inflectional languages like Hindi. This approach combined the BERT Transformer model with the Levenshtein Edit Distance algorithm, a dictionary, and Named Entity Recognition (NER) to detect spelling errors based on context. The system achieved an accuracy of 81%, outperforming previous methods and showing promise for providing real-time auto-correct suggestions in text editors.

In a more recent study, Bijoy et al. (2025) employed a denoising Transformer for Bangla spelling correction, prioritising the correction of contextual spelling errors. To

address resource limitations, they proposed a method for generating large-scale corpora. The developed spell checker achieved a 94.78% exact match score, with precision, recall, and F1-scores around 0.948, surpassing prior approaches.

Keita et al. (2024) focused on grammatical error correction for the Zarma language, evaluating rule-based techniques, Machine Translation (MT) models, and LLMs. They found that the MT-based M2M100 model outperformed other approaches, achieving a 95.82% detection rate and 78.90% suggestion accuracy. On the other hand, the employed LLM, the MT5-small model, demonstrated strong performance, with a 90.62% detection rate and 57.15% suggestion accuracy.

Thapa et al. (2025) extended Transformer-based grammatical error correction to Nepali, a morphologically rich and low-resource language. Their system employs an encoder-decoder Transformer architecture to model contextual relationships and correct grammatical errors. The study compared this approach to RNN-based models, including stacked LSTMs and BiLSTMs with attention, and found that the Transformer significantly outperformed them. The model achieved 90.45% training accuracy, 92.15% validation accuracy, and strong Bilingual Evaluation Understudy (BLEU) scores of 0.9037, 0.8170, 0.7838, and 0.7694 for unigram to 4-gram matches, indicating high similarity between corrected and reference sentences and strong fluency in output.

Table 2.5 offers a comparative overview of results achieved by Transformer-based learning techniques applied to different low-resource languages discussed in this section.

Transformer-based models and LLMs represent the current state of the art in the field of NLP. Their application in developing spell checkers for low-resource languages has shown considerable promise, as demonstrated by the studies discussed above. The findings highlight the adaptability and effectiveness of the Transformer architecture in addressing this specific task. The following section (Section 2.4.3.6) discusses how a Maltese spell checker can benefit from Transformer-based learning.

Table 2.5: Transformer-based learning results in low-resource languages.
Source: Author.

Study	Language	Key Features	Model	Results
Pal and Mustafi (2020)	Hindi	Context-aware correction for OCR using BERT, dictionary, edit distance, and NER	BERT-based Transformer	Accuracy: 81%
Keita et al. (2024)	Zarma	MT and LLM comparison; rule-based vs. M2M100 and MT5-small	M2M100 (MT), MT5-small LLM	MT: Detect: 95.82%, Suggest: 78.90%; LLM: Detect: 90.62%, Suggest: 57.15%
Phyu et al. (2024)	Myanmar	Social media text; 10 error types as input features	Transformer with error-type features	F1-score: 86.71%, GLEU: 97.67%, Word error rate: 2.0%
Bijoy et al. (2025)	Bangla	Denoising model; synthetic corpus generation for low-resource settings	Transformer	Exact match: 94.78%, F1-score/Precision/Recall: $\approx$ 94.8%
Thapa et al. (2025)	Nepali	Compared with RNN-based models; BLEU score evaluation	Transformer encoder–decoder	Train accuracy: 90.45%, Validation accuravy: 92.15%, BLEU - unigram: 0.9037; BLEU - bigram: 0.8170; BLEU - trigram: 0.7838; BLEU - 4-gram: 0.7694

2.4.3.6 | Transformer-Based Learning: Relevance and Applicability for a Maltese Spell Checker

Transformer-based learning holds significant promise for the development of a Maltese spell checker. These models can be effectively adapted through fine-tuning on smaller, task-specific datasets. Despite their computational demands, their ability to capture long-range dependencies and contextual variation makes them well suited to the linguistic complexity of Maltese. When carefully applied, Transformer-based approaches offer a strong alternative to traditional, statistical, and earlier deep learning methods, with the potential to achieve superior performance in both error detection and correction.

2.4.4 | Conclusion

In conclusion, Section 2.4 has examined modern approaches to spell checker development, with a focus on statistical learning, deep learning, and Transformer-based learning.

These methods represent a significant advancement over the traditional techniques discussed in Section 2.3, offering improved accuracy, adaptability, and overall performance. The introduction of advanced models, particularly deep neural networks and Transformer architectures, has substantially improved the capacity of spell checkers to handle linguistic complexity and variation.

These advancements have not only transformed the development of spell checkers but also opened new opportunities for ongoing research and practical applications, especially in low-resource language contexts such as Maltese.

3

Literature Review

This chapter provides a critical review of prior work on spell checkers with particular attention to the Maltese language. It begins by examining Maltese spell checkers, including both academic research and practical systems such as spelling.mt, with a focus on their design decisions, evaluation strategies, and practical applications. It then situates these within a broader context by comparing methodologies developed for other low-resource languages, organised by approach, to illustrate how data scarcity is addressed across different settings. The discussion subsequently turns to datasets and evaluation practices, considering available corpora, annotation conventions, and error classification schemes. Attention is also given to emerging directions in the field, notably transformer-based models, to identify persisting research gaps and to motivate the deep learning framework employed in this dissertation. The chapter closes by drawing these strands together, positioning the dissertation's proposed approach as both relevant and timely.

3.1 | State of the Art in Maltese Spell Checking

The earliest reference to a Maltese spell checker study is Mangion (1999), who introduced MSpell, a rule-based system built around a morphological generator and a lexicon of root words. The design was motivated by Maltese's rich and non-concatenative morphology, which made dictionary-based coverage impractical. Although the system indicated that morphological analysis was essential, it also generated too many candidate forms and experienced performance issues. Evaluation was modest, focusing on examples rather than quantitative measures. Words were grouped into correct items recognised accurately, correct items wrongly flagged, incorrect items identified successfully, and incorrect items mistakenly accepted as valid. The last case was the most problematic, as it meant that misspellings could be classified as legitimate due to gaps in the lexicon or rule overgeneration. While this highlighted useful weaknesses in the design,

the absence of metrics such as precision or recall made it difficult to judge performance precisely, and there was no evidence of adoption beyond academic use.

A year later, Mizzi (2000) developed a statistical spell checker based on n-gram models and probabilistic inference, marking a move away from strictly rule-driven methods. Evaluation was structured in two parts: spelling verification, which tested whether words were valid using smoothed n-gram probabilities, and context-sensitive correction, which examined whether real-word errors could be resolved using similarity sets, context words, and collocations. Tests were performed on both the Maltese and English corpora, and the results showed broadly comparable results across the two languages, suggesting that statistical techniques can be applied effectively even in a morphologically rich language. The evaluation, however, relied mainly on frequency counts and illustrative examples rather than reproducible metrics such as precision or recall. While the work usefully highlighted the potential of statistical methods for Maltese, the absence of large-scale quantitative testing limited the ability to rigorously judge accuracy, and the system remained a research prototype with no evidence of real-world deployment.

The first widely available public spell checker for Maltese appeared in the form of spelling.mt[1] developed by Casha (2004). This tool was built on Unix Aspell and generated correction candidates through dictionary lookup and edit-distance matching. Its public release marked a significant milestone, but the system also showed apparent limitations. As Debattista (2022) notes, it ignored phonetic variation and grammar, which reduced its usefulness for a morphologically rich language. An empirical evaluation by Buhagiar (2021) reported a hit rate of only 48.8%, meaning that in more than half of cases the intended correction was not ranked first. This reflects a frequent tendency to produce irrelevant or implausible suggestions, undermining effectiveness in real-world contexts. Despite these weaknesses, spelling.mt[1] remains the only publicly available spell checker for Maltese, highlighting both the demand for such tools and the gap that persists between academic prototypes and practical applications.

In recent years, Buhagiar (2021) investigated techniques for spelling and grammar checking in Maltese, motivated by the observation that existing tools, such as spelling.mt[1] failed to provide reliable suggestions for many common errors. The system of Buhagiar (2021) was implemented as a web-based application with a Python back-end, combining the Symmetric Delete algorithm with a novel Maltese Soundex method. This design was chosen to address the shortcomings of spelling.mt, which relied solely on edit distance and therefore ignored pronunciation. Evaluation was carried out through di-

[1] `https://spelling.mt/`

rect comparison with spelling.mt[1], using a set of test spelling mistakes. The results showed that the system of Buhagiar (2021) achieved a hit rate of 80.6%, compared to 48.8% for spelling.mt[1], indicating a clear improvement in the quality of candidate ranking. Buhagiar (2021) also reported that the tool was limited by execution speed and by the small sample of grammar rules implemented. While the dissertation demonstrated that phonetic-aware approaches could significantly outperform the existing public tool, it remained a research prototype with no evidence of sustained real-world use.

Debattista (2022) explored Grammatical Error Correction (GEC) for Maltese using Neural Machine Translation (NMT) approaches. The study implemented sequence-to-sequence (Seq2Seq) and Transformer-based architectures, with improvements including tied embeddings, source word corruption, and transfer learning. Evaluation followed established practice, using the Error Annotation Toolkit (ERRANT) scorer to measure precision, recall, and $F_{0.5}$. The best system achieved an $F_{0.5}$ of 31.84%, a result comparable to other low-resource submissions in the BEA-2019 shared task. In addition to quantitative metrics, the dissertation provided qualitative analysis of errors, noting challenges such as False Negatives (FNs), invalid corrections, and domain bias. This study marked a methodological shift from earlier traditional approaches and established a more comprehensive evaluation framework for Maltese error correction, offering a valuable reference point for future research.

The most recent contribution to Maltese spell checking is by Oliveri (2025), who examined GEC using a Transformer-based Seq2Seq model. The study addressed data scarcity by generating large-scale synthetic errors at the token and character levels through adversarial operations and from Automatic Speech Recognition (ASR) output, while also incorporating smaller authentic resources, such as Qari tal-Provi, transcription datasets, and Wikipedia edits. Training involved pre-training on synthetic corpora followed by fine-tuning with authentic data in different proportions. Evaluation employed ERRANT (precision, recall, $F_{0.5}$) and Bilingual Evaluation Understudy (BLEU), with the best configuration reaching an $F_{0.5}$ score of 33.40 and an error detection score of 64.2, representing incremental gains over earlier Maltese systems using neural models. Analysis showed particular success in correcting orthographic problems such as diacritics and anglicisation, though persistent weaknesses included over-correction, under-correction, and misclassifications, while ASR-derived errors added little value. Overall, the study demonstrated the usefulness of synthetic data for low-resource settings but also highlighted its limitations, as the artificially generated errors did not always mirror real-world use.

The work of Oliveri (2025) also references an unpublished study by Busuttil (2025), which developed a mixed dataset consisting of synthetically generated errors and a

smaller authentic dataset collected using a custom transcription tool. This tool was configured to capture genuine human errors by turning off auto-correct and backspace functions while participants transcribed spoken Maltese sentences from the Common Voice dataset, thereby producing realistic error-correction pairs. The system employed an adapter-based encoder-decoder architecture built upon Maltese-specific Bidirectional Encoder Representations from Transformers (BERT) models. It achieved an $F_{0.5}$ score of 34.66, marking the highest reported performance for Maltese at the time.

Taken together, these projects trace a clear development path: from morphology-based prototypes, to statistical models, to the first publicly available dictionary-based tool (spelling.mt[1]), followed by phonetic-statistical hybrids, and most recently neural approaches. Over time, evaluation practices progressed from illustrative examples to controlled comparisons and internationally recognised scoring methods. This trajectory highlights the academic value of these studies in advancing methodological understanding, while also underlining the continuing gap between research and usable tools for Maltese. Persistent challenges with limited resources, over-generation, low hit rates, and lack of integration have so far prevented any system from achieving sustained adoption.

3.2 | Comparative Insights from Other Low-Resource Languages

The lack of large annotated corpora can hinder spell checking and grammar correction in low-resource languages. As outlined in Chapter 2, various approaches have been developed over time, each with distinct data requirements and challenges. Rule-based systems rely on handcrafted linguistic rules, while statistical methods draw on smaller corpora and probabilistic models, and neural approaches employ machine learning techniques to learn correction patterns from larger datasets. This section examines how each methodological family addresses data sparsity and highlights points of divergence that may inform the methodology adopted in this dissertation.

3.2.1 | Rule-based Technique

Rule-based spell checkers use manually handcrafted linguistic rules tailored to the specific characteristics of a language to identify and correct errors. Instead of learning from large datasets, they rely on a priori knowledge of the grammar patterns and spelling patterns of the language (Hládek et al., 2020).

Ahmadi (2021) developed a rule-based morphological analyser and spell checker for Sorani Kurdish using Hunspell. To address the scarcity of annotated resources, the study developed a lexicon comprising 23,223 entries, each tagged with grammatical information, and defined 4,293 rules to handle affixes, clitics, and sound changes. These rules allowed the system to generate and analyse many inflected and derived word forms, which reduced the need for an exhaustive dictionary listing every possible word.

Similarly, Gamu and Woldeyohannis (2023) introduced a morphology-based spell checker for Dawurootsuwa, a low-resource language of Ethiopia. The system combined a lexicon of approximately 5,000 root words with more than 2,500 morphological rules, thereby covering derivational, inflectional, and compound forms. Data sparsity was addressed through morphological generalisation, enabling recognition of unseen word forms without the need for an exhaustive dictionary. The drawback, however, is the intensive manual effort required, as constructing and maintaining such rule sets demands specialised linguistic expertise. While the method is not directly transferable to all languages, it offers a promising model for morphologically rich languages, which can reduce reliance on large lexicons by exploiting similar principles.

On the other hand, Olayiwola et al. (2023) took a different approach by implementing a grammar checker for Yorùbá using Government and Binding theory, a generative framework that models sentence structure through hierarchical phrase rules and constraints on how elements, such as nouns and pronouns, relate. Their system encoded deep syntactic rules to handle simple, complex, and compound sentences. This approach addresses data sparsity by relying on formal syntactic modelling rather than data-driven learning, allowing for robust grammar checking without the need for annotated corpora. However, as the authors note, FPs and False Negatives (FNs) emerged due to the difficulty of encoding every syntactic construction in Yorùbá.

Together, these studies demonstrate that rule-based systems can cope with limited data in different ways, such as through morphological or syntactic rules. The key takeaway is that while each approach has its strengths, they also face notable constraints. Rules can reduce the need for large dictionaries, but extensive dictionaries are fragile when faced with dialectal variation. Morphological rules improve coverage, but require expert knowledge to design and maintain. Sentence-level rules capture syntax effectively, but cannot easily account for every possible construction. A more promising direction is to combine the accuracy of linguistic rules with flexible methods that can adapt to new situations, creating systems that are both precise and robust in low-resource settings.

3.2.2 | Edit Distance Algorithms

Edit Distance algorithms provide an alternative approach that is less dependent on annotated data. By modelling insertions, deletions, substitutions, or transpositions, they address data sparsity by generating candidate corrections without large training resources (Chaabi & Ataa Allah, 2022). A key limitation, however, is reliance on a comprehensive dictionary for adequate coverage (Patil et al., 2021). This section reviews how such methods have been applied to spell checking in different low-resource languages.

Khairul Islam et al. (2019) applied the Levenshtein algorithm to build a spell checker for Bangla, while Patil et al. (2021) used Minimum Edit Distance (MED) for Marathi. In both cases, annotated corpora were replaced with dictionary coverage, utilising dictionaries of approximately 4,000,000 entries for Bangla and 900,000 for Marathi. While this avoided the need for error-correction datasets, it introduced a significant resource demand, namely reliance on extensive dictionaries that not all low-resource languages can adopt. The discrepancy in dictionary sizes is evident between the mentioned studies because Bangla employed more than four times the number of entries used for Marathi. Part of this difference reflects the availability of digital resources, but it also stems from natural variation in the vocabulary size of the two languages. For this approach to be effective, the dictionary must cover as much of the language as possible, since Edit Distance algorithms can only suggest words that are present in the lexicon. Therefore, coverage is critical. Moreover, when several candidates share the same edit distance, the absence of contextual or frequency-based signals hinders reliable prioritisation. Hence, although Edit Distance systems appear corpus-light, its effectiveness still depends on substantial lexical resources.

Santoso et al. (2019) compared the Levenshtein with Damerau-Levenshtein to handle spelling errors in Indonesian. Their approach to address the lack of data was to model common mistakes, especially transposition errors. This method enabled the capture of typical typing errors without relying on extensive collections of annotated examples.

Cissé and Sadat (2023) built a Wolof spell checker using Weighted Edit Distance with trie-based search. To address sparsity, they exploited the orthography of Wolof. Substitutions between accented and unaccented characters were assigned lower costs, thereby reducing the need for a large dictionary and allowing for efficient matching within a compact lexicon of just 1,410 entries. This approach made the system more feasible for a low-resourced language. However, this came with trade-offs, including higher computational costs and persistent FPs, since the algorithm does not account for real-world errors.

Mukazhanov et al. (2023) tackled data sparsity in Kazakh by extending the Damerau-Levenshtein algorithm. They added rules for common, language-specific mistakes. These mistakes included confusing Cyrillic and Kazakh letters or typing digits instead of letters with diacritics. By incorporating these error patterns into the algorithm, the system could correct common errors without needing large annotated datasets. This method made the spell checker effective despite limited resources, but its accuracy depended on predefined rules, and it struggled with error types not covered by them.

The aforementioned studies demonstrate that Edit Distance systems address data sparsity in various ways. Some rely on comprehensive dictionaries, while others employ minor algorithmic tweaks, orthographic shortcuts, or language-specific error rules. Each of these strategies reduces the need for annotated corpora, but each also comes with its own drawbacks. Large dictionaries cover more words, but are not always feasible. Algorithmic tweaks can catch common typos, yet their impact is limited. Orthographic weighting enables small dictionaries to go further, but it can also increase FPs. Handcrafted rules handle predictable errors well, though they do not transfer easily to other languages. Taken together, these differences suggest that no single approach is enough. A more promising path is to combine them by using Edit Distance as a starting point for generating candidates, and then adding lightweight rules or contextual ranking to improve accuracy and adaptability in low-resource settings.

3.2.3 | N-grams

Ambili et al. (2016) developed a spell checker for Malayalam that combined dictionary look-up with character n-gram similarity. To manage the limited resources, the system relied on a small, handcrafted dictionary of approximately 10,000 words rather than large annotated datasets or extensive text collections. This kept the method lightweight and suitable for very low-resourced scenarios. At the same time, the limited dictionary reduced coverage, and the system was primarily designed to address single-character errors, leaving more complex mistakes and morphological variations unaddressed. The study shows that dictionary-plus-n-gram methods can work with very little data, but their capacity to scale is limited.

Sakuntharaj and Mahesan (2018) and Sakuntharaj and Mahesan (2021) developed Tamil spell checkers that tackled both real-word errors and missing-word errors. In both studies, the minimum edit distance was used to generate candidate corrections. In addition, unigram, bigram, and trigram language models were evaluated to determine which words best fit the surrounding context. Crucially, instead of annotated error data, the models were trained on a large, unlabelled Tamil corpus of approximately 3 Giga-

bytes (GBs), covering 1.78 million word types collected from online sources. This approach allowed the systems to capture contextual patterns directly from raw text. However, while effective for Tamil, the reliance on such a large corpus raises concerns about feasibility in other low-resource languages, where such digital resources are unlikely to exist.

Similarly, in a more recent study, Oluwaseyi et al. (2024) presented a Yorùbá spell checker that combined edit distance for candidate generation with unigram, bigram, and trigram language models for ranking. To cope with limited resources, the authors scraped Yorùbá text from the web using Sketch Engine. The scraped data was simplified by removing diacritics, a process referred to as asciification, which made the corpus easier to process but discarded tone information crucial to the language. The n-gram models were then trained on this corpus by calculating the frequency of word occurrences and word sequences. These probabilities were used to rank correction candidates, with those that best fit the surrounding context given higher priority. This approach avoided the need for annotated error–correction data, but its reliance on a web-scraped corpus and the removal of diacritics limited its linguistic accuracy. Moreover, the system addressed only non-word errors, leaving real-word mistakes unresolved.

These studies highlight different approaches to addressing data sparsity in n-gram and hybrid systems. The Malayalam study demonstrates that a small dictionary can support a lightweight solution, though this comes at the cost of limited coverage. The Tamil studies take the opposite approach, drawing on a large unlabelled corpus to add contextual information, which works well but is unrealistic for most low-resource languages. The Yorùbá study offers a compromise, where web scraping created a usable corpus, but the removal of diacritics weakened linguistic accuracy. For Maltese, a potential direction is a middle ground where candidate corrections are generated with Edit Distance, the dictionary is expanded with realistically obtainable unlabelled text, and n-gram models that preserve diacritics help select the most appropriate corrections while maintaining spelling accuracy.

3.2.4 | Statistical Learning

This section examines statistical spell checkers developed for a range of low-resource languages, with a focus on the strategies employed to manage data sparsity. Although all of the systems are statistical in nature, they differ in how they confront the challenge of limited data across three critical components of the spell checking pipeline: the lexicon, which determines the set of valid words; the error model, which estimates the likelihood of particular misspellings in the absence of large annotated corpora; and

the context model, which evaluates how well candidate corrections fit within surrounding text. Framing the discussion around these three components highlights the distinct forms of sparsity that each presents, and clarifies the diverse strategies adopted across languages to overcome them.

3.2.4.1 | Lexical Coverage

In terms of lexical coverage, Gezmu et al. (2018) demonstrates for Amharic that high-frequency term lists derived from a carefully curated corpus, specifically the CACO corpus compiled in the study and containing approximately nineteen million words, outperform dictionaries based on noisier web sources, resulting in improvements in both error detection and correction. The authors further mitigate noise in their language models by excluding words that occur only once. For Indonesian, Soleh and Purwarianti (2011) reduce sparsity through morphological normalisation, using an analyser to strip words to their root forms before candidate generation, thereby allowing a smaller dictionary to remain effective in a morphologically complex language. Kamayani et al. (2011), by contrast, intentionally restrict their dictionary to about thirteen thousand entries and rely on forward-reverse dictionaries combined with Hidden Markov Model (HMM) ranking to extend coverage. However, they acknowledge that missing entries continue to cause errors. A different strategy is employed by Hladek et al. (2013) for Slovak, who utilise an extensive, manually verified dictionary of six and a half million words, along with n-grams trained on fiction. However, they note that domain mismatches between the training and test data worsen out-of-vocabulary issues.

Collectively, these approaches suggest that lexical sparsity can be alleviated either by curating clean corpora, applying morphological analysers, or investing in large-scale dictionary development, each with distinct trade-offs in resources and accuracy.

3.2.4.2 | Error Modelling

Sheykholeslam et al. (2012) addressed Persian by creating a large native error corpus through double-keying more than 2,000 books, yielding around three million error-correction pairs. This approach enabled the construction of detailed confusion matrices, producing the strongest results but at considerable resource cost. Hladek et al. (2013) for Slovak, adopted a heuristic observation model, using the rank of candidates suggested by an external spell-checker as an estimate of error probability. Gezmu et al. (2018) faced the absence of an Amharic error corpus and therefore transliterated text into Latin script, importing an English error model (Norvig's substring-based statistics) to approximate

typing mistakes across scripts. More recently,Luitel et al. (2024) proposed an unsupervised solution for Nepali, assuming that correct words occur more frequently than close misspellings and mining raw corpora under this assumption to infer error likelihoods without annotation.

These studies highlight four viable responses to error sparsity: direct large-scale data creation, heuristic approximation, cross-lingual transfer, and unsupervised inference.

3.2.4.3 | Contextual Modelling

Contextual modelling was featured in several statistical systems, helping to determine which candidate correction was best suited for the surrounding text, even when limited data was available. Gezmu et al. (2018) demonstrated that, in the case of Amharic, trigram language models can still be effective under low-resource conditions if carefully managed. The use of modified Kneser-Ney smoothing reduced the impact of missing word sequences, while excluding words that occurred only once in the corpus, limited noise that might otherwise distort probability estimates. Soleh and Purwarianti (2011) for Indonesian and Hladek et al. (2013) for Slovak both relied on Hidden Markov Models (HMMs) with Viterbi decoding to ensure sentences remained grammatically consistent. This approach functioned with relatively small corpora, although both studies stressed that the accuracy of such sequence models was strongly tied to the quality of bigram counts and the effectiveness of smoothing, which are especially vulnerable to sparsity. Additionally, heuristics also contributed to sparsity management by constraining the candidate pool, including morphological reduction in Indonesian. Collectively, these studies demonstrate that careful smoothing and sequence-based decoding represent practical strategies for coping with sparse contextual data in low-resource languages.

In summary, these studies reveal divergent strategies for coping with data sparsity. Some emphasise compact, morphology-aware dictionaries, while others depend on larger curated corpora. Error models similarly vary, ranging from costly corpus creation to cross-lingual transfer or unsupervised inference. For contextual modelling, approaches diverge between smoothed n-grams and sequence decoding. Taken together, these contrasts suggest that, for Maltese, a balanced approach may be most effective, combining a realistically obtainable lexicon, lightweight error modelling, and robust contextual ranking without the need for substantial resources.

3.2.5 | Deep Learning

Developing spell checkers for low-resource languages with deep learning continues to face many of the same challenges as traditional techniques, particularly the shortage of sufficient and reliable data. To address this, researchers have explored strategies that can be grouped into two distinct categories: representation and pre-processing, and error data and augmentation. Each of these will be examined in the following subsections.

3.2.5.1 | Representation and Pre-processing

Low-resource corpora provide incomplete coverage of the vocabulary because numerous words could be missing entirely, and those present are represented by too few examples to support reliable learning. Deep learning systems address this lexical sparsity by improving word representation and how data is prepared before training. Character and sub-word level modelling allows systems to construct representations of words that do not appear as complete tokens in the training data, making them more robust to rare or unseen forms. In addition, pre-processing techniques reduce the noise and variability that dilute small corpora, allowing the limited data available to be used more effectively. Sooraj et al. (2018) reported that a character-based Long Short-Term Memory (LSTM) for Malayalam enabled the system to handle unseen word forms without relying on an extensive dictionary. S. Singh and Singh (2021), working on Hindi, used word2vec embeddings (Mikolov et al., 2013) with a Bidirectional Recurrent Neural Network (BiRNN) and attention. This representation compressed Hindi's significant character set into dense vectors and reduced sparsity caused by rare word forms. Gamo et al. (2024) trained Recurrent Neural Network (RNN), LSTM, Gated Recurrent Unit (GRU), and Bidirectional Long Short-Term Memory (BiLSTM) models on a Wolaita corpus of 23,000 sentences, applying tokenisation, normalisation, and stemming to maximise the value of the limited data. For Indonesian, Lubis et al. (2023) showed that pre-processing noisy social media text through case-folding, stop-word removal, and stemming substantially improved Convolutional Neural Network (CNN) performance, with reported accuracy rising from 0.70 to 0.89. Taken together, these studies suggest that deep learning systems in low-resource settings address lexical sparsity not by expanding dictionaries but through careful representation design and rigorous pre-processing.

3.2.5.2 | Error Data and Augmentation

A second challenge is the scarcity of error corpora for training models to recognise and correct mistakes. Constructing extensive collections of authentic spelling errors is costly

and rarely feasible for low-resource languages, so researchers often resort to augmentation techniques that artificially expand training data. Sooraj et al. (2018) addressed this problem for Malayalam by deliberately introducing spelling errors, such as character insertions, deletions, substitutions, and transpositions, into clean training text. This synthetic noise enabled their LSTM model to learn correction patterns without a dedicated error dataset. Ahmadzade and Malekzadeh (2021), working on Azerbaijani, applied a similar strategy by generating artificial errors through insertion, deletion, substitution, and transposition, thereby enlarging their training corpus. For Persian, Kasmaiee et al. (2023) created a synthetic corpus of 1.2 million sentences using 112 handcrafted rules that simulated common typographical and phonetic errors. Together, these studies demonstrate that in low-resource settings, data augmentation is a practical approach to addressing data sparsity.

In summary, these deep learning studies reveal various strategies for addressing data sparsity. Some systems emphasise character or sub-word level modelling to bypass the limits of small dictionaries. In contrast, others rely on large-scale artificial error generation to compensate for the absence of annotated corpora. These divergences suggest that, for Maltese, a promising direction would be to combine character or sub-word level representations with realistic data augmentation. Such a design would respect the constraints of a low-resource setting while still effectively leveraging contextual information and error patterns.

3.2.6 | Transformer-Based Learning

Transformer-based models have recently been explored for developing spell checkers, including those for low-resource languages. Their strong performance in high-resource contexts makes them appealing. However, their dependence on large annotated datasets creates a serious challenge when such resources are scarce. This section examines how recent studies have addressed data scarcity in the context of building transformer-based spelling and grammar correction systems.

Following many deep learning approaches, researchers developing transformer-based spell checkers for low-resource languages have sought to overcome the lack of annotated data by generating synthetic corpora. Bijoy et al. (2025), Keita et al. (2024), and Thapa et al. (2025) all follow this path, though they employ different techniques.

The study of Bijoy et al. (2025) for Bangla is the most ambitious, as the authors injected fourteen error types into clean sentences and produced over 1.3 million error-correction pairs, enough to lift Bangla out of its low-resource status for spelling correction. Similarly, Keita et al. (2024) applied the same idea for Zarma, using a noise

script to introduce deletions, insertions, substitutions, and transpositions. Each sentence yielded four corrupted variants, resulting in more than 250,000 examples; however, the authors note that synthetic noise may not accurately replicate more complex logical errors. Thapa et al. (2025) employed a comparable strategy for Nepali, defining five corruption operations, including word removal, verb replacement, and random substitution, to construct a dataset of 1.58 million sentences. While this offered substantial scale, the data consisted mainly of political, social, and business texts, raising questions about how generalised the data is.

In contrast, Phyu et al. (2024) addressed data scarcity in Myanmar by constructing their own dataset from social media. They manually annotated 63,036 error-correction pairs and categorised each mistake into one of ten types, such as phonetic, typographic, or slang.

The reviewed studies show that data sparsity can be tackled through different strategies. Augmenting data by injecting errors can quickly produce large corpora, but it may overlook the patterns of real mistakes, whereas manually collected errors provide authenticity but are expensive and limited in quantity. For a Maltese spell checker, the most practical approach is to create synthetic errors on top of carefully prepared gold-standard data, ensuring realism by modelling the kinds of mistakes that actually occur in practice.

3.3 | Datasets and Evaluation Frameworks

The success of spell checkers and grammar correction systems depends not only on the algorithms they use but also on the resources and evaluation methods that support them. For low-resource languages, the choice of corpus, annotation practices, error categorisation, and evaluation framework is especially significant because these elements determine both the feasibility of development and the fairness of performance comparisons. This section surveys how these elements have been addressed across studies, highlighting the lessons that inform the approach taken in this dissertation.

3.3.1 | Corpora

A corpus is a collection of written texts, often comprising the complete works of a particular author or writings related to a specific subject area. In contrast, a lexicon is the vocabulary of a language, consisting of its words and units of meaning. Typically, a lexicon is constructed from an extensive corpus (Hossain et al., 2021).

One of the latest studies on Maltese, a GEC tool developed by Debattista (2022), highlighted three key corpora as essential resources: Qari tal-Provi, Mozilla's Common Voice[2], and the Maltese Language Resource Server (MLRS) Korpus Malti[3]. The Qari tal-Provi corpus originates from the Maltese Diploma in Qari tal-Provi (Proofreading) and is distinctive because it contains parallel examples of incorrect and corrected texts that were initially developed as teaching materials for students. Two main batches of material were used by Debattista (2022). The first batch was distributed as Microsoft Word documents in which errors and corrections were tracked in Extensible Markup Languages (XMLs) markup. The second batch was released as sets of PDF exercises containing original erroneous texts, highlighted versions, and corrected versions. Together, these resources provide roughly 4,000 pairs of aligned erroneous and corrected sentences, making Qari tal-Provi the only curated Maltese corpus explicitly focused on error-correction material.

Mozilla's Common Voice corpus, although designed primarily as a speech dataset, provides over 5,000 Maltese sentences aligned with audio recordings and transcriptions, making it a practical textual corpus as well.

The most comprehensive resource is the MLRS Korpus Malti[3], which was recently expanded by Micallef et al. (2022). This large-scale, general-purpose, monolingual corpus covers 19 domains, including newspapers, parliamentary debates, European Union (EU) texts, blogs, literature, and Wikipedia[4], and contains more than 466 million tokens. Unlike opportunistic web-scraped collections, the MLRS Korpus Malti[3] applies careful filtering of non-Maltese material and eliminates duplicates, resulting in a cleaner and more reliable resource.

More recently, Oliveri (2025) highlighted the Maltese Wikipedia revision history as an additional corpus beyond those identified by Debattista (2022). This source provides around 8,000 correction pairs, automatically extracted as grammatical edits, and is therefore regarded as a silver standard dataset.

Taken together, these corpora may serve as a useful starting point for the development of Natural Language Processing (NLP) systems in Maltese, as they offer raw material that could support applications such as spell checking tools.

[2] https://commonvoice.mozilla.org/en
[3] https://mlrs.research.um.edu.mt/
[4] https://www.wikipedia.org/

3.3.2 | Annotation Standards

Since spell checkers are trained and tested on annotated data, the quality of the annotation standards directly shapes their accuracy and reliability. A gold-standard dataset is a collection of texts that has been carefully annotated by professionals following agreed guidelines, making it the most dependable reference against which systems can be developed and evaluated. In the context of spelling and grammar correction, such datasets include both the original text with errors and the corrected versions, sometimes enriched with labels describing the type of error. What makes a dataset truly "gold" is not only the data itself, but the annotation standards behind it. Annotation standards are the guidelines that instruct annotators on how to correct errors and maintain consistency across their work. When applied systematically, these standards make datasets more robust and valuable for both training and evaluation.

Stymne et al. (2017) demonstrates the value of detailed rules for marking errors. In their study, mistakes were categorised into types such as spelling errors, missing spaces, or incorrect word forms. Both the original and the corrected versions were preserved to ensure transparency. To maintain consistency, they established clear guidelines and checked that different annotators produced uniform results. This approach reflects a traditional gold standard encompassing precise categories, consistent corrections, and reliability, all of which support the development of more accurate tools.

Building on this, Napoles et al. (2017) take a broader perspective. Instead of focusing only on fixing mistakes, annotators were asked to make sentences sound fluent and natural, closer to native usage. Each sentence was rewritten by four different people, showing that there are often multiple valid corrections. Although this corpus extends beyond spelling, it illustrates how annotation standards can prioritise not just correctness but also fluency, providing richer training data for systems that aim to generate more natural text.

In low-resource contexts, annotation standards differ depending on whether datasets contain authentic errors or synthetically generated ones. Authentic annotation, as in the Myanmar error-correction dataset, involves professionals correcting real user errors and assigning them to explicit categories such as typographic, phonetic, or slang (Phyu et al., 2024). This approach creates a dependable gold standard, but it requires substantial effort and agreement protocols to maintain consistency. Synthetic annotation, by contrast, relies on predefined rules to label errors automatically. In Azerbaijani, four edit operations were applied to generate error-correction pairs, some of which reflect common keyboard confusions (Ahmadzade & Malekzadeh, 2021). For Persian, handcrafted rules were used to tag errors as typographic or phonetic and to align them with corrected ver-

sions (Kasmaiee et al., 2023). Such practices produce large annotated datasets quickly, but they lack the reliability and linguistic sensitivity expected of a gold standard.

Together, these studies show that annotation standards determine what a dataset can offer. For spelling correction-focused tasks, defining error categories and methods of correction matters, but the consistency of corrections and the careful construction of the corpus are decisive. For systems that also aim at natural fluency, multiple human rewrites provide a more flexible benchmark. These choices affect not only how systems are evaluated, but also how effectively they learn. For spell checkers in particular, clear and consistent annotation standards produce stronger training data and, ultimately, more reliable tools.

3.3.3 | Evaluation Frameworks

The evaluation of spell checkers has evolved in parallel with the systems themselves, and the choice of metrics reflects different perspectives on the task. Early dictionary-driven and rule-based systems were typically assessed primarily in terms of correction accuracy, that is, the proportion of spelling errors for which the correct form was proposed. In research contexts, these approaches are often viewed as classification problems, motivating the use of metrics such as precision, recall, and F1-score to quantify error detection and correction (Hládek et al., 2020; Näther, 2020). Such token-level metrics remain particularly useful for non-learning systems and for morphology-aware spell checkers developed for low-resource languages (Ambili et al., 2016; Sakuntharaj & Mahesan, 2018).

Hládek et al. (2020) also point out that spelling correction can be evaluated through an information retrieval lens, where an incorrect token is treated as a query and the ranked list of candidate corrections is treated as the response. This approach enables the use of Mean Reciprocal Rank (MRR) and Mean Average Precision (MAP), which measure the position of correct suggestions within ranked lists of candidates. These metrics are particularly informative when assessing the quality of suggestion lists.

With the emergence of neural and sequence-to-sequence (Seq2Seq) models, spelling correction has increasingly been viewed as a form of translation, leading to the adoption of metrics from Machine Translation (MT). Hládek et al. (2020) report that Bilingual Evaluation Understudy (BLEU) and Grammar Language Evaluation Understudy (GLEU) are commonly used to evaluate corrected text against gold-standard references at the sentence level. Building on this, later low-resource studies employ these same metrics in practice, often in neural settings. For example, Keita et al. (2024) evaluates Zarma and Bambara systems built with transformer-based models, applying BLEU,

GLEU, and the M^2 scorer for edit-level analysis. Similarly, Phyu et al. (2024) assesses Myanmar spell correction using a Seq2Seq architecture with attention, employing BLEU and GLEU for sentence-level evaluation. Additionally, this study employs the Word Error Rate (WER) metric. This metric is intended for speech recognition evaluation, but for this study, it was used to capture substitutions, insertions, and deletions.

More recently, evaluation frameworks for grammar and spelling correction have shifted their focus to F0.5 and GLEU, which are considered particularly well-suited to large neural models, such as Seq2Seq models with attention, transformer-based models, and Large Language Models (LLMs). F0.5, which has become the standard in grammatical error correction shared tasks, gives twice as much weight to precision as to recall. This means it rewards systems that make fewer but more reliable corrections, rather than ones that try to change too much. GLEU, on the other hand, measures how fluent and natural the corrected text is by comparing it with gold-standard references, without relying on a fixed set of error categories (Arnardóttir et al., 2024).

Taken together, these metrics illustrate a trajectory from token-level accuracy and classification measures in early systems to ranking-based evaluation for suggestion quality, and finally to translation-style and fluency-based metrics for modern deep learning models and LLMs. For the development of a Maltese spell checker, this review highlights the importance of selecting metrics that align with both the system's architecture and its intended use. A dictionary-based and rule-based system can be effectively assessed in terms of precision, recall, and accuracy. On the other hand, more advanced neural models for Maltese would benefit from sentence-level metrics such as BLEU, GLEU, or F0.5 to capture fluency and correction quality.

3.4 | Emerging Trends and Research Gaps

In recent years, research on spelling and grammatical error correction has undergone significant changes, shifting away from the rule-based and statistical approaches towards deep learning and transformer-based models, which have become the benchmark for state-of-the-art performance (Klemen et al., 2024).

Building on this shift, researchers have increasingly applied deep learning methods such as LSTMs, BiRNNs, and CNNs to spell checkers for low-resource languages, as demonstrated in the works of (Gamo et al., 2024; Kasmaiee et al., 2023; Lubis et al., 2023; S. Singh & Singh, 2021; Sooraj et al., 2018). More recently, the field has advanced further towards Transformer-based models, with studies such as (Bijoy et al., 2025; Keita et al., 2024; Phyu et al., 2024; Thapa et al., 2025) showing that pre-trained and fine-

tuned Transformers consistently outperform earlier neural approaches even under data scarcity. According to Bryant et al. (2023), because Transformers are very large models that require vast amounts of training data, researchers now routinely pre-train them on related tasks or synthetic error data before fine-tuning them for error correction. This strategy has become standard practice. More recently, large pre-trained language models (LLMs) have been shown to achieve state-of-the-art results when fine-tuned on curated data, reducing the need to create vast synthetic corpora. Although these models are demanding in terms of computational resources and can act as "black boxes," their ability to learn end-to-end from data and capture subtle contextual cues makes them the most effective approach available. The demonstrated success of pre-trained and fine-tuned transformer models in systems such as GEC suggests that extending transformer-based architectures to spell checking in low-resource languages could offer significant advances over traditional methods.

Although Maltese research has adopted Transformer-based methods for GEC (Debattista, 2022; Oliveri, 2025), performance remains constrained by the limited availability of authentic corpora, the predominance of synthetic training data, and the reliance on standard Transformer or adapter-based architectures. By contrast, comparable low-resource studies in Nepali, Bangla, Myanmar, and Zarma report more ambitious setups, such as multi-stage Transformer pipelines, error-type-aware modelling, and the systematic use of multilingual pre-trained models like MT5 and M2M100, which highlight practices that are not yet adopted in Maltese work. Consequently, the main gaps for Maltese are the absence of large-scale fine-tuning of multilingual LLMs, limited integration of linguistically grounded error categories, and the lack of a robust gold-standard evaluation resource. Addressing these would provide a clear route to advance Maltese spell checking beyond the current state.

3.5 | Synthesis and Implications for This Dissertation

The development of Maltese spell checking shows a clear trajectory. Early studies (Mangion, 1999; Mizzi, 2000) demonstrated that morphology-aware rules and probabilistic inference could provide some coverage; however, both approaches suffered from over-generation, weak candidate ranking, and limited evaluation. The release of spelling.mt1 marked the first public tool, but its reliance on dictionary look-up and edit distance meant that phonetic and grammatical variation were ignored, producing low hit rates. Later work, such as the phonetic prototype of Buhagiar (2021), improved accuracy but remained experimental. The most substantial advances were made by Debattista (2022),

who tested both Seq2Seq and Transformer architectures, and Oliveri (2025), who developed a Transformer-based Seq2Seq system. Both adopted international evaluation frameworks; however, performance was constrained by their dependence on synthetic corpora and the absence of sufficiently rich, authentic data.

Comparative insights from other low-resource languages reinforce this picture. Rule-based systems cope with data scarcity but require intensive manual effort. Edit-distance approaches are lightweight but generate many False Positives (FPs). N-gram models improve contextual ranking, though their effectiveness depends on the availability of unlabelled corpora and accurate orthographic representation. Statistical systems mitigate data sparsity primarily through curated lexicons, morphological reduction, and sometimes cross-lingual transfer. In contrast, neural systems rely on strategies such as transfer learning and data augmentation. More recently, Transformer-based approaches have demonstrated that combining multilingual pre-training with carefully designed noise yields the strongest results, particularly when paired with richer evaluation.

For Maltese, three gaps stand out: weak contextual modelling, synthetic errors that do not adequately reflect real-world mistakes, and narrow evaluation practices. Addressing these requires moving beyond dictionary-based or small-scale prototypes toward fine-tuned multilingual models that integrate orthographic sensitivity with sentence-level context.

Taken together, these insights underscore the need for a new approach to Maltese spell checking. This dissertation positions itself as the logical next step. By fine-tuning a multilingual LLM, it will build on existing authentic resources and carefully supplement them with synthetic errors modelled on real usage. Evaluation will follow established practices, balancing token-level measures with sentence-level metrics and, where possible, complemented by analyses of error patterns, such as over- and under-correction. In doing so, the project not only addresses the specific gaps left by earlier Maltese studies but also aligns with the broader shift towards LLMs as the latest generation of NLP technology identified in recent research.

4

Methodology

This chapter presents the methodological framework adopted for this study, describing the overall research approach and explaining its relevance to the research questions. An overview of the system's architecture is illustrated in Figure 4.1, which visually summarises the main stages. The following sections provide an account of the datasets used, including their sources, collection procedures, and the pre-processing steps implemented to ensure data quality and consistency. It then outlines the fine-tuning strategy, the models employed, and the evaluation metrics used to evaluate system performance. Moreover, it discusses the main technical and data-related limitations encountered throughout the research and the strategies adopted to mitigate them. Finally, it highlights the ethical considerations relevant to the study and details the measures taken to ensure compliance with ethical standards.

4.1 | Approach and Justification

This study employs a methodological framework that investigates how Large Language Models (LLMs) can be adapted to develop a resource-efficient Maltese spell checker, thereby demonstrating the feasibility of this approach within a low-resourced linguistic context. Furthermore, the fine-tuned model is evaluated using different evaluation frameworks to determine its effectiveness in accurately correcting spelling errors, whether it achieves improved performance compared to prior studies, and its potential to provide a more comprehensive and robust solution than existing Maltese spell checkers. Collectively, these aspects address the core research questions examined in this study.

The approach focuses on fine-tuning an LLM, specifically Meta's LLaMA-3-8B-Intstruct on pairs of incorrect-correct Maltese words and sentences, enabling the model to learn how to transform spelling errors into their correct form. Unlike traditional systems that first detect spelling errors before proposing corrections, the model employed in this re-

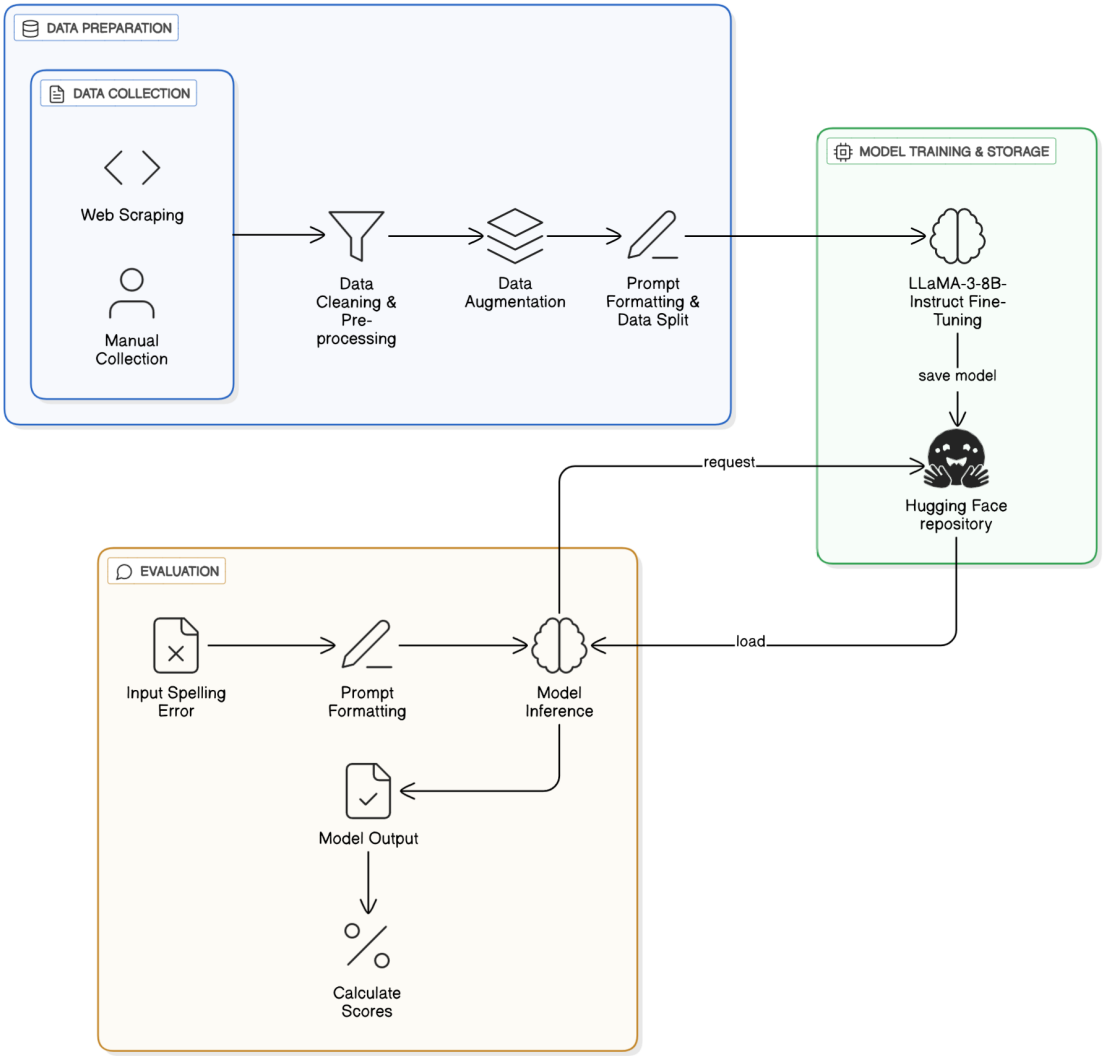


Figure 4.1: Overview of the system architecture illustrating the model-building pipeline (data collection, pre-processing and fine-tuning) and the inference stage (evaluation and user interaction), where the trained model processes input to generate corrected output. Source: Author.

search performs a single-step correction, directly generating the corrected output from the input sequence.

The fine-tuning dataset used for both training and evaluation was constructed by introducing synthetic spelling and grammatical errors into correctly written Maltese text, together with real-world errors and their corresponding corrections collected from past studies.

Synthetic error generation is a data augmentation technique in which controlled er-

rors are systematically inserted into clean text to replicate real-world linguistic inaccuracies and expand the training corpus. The use of synthetic error generation is supported by research that identifies it as an effective strategy for addressing data scarcity and improving model performance (Marier et al., 2025).

The word-level and sentence-level error types that guided the creation of synthetic spelling errors are detailed in Appendix B and Appendix C, respectively. These appendices list the orthographic and grammatical mistakes addressed in this project, which were derived from the linguistic studies of Bezzina (2020), Blundell (2014), Dimech (2013), and Schembri (2016), all of which investigated common spelling mistakes in Maltese. This linguistic foundation ensures that the synthetic data accurately reflects the types of errors encountered in authentic text.

Using these error patterns, synthetic incorrect-correct pairs were generated by systematically replacing correct words or sentence elements with errors according to the documented patterns, while retaining the original correct forms as references. This approach ensures that the resulting dataset reflects realistic error distributions and provides reliable supervision for model training. For illustration, Table 4.1 presents three representative examples of how correct text was transformed into synthetic errors.

The methodological choice to fine-tune an LLM is supported by growing evidence in the Natural Language Processing (NLP) literature, which demonstrates that LLMs, including Meta's LLaMA 3 models, offer substantial advantages over traditional spell checking techniques in terms of contextual understanding, adaptability, and overall performance (X. Han et al., 2021; Martynov et al., 2024; Naveed et al., 2023).

Traditional approaches such as dictionary look-up, edit distance algorithms, and n-gram language models have historically struggled with real-word errors and context-sensitive corrections. These methods operate primarily on surface-level character edits or limited statistical patterns, limiting their ability to account for broader linguistic context. In contrast, LLMs leverage full-sequence modelling and deep contextual representations to detect and correct context-dependent spelling errors. This capability accounts for their improved effectiveness relative to traditional approaches (de Araújo et al., 2024; Penteado & Perez, 2023). In addition, LLMs have also been explored for spelling and grammatical correction in low-resource languages, where results have been particularly promising (Kapelles et al., 2025; Keita et al., 2024; Nurhasanah et al., 2025; Parankusham et al., 2025; Thapa et al., 2025).

In this study, Meta's LLaMA 3 instruction-tuned model with 8 billion parameters is used rather than the standard 8 billion-parameter version. This choice is motivated by evidence showing that instruction tuning substantially improves a model's ability to interpret and execute user-defined tasks. Studies indicate that instruction-tuned models

Table 4.1: Examples of synthetic incorrect-correct pairs generated from the Maltese corpus using documented error patterns. Source: Author.

Correct Version	Synthetically Generated Error	Error Type
kelma	klma	Omission error.
ktieb	kttieb	Insertion error.
abbanduna	abanduna	Replace double consonant with single.
abbażi	abbasi	żchanged to s
Kienet għajta li kellha warajha wkoll il-polz ta' bosta	kienet għajta li kellha warajha wkoll il-polz ta' bosta.	Convert first word to lower case.
Kien hemm il-bżonn li nqajmu u nġeddu kuxjenza u għarfien dwar is-sbuħija tal-Malti, ilsienna, u kif dan jinħtieġjintuża b'mod xieraq u f'waqtu.	Kien hemm bżonn li nqajmu u nġeddu kuxjenza u għarfien dwar sbuħija Malti, ilsienna, u kif dan jinħtieġjintuża mod xieraq u waqtu	Removed articles.
L-istatistika turina b'mod ċar li l-poplu Malti jibqa' jagħżel li jikkomunika bi lsien art twelidu.	L-istatistika turina b'mod ċar li l-poplu Malti jibqa' tagħżel li tikkomunika bi lsien art twelidu.	Generate verb grammar errors.
Nixtieq nippreżenta lill-qarrej Malti xi xogħlijiet tal-poeti nazzjonali ta' tliet pajjiżi Slavoniċi.	Nixtieq nippreżenta lill-qarrej Malti xi xogħlijiet tal-poeti nazzjonali ta' tlieta pajjiżi Slavoniċi.	Number agreement.

follow task instructions more accurately and generalise better across different contexts than their standard counterparts, which, although linguistically competent, are primarily trained for open-ended text continuation rather than explicit instruction-following behaviour (Ren et al., 2024; S. Zhang et al., 2025). Empirical findings further reveal that such models achieve improved performance on structured, context-dependent tasks while retaining their underlying linguistic knowledge (Weerawardhena et al., 2025). These capabilities make the instruction-tuned variant particularly well-suited for devel-

oping a spell checker, where the model must reliably follow the instruction to evaluate the input and generate corrected output if necessary.

To optimise model performance, the fine-tuning process involves testing a range of model configurations to identify which parameters and components have the most significant influence on performance. This iterative approach aims to achieve a balanced model capable of effectively handling diverse types of spelling errors while minimising the risks of overfitting and underfitting. Further details regarding the model settings, tuning procedures, and evaluation criteria are outlined in the subsequent sections.

In summary, the methodological framework presented in this section establishes a structured approach for developing a resource-efficient Maltese spell checker using Meta's LLaMA-3-8B-Instruct model as the base model. By integrating linguistic insights, synthetic error generation, and model optimisation strategies, the approach provides a robust foundation for investigating how effectively the fine-tuned model can address the research objectives. The following sections explain the datasets used, together with the experimental design, training configuration, and evaluation procedures used in this research, in further detail.

4.2 | LLaMA-3-8B-Instruct Model

Meta's LLaMA 3 family encompasses a range of models with 8 billion, 70 billion, and 405 billion parameters, each released in both base and instruction-tuned variants. Recent empirical research has identified the LLaMA 3 models as among the most capable open-source LLMs, with the models achieving state-of-the-art results across numerous benchmarks. Collectively, these findings firmly establish the LLaMA models among the most capable open-source LLMs for a broad spectrum of NLP applications (Grattafiori et al., 2024; Y. Hu et al., 2025; Huang, Ma, et al., 2024; Huang, Zheng, et al., 2024).

Nevertheless, while often described as open-source, LLaMA 3 is more accurately characterised as an open-weight model. Its pre-trained parameters are publicly available via Meta's official repository but are distributed under the Community Licence, which regulates their use and redistribution. The licence explicitly permits research and academic applications, allowing the model to be freely employed for scholarly and limited commercial purposes, provided that proper attribution and compliance with the specified conditions are maintained (Meta AI, 2024).

LLaMA 3 employs a Byte-Pair-Encoding (BPE) tokeniser implemented through the `tiktoken` framework. The LLaMA 3 tokeniser comprises a 128,000-token vocabulary that extends the original 100,000-token, English-oriented `tiktoken` base with an addi-

tional 28,000 tokens drawn from various non-English languages (Grattafiori et al., 2024). This expanded design improves multilingual coverage, enabling the system to process diacritic and language-specific characters not present in English, such as those found in Maltese. As illustrated in Table 4.2, the LLaMA-3-8B-Instruct tokeniser effectively handles Maltese letters including **ċ, ġ, ħ, ż**. Although these characters are not represented as single-token entries within the vocabulary, they are encoded as sequences of Unicode Transformation Format (UTF)-8 bytes. This byte-level representation allows the model to interpret and process all Maltese letters without producing Out-of-Vocabulary (OOV) tokens. Consequently, despite the absence of distinct token entries, the tokeniser remains fully capable of encoding Maltese text accurately and consistently.

Table 4.2: Examples of Maltese-specific letters not present in the English alphabet and their corresponding token representations as processed by the LLaMA 3 tokeniser. Source: Author.

Example	**Visualised UTF-8 Encoding Result**	**Tokenised Result**	**Visualised Token Result**
ċ	Äĭ	[128, 233]	[Ä, ĭ]
ċurkett	Äĭurkett	[128, 233, 324, 74, 7211]	[Ä, ĭ, ur, k, ett]
ġ	Ä¡	[128, 94]	[Ä, ¡]
ġugarell	Ä¡ugarell	[128, 94, 773, 548, 657]	[Ä, ¡, ug, are, ll]
ħ	Ä§	[128, 100]	[Ä, §]
ħanut	Ä§anut	[128, 100, 38918]	[Ä, §, anut]
ż	Å¼	[6077]	
żejt	Å¼ejt	[16020, 58305]	[Å¼e, jt]

Given the scope and computational constraints of this dissertation, the 8-billion-parameter variant was selected. Although larger models within the LLaMA 3 family achieve higher benchmark results, their hardware requirements significantly exceed the capacity of consumer-grade GPUs. The 8 billion variant, therefore, offers an effective balance between performance and efficiency, making it practical for fine-tuning within available computational limits.

Furthermore, the instruction-tuned version was chosen to ensure the model could better interpret user intent and produce contextually appropriate responses. This decision is supported by the findings of Wei and Liu (2024), who report that instruction-tuned variants of LLaMA 3 exhibit improved alignment with user instructions and enhanced response relevance compared to their base counterparts.

Taken together, these characteristics make the LLaMA-3-8B-Instruct model particularly well suited for developing a Maltese spell checker, where precise language modelling and instruction-following behaviour are essential, and the process must remain feasible within the available computational resources.

4.3 | Dataset Collection and Preparation

This section describes the data used in this study, outlining their sources, structure, and the pre-processing steps applied to ensure consistency and quality. The datasets were designed to provide the model with reliable examples of both correct and incorrect Maltese text, supporting the fine-tuning process for spelling and grammatical correction. Particular attention was given to data selection and preparation to ensure that the collected Maltese text adhered, as closely as possible, to correct linguistic forms ensuring accurate linguistic standards for model training.

4.3.1 | Data Sources

A gold-standard dataset of incorrect-correct Maltese text pairs for spell checking purposes does not currently exist. Initially, the MLRS Korpus Malti was intended to serve as the primary data source for this research due to its size and linguistic relevance. However, upon closer inspection, it was observed that the corpus did not consist entirely of error-free Maltese text and therefore could not be used as a definitive gold-standard reference for this task. Although it remains a valuable linguistic resource, occasional spelling and grammatical inconsistencies make it unsuitable as the sole basis for developing an accurate error-correction dataset. Samples from the corpus were examined and validated in consultation with linguistic experts to confirm the identified inconsistencies. Table 4.3 illustrates a few examples of spelling errors identified within the MLRS Korpus Malti.

Table 4.3: Examples of errors in the MLRS Korpus Malti. Source: Author.

Spelling error in MLRS Korpus Malti	**Expected**	**Error Types**
B'kollox għamlet **xagħar** u nofs l-isptar	B'kollox għamlet **xahar** u nofs l-isptar	Real-word/Grammatical error: Although "xagħar" is a correct Maltese word, it is contextually incorrect here, having been used in place of its homophone "xahar".
u joqgħod jiżfen **zaqqu** ma' dahar l-ewwel waħda li jsib	u joqgħod jiżfen **żaqqu** ma' dahar l-ewwel waħda li jsib	Diacritic spelling error
imqar nhar ta' **ħadd** biss	imqar nhar ta' **Ħadd** biss	Grammatical error: The word "ħadd" should be capitalised in this instance, since it denotes the name of a day.
hawn bank robber armat sa **snienul'** qalli minn taħt l-ilsien	hawn bank robber armat sa **snienu** qalli minn taħt l-ilsien	Insertion error.
jitolbuh il-firma **tiegħul**	jitolbuh il-firma **tiegħu**	Insertion error.
bissnien irqaq u spazji dojoq	**bis-snien** irqaq u spazji dojoq	Article error: no hyphen and it is combined with word.
jiffurmaw klikek biex ikunu jistgħu **jhokku** dahar xulxin	jiffurmaw klikek biex ikunu jistgħu **jħokku** dahar xulxin	Diacritic error.
ax pruvajt kemm **il darba naghmel** server biex jaqbdu **shabi mijaj**	**għax** pruvajt kemm **il-darba nagħmel** server biex jaqbdu **sħabi miegħi**	Omission of "għ" in words; no hyphen in article; diacritic errors; non-word error - "mijaj" does not exist in Maltese.
hbieb jien **diga** impjegajt **wihed mijaj** biex waqt li jkunu **addejjin il miljuni ta** gamiem	**ħbieb** jien **digà** impjegajt **wieħed miegħi** biex waqt li jkunu **għaddejjin il-miljuni ta'** gamiem	Diacritic error; omission of accent from word; using "i" instead of "ie"; non-word error - "mijaj", and "addejjin" do not exist in Maltese; omission of article hyphen; omission of apostrophe.

To address this limitation, the study drew on other reliable sources that contained correctly written Maltese text. These texts were then used to create artificial spelling and grammatical mistakes that resemble real-world writing errors, based on findings from the literature outlining common spelling errors in Maltese, resulting in pairs of incorrect and corrected sentences.

The correctly written Maltese text used to construct the dataset was collected from several high-quality sources, chosen to ensure both linguistic accuracy and diversity. These included linguistic resources such as the publication *L-Ilsien Malti Għal Qalbi*, *Sagħtar*[1] publications, and Curia-related court cases. These data sources were provided for use in this study by different collaborators. *L-Ilsien Malti Għal Qalbi* publication was provided by Prof. Michael Spagnol and Mr. Thomas Pace, while Mr. Christopher Giordano provided the *Sagħtar* publications. Additionally, Mr. Robert Camenzuli provided the publicly available Curia-related court cases, containing Maltese legal texts. These sources offered high-quality examples of accurately written Maltese across educational, formal, and legal domains.

Additional material was obtained from verb.mt[2] and the Maltese dictionary from the Intercontinental Dictionary Series (IDS)[3]. The verb.mt platform, is a publicly accessible online resource developed and maintained by Mr. Reuben Degiorgio, featuring a substantial collection of correctly written Maltese verbs and their conjugates.

The IDS Maltese dictionary, developed by Spagnol et al. (2020), is an open-source linguistic dataset that provided an additional source of correctly written Maltese words. Together, these datasets contributed to a broader and more representative lexical coverage of Maltese words.

Furthermore, a small set of incorrect-correct pairs was derived from assignments involving the correction of local newspaper articles provided by Mr. Thomas Pace, as well as a few manually corrected examples from the MLRS Korpus Malti. The subset of manually corrected spelling errors from the MLRS Korpus Malti was prepared by Ms. Amy Borg. However, due to the extensive size of the corpus and the substantial time required for manual correction, it was not feasible to expand this dataset further. Consequently, the portion already completed was retained and incorporated into this dissertation as part of the collected data.

Finally, selected datasets from previous Maltese studies were also incorporated. These included the *Qari tal-Provi* dataset compiled by Debattista (2022), as well as the real-world spelling errors collected by Busuttil (2025).

[1] https://saghtar.org.mt/
[2] https://verb.mt/
[3] https://ids.clld.org/contributions/843

Table 4.4 provides a detailed overview of the data sources used, their licensing conditions, and the necessary approvals obtained.

Table 4.4: Overview of the data sources used in this study, including their topics, licensing conditions, and ethical approvals. Source: Author.

Data Source	License	Approval	Topic
L-Ilsien Malti Għal Qalbi	Appendix G.0.1	Appendix G.0.1	Linguistics and education
Curia-related court cases	Appendix G.0.2	Appendix G.0.2	Legal
verb.mt	Appendix G.0.3	Appendix G.0.3	Conjugates of verbs
Sagħtar	Appendix G.0.4	Appendix G.0.4	General interest articles, literature and education
IDS Dictionary	Creative Commons Attribution 4.0 International License	Appendix G.0.5	Maltese dictionary of words
Correction of newspaper articles	Appendix G.0.5	Appendix G.0.5	Current affairs
Subset of manually corrected errors from MLRS Korpus Malti	Appendix G.0.5	Appendix G.0.5	Incorrect word forms
Qari tal-Provi (Debattista, 2022)	Appendix G.0.5	Appendix G.0.5	Corrections of spelling errors
Human collected error data (Busuttil, 2025)	Appendix G.0.5	Appendix G.0.5	Authentic human spelling errors

4.3.2 | Data Collection

This section describes the pre-processing and data cleaning procedures applied to extract sentences and words from the data sources outlined in Section 4.3.1, detailing how clean lists of correctly written Maltese sentences and words were produced for each source. Specifically, Sections 4.3.2.1 and 4.3.2.2 provide a detailed explanation of the methods used to collect and process data from unstructured and structured sources,

respectively.

It is important to note that the subset of manually corrected errors from the MLRS Korpus Malti, *Qari tal-Provi* (Debattista, 2022), and the real-world spelling errors (Busuttil, 2025) required no additional processing, as these datasets were already organised into source-target pairs. The use and integration of these datasets into the final compiled dataset are discussed further in Section 4.3.4.

4.3.2.1 | Processing of Text-Based Documents and Publications

The *L-Ilsien Malti Għal Qalbi* and *Sagħtar* publications were provided as text-based documents, whereas the Curia-related court cases were available through publicly accessible web links. The Maltese text from the document-based sources was manually copied into a text editor to facilitate processing. This approach was adopted because the documents contained structural inconsistencies, varied layouts, and formatting artefacts, making the development of an automated extraction script impractical. Manual selection, therefore, provided a more feasible means of isolating relevant content and establishing a standardised structure for subsequent pre-processing. Similarly, inconsistencies in the Hypertext Markup Language (HTML) structure and formatting of the court case web pages also made automated web scraping unfeasible, hence, the same manual extraction procedure used for the text-based documents was applied. Following this stage, additional pre-processing steps were performed to segment the text into sentences and, subsequently, into words, as the data were, at this point, organised into larger chunks of relevant content.

From the data obtained through the aforementioned extraction process, text from the three data sources was first automatically processed using Python scripts to extract individual sentences. This was achieved by splitting the text based on terminal punctuation marks, as outlined below:

- Full stop – “.”
- Question mark – “?”
- Exclamation mark – “!”
- Ellipsis – “. . . ”
- Full stop followed by double quotation mark – “."”
- Question mark followed by double quotation mark – “?"”
- Exclamation mark followed by double quotation mark – “!"”

- Ellipsis followed by double quotation mark – "…"
- Full stop followed by single quotation mark – ".”
- Question mark followed by single quotation mark – "?"
- Exclamation mark followed by single quotation mark – "!"
- Ellipsis followed by single quotation mark – "…"'

This approach mirrors the natural way sentences are typically delineated, however, it has certain limitations. For instance, abbreviations such as "Prof.", "Dr.", or "Kap." include full stops that do not indicate the end of a sentence, yet are mistakenly treated as such by the automated script. Consequently, after automatic segmentation, the resulting output was manually reviewed to correct any sentence splits caused by this limitation and other notable errors, ensuring that sentences were preserved as accurately as possible, which is a crucial step in developing an error-free dataset.

Once the list of Maltese sentences was compiled, further data cleaning was performed using Python scripts. This process included standardising punctuation marks, removing redundant or double white spaces, eliminating spaces after articles, and deleting newline characters. Additional automated cleaning steps ensured proper capitalisation, added missing terminal punctuation marks, and flagged mismatched punctuation marks (e.g., the presence of a single quotation mark in a sentence when a pair is expected) for manual correction based on the original sources. The result of this process was a refined collection of correctly written Maltese sentences for each data source.

From these cleaned sentences, a separate list of individual words was generated. Words were automatically extracted using whitespace separation, and the output was then automatically processed to identify and remove tokens that do not contain any alphabetic characters. Remaining tokens were cleaned of punctuation marks and detached from articles to retain only valid word forms. Further cleaning steps eliminated unwanted spaces, null values, and duplicate entries. Finally, the list was manually inspected to ensure linguistic accuracy and consistency, with any invalid tokens and non-Maltese words omitted. This process yielded a high-quality list of correctly written Maltese words for each data source.

4.3.2.2 | Processing of Structured and Web-Based Sources

Structured and web-based sources required different extraction and cleaning procedures compared with text-based publications. These resources were processed primar-

ily using Python scripts, with manual review applied where necessary to ensure data quality and consistency.

The IDS Maltese dictionary was initially distributed in spreadsheet format and was subsequently converted into a Comma Seperated Values (CSV) file to facilitate automated processing. Relevant Maltese entries were extracted automatically from the CSV file using Python, resulting in a comprehensive list of lexical items. The extracted data was then subjected to a series of cleaning and normalisation procedures, implemented in Python, consistent with the word-level token processing methods described in Section 4.3.2.1. This process generated a curated list of correctly written Maltese words that adhered to standard orthographic conventions.

Another essential source of data used in this study was verb.mt. Following Mr. Reuben Degiorgio's approval, a Python-based web scraper was implemented to collect the verbs and their corresponding inflectional forms. A subset of the extracted data was then manually reviewed to verify that the inflectional patterns were accurately captured and that no encoding or formatting inconsistencies remained. This process resulted in an extensive list of correctly written Maltese words.

Finally, another data source that required pre-processing and cleaning was the set of documents containing newspaper correction assignments. Relevant data were extracted from these documents, with each entry presenting a corrected form followed by the original spelling error in brackets. These pairs were automatically extracted using a Python script, which formatted the data into a two-column structure, with one column containing the correction and the other the corresponding spelling error. The identification of spelling errors was determined based on the presence of text enclosed in brackets. Once the data were correctly structured as incorrect-correct pairs in this two-column format, the entries in the spelling error column were cleaned to remove any surrounding brackets. This process produced a curated list of spelling errors and their corresponding corrections.

4.3.3 | Synthetic Errors

After collecting and processing the data as described in Section 4.3.2 from the data sources: *L-Ilsien Malti Għal Qalbi*; *Sagħtar* publications; Curia-related court cases documents; verb.mt; IDS Maltese dictionary; and correction of newspaper articles, the resulting data from each data source was compiled into two main lists: one containing correctly written Maltese sentences and another containing individual words. Table 4.5 presents the data sources used to create these lists and the number of entries each contributed.

Table 4.5: Overview of data sources and the number of words and sentences extracted from each source. Source: Author.

Data Source	Number of Sentences	Number of Words
L-Ilsien Malti Għal Qalbi	869	3,484
Sagħtar Publications	1,273	4,233
Curia Legal Texts	2,621	5,262
verb.mt	—	161,363
IDS Dictionary for Maltese	—	1,603

A Python script was used to merge the outputs from all the sources listed in Table 4.5. The sentence data from the various sources were combined to form a unified sentence list, while the word data were merged separately to produce a consolidated word list. Both lists were then automatically processed to remove duplicates and null entries. The final dataset comprised **169,618** distinct words in the word list and **4,763** distinct sentences in the sentence list. These two lists were subsequently used to generate word-level and sentence-level synthetic errors, thereby creating word-level and sentence-level incorrect-correct datasets.

The synthetic errors employed in this study were designed to reflect realistic writing mistakes commonly observed in the Maltese language. To ensure authenticity, the error patterns were informed by linguistic studies investigating frequent spelling and grammatical errors in written Maltese (Bezzina, 2020; Blundell, 2014; Dimech, 2013; Schembri, 2016). These studies provided insights into the most prevalent error types in Maltese writing, which were used to guide the systematic introduction of controlled noise into the collected data. This process ensured that the generated incorrect forms simulated natural error distributions representative of genuine human writing mistakes. Appendix B and Appendix C describe the different word-level and sentence-level error types observed in real-world writing and those used to create the incorrect-correct datasets.

The process of generating synthetic errors was carried out automatically using a Python script. Appendix E lists all the word-level spelling error operations implemented by the script, which were developed based on the aforementioned literature addressing common mistakes in Maltese writing. These operations were designed to

correspond to the error types outlined in Appendix B, as supported by the same body of literature. For each correctly written word in the unified word list, the script automatically applied every applicable error operation to generate the corresponding incorrect variants.

The process of generating sentence-level synthetic errors was also carried out automatically using a Python script. Appendix F lists all the sentence-level error operations implemented by the script, which were developed based on the same aforementioned literature addressing common grammatical and orthographic mistakes in Maltese writing. These operations were designed to correspond to the error types outlined in Appendix C, as supported by the same body of literature. In addition to these sentence-level operations, word-level spelling errors were also incorporated into the sentences to ensure that each example reflected a realistic combination of orthographic and grammatical mistakes. For each correctly written sentence in the unified sentence list, the script automatically applied every applicable error operation to generate the corresponding incorrect variants.

Once the synthetic error generation for words and sentences was completed, it resulted in two incorrect-correct datasets, one for words and one for sentences. In total, the incorrect-correct word dataset comprised **2,042,310** pairs, while the incorrect-correct sentence dataset contained **187,154** pairs.

The following section discusses in further detail how these datasets were combined with the other data sources that were already formatted in incorrect-correct form, namely the subset of manually corrected errors from the MLRS Korpus Malti; *Qari tal-Provi* (Debattista, 2022), and the human-collected error data (Busuttil, 2025), to create the training, evaluation and test datasets.

4.3.4 | Final Dataset

Following the generation of the word and sentence-level incorrect-correct datasets using synthetic spelling errors described in Section 4.3.3, the final stage involved combining these datasets with the other collected incorrect-correct datasets identified in Section 4.3.1, namely the subset of manually corrected errors from the MLRS Korpus Malti, *Qari tal-Provi* (Debattista, 2022), and the real-word error data (Busuttil, 2025). This step ensured the creation of a comprehensive dataset encompassing both synthetically generated and authentic human-derived error-correction pairs. In addition to these error-correction pairs, certain datasets also contained correct-correct pairs, representing cases where both the source and target entries were already error-free. These instances were retained to preserve contextual diversity, maintain dataset balance, and

expose the model to examples where inputs are correct and should therefore remain unchanged.

For the *Qari tal-Provi* and human-collected error datasets, 50% of the data were reserved to act as a test set for testing the model's generalisation capability on unseen real-world errors, while the remaining 50% were included into the training data to expose the model to real-world errors.

Consequently, the training, evaluation, and test datasets were compiled using the sources listed in Table 4.6 for the word-level data and Table 4.7 for the sentence-level data. Each dataset was divided into training, evaluation, and testing subsets following an 80:10:10 ratio to ensure balanced representation and consistent model assessment. On the other hand, Table 4.8 presents the data sources and record counts used to construct the unseen real-world error test dataset for sentences. A corresponding real-world error dataset was not available for the word-level data.

Table 4.6: An overview of the datasets used to compile the word-level training (80%), evaluation (10%), and test (10%) datasets, including both synthetically generated and collected incorrect-correct data sources. Source: Author.

Data Source	**Incorrect-Correct records**	**Correct-Correct records**
Newspaper corrections	1,381	6
Subset of manually corrected errors from MLRS Korpus Malti	753	0
Incorrect-correct word dataset of synthetic errors described in Section 4.3.3	2,042,310	0
Total (with duplicate rows present)	**2,044,450**	
Total (without duplicate rows)	**2,044,368**	

4.3.4.1 | Model Prompt Format

The training and evaluation datasets compiled from Table 4.6 and Table 4.7 were formatted into clear instruction-response pairs to align with the model's instruction-tuning framework. Each instruction presented an input containing a spelling error or an er-

Table 4.7: An overview of the datasets used to compile the sentence-level training (80%), evaluation (10%), and test (10%) datasets, including both synthetically generated and collected incorrect-correct data sources. Source: Author.

Data Source	Incorrect-Correct records	Correct-Correct records
Newspaper corrections	1,381	6
50% of Qari tal-Provi	2,131	73
50% of the human collected data by Busuttil (2025)	1,294	113
Incorrect-correct sentences dataset of synthetic errors described in Section 4.3.3	187,154	0
Total (with duplicate rows present)	**192,152**	
Total (without duplicate rows)	**192,106**	

roneous sentence, while the corresponding response provided the corrected version of that same text. This structure focused the model's learning on correction-specific behaviour rather than conversational interaction, ensuring that generated outputs remained concise, accurate, and contextually appropriate. The prompt employed for fine-tuning was structured as follows:

```
FINE_TUNING_MODEL_TEMPLATE = (
    "<s> [INST] <<SYS>>Analyse the Maltese input text and if required
   provide a suitable Maltese correction <</SYS>>"
    " {input} [/INST] {output} </s>"
)
```

In contrast, the test datasets derived from Table 4.6, Table 4.7, and the real-world test set did not follow the same format, as the correct responses were intentionally omitted from the test prompts. This allowed the erroneous inputs to be passed to the model without their corresponding corrections, enabling the model to generate its own predictions. The prompt employed for inference was structured as follows:

```
INFERENCE_MODEL_TEMPLATE = (
    "<s> [INST] <<SYS>>Analyse the Maltese input text and if required
   provide a suitable Maltese correction <</SYS>>"
```

Table 4.8: A summary of the datasets and record counts used to construct the real-world error test dataset for the sentence-level model. Source: Author.

Data Source	Incorrect-Correct records	Correct-Correct records
50% of Qari tal-Provi	2,202	3
50% of the human collected data by Busuttil (2025)	1,403	4
Total (with duplicate rows present)	**3,612**	
Total (without duplicate rows)	**3,603**	

```
    " {input} [/INST]"
)
```

Following the prompt formatting stage, each dataset underwent an additional validation step to identify and remove any duplicate instruction-response pairs. This cleaning process resulted in a total of **1,910,620** records for the word-level data and **192,106** records for the sentence-level data. The final datasets were then divided into training, evaluation, and testing subsets following an 80:10:10 ratio, corresponding to **1,528,503**, **191,063**, and **191,063** records for the word-level data and **153,684**, **19,211**, and **19,211** records for the sentence-level data, respectively.

After constructing the required datasets, the subsequent stage involved configuring and executing the fine-tuning process, which is described in detail in Section 4.4.

4.4 | Model Fine-Tuning

The fine-tuning process aims to adapt the LLaMA-3-8B-Instruct model for Maltese spell checking using the datasets described in Section 4.3. As the task involves learning explicit correction mappings between spelling errors and correctly spelt text, the training adopted a Supervised Fine-Tuning (SFT) approach.

Recent research highlights the effectiveness of SFT in aligning LLMs with specific instructions, improving their ability to perform task-oriented behaviours across a range of domains. Through exposure to labelled examples, SFT enables a pre-trained model to learn direct input-output mappings, allowing it to specialise in particular applications.

This mechanism extends naturally to sequence-to-sequence (Seq2Seq) correction tasks, where the model is trained on pairs of incorrect and corrected sentences, enabling it to learn the transformation from spelling errors to correctly written text (Dong et al., 2024). In this dissertation, the same principle is applied to Maltese spell checking, where SFT allows the model to learn orthographic and grammatical correction patterns directly from labelled examples rather than relying on manually defined linguistic rules.

To efficiently apply this training strategy on a large-scale model, the approach used PEFT techniques. Specifically, it employed the Quantised Low-Rank Adaptation (QLoRA) method, which combines low-bit quantisation with Low-Rank Adaptation (LoRA) adaptation to enable large models to be fine-tuned on limited hardware while maintaining the performance of the pre-trained base model. During fine-tuning, the pre-trained weights remain frozen, while only the newly incorporated low-rank adapter parameters are updated via back-propagation (Dettmers et al., 2023; X. Zhang, Rajabi, et al., 2023). This approach substantially reduced the computational and memory demands compared with full fine-tuning, making the process more feasible while preserving the performance of the base model. Figure 4.2 illustrates a comparison showing how QLoRA differs from both full fine-tuning and LoRA fine-tuning.

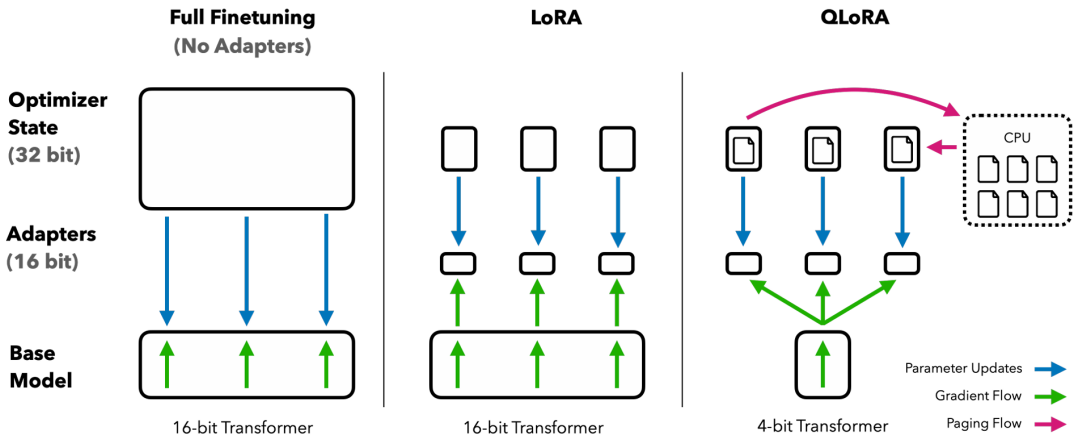


Figure 4.2: Comparison of different fine-tuning methods and their memory requirements. In full fine-tuning, all model parameters are updated in high precision, resulting in the highest computational and memory cost. LoRA reduces this cost by introducing trainable low-rank adapters while keeping the original model weights frozen. QLoRA further improves efficiency by quantising the transformer model to 4-bit precision and using paged optimisers to manage memory usage during training. Source: Dettmers et al. (2023).

Section 4.4.1provides a detailed examination of the methodological processes, im-

plementation procedures, and configuration choices employed throughout this dissertation.

4.4.1 | Training Setup

Before initiating fine-tuning, it was essential to establish all model, quantisation, and training configurations. This ensured that the fine-tuning process was both stable and computationally efficient. The configuration parameters were determined through a review of empirical research on QLoRA and other PEFT strategies applied to LLMs.

The subsequent sections describe the specific configuration values used to fine-tune the model and explain the role of each parameter within the overall training process. Any configurations that are not listed means that the default values were applied by the `SFTConfig` class from the `trl`[4] library (von Werra et al., 2020).

4.4.1.1 | QLoRA Configuration

The QLoRA framework combines low-bit quantisation with low-rank adaptation (LoRA) to enable PEFT of LLMs on limited hardware resources (Dettmers et al., 2023). In this study, both the quantisation and adapter components were configured according to empirically supported practices established in prior research.

Quantisation Quantisation determines how the model's weights are represented in lower precision to reduce memory requirements while maintaining numerical stability. In this study, the base LLaMA-3-8B-Instruct model was loaded in 4-bit NormalFloat (NF4) precision with double quantisation and Brain Floating Point 16-bit (BFloat16) computation. This configuration, implemented using the `BitsAndBytesConfig` class from the `transformers`[5] library (Wolf et al., 2020), was applied consistently across both word-level and sentence-level fine-tuning experiments.

The selected quantisation setup was informed by prior research demonstrating that low-precision representations can significantly reduce memory consumption while preserving model fidelity. Dettmers et al. (2023) showed that combining NF4 quantisation with double quantisation provides an effective balance between computational efficiency and model performance in large-scale fine-tuning. This is further supported by Lee (2025), who highlights that 4-bit quantisation offers an optimal trade-off, maintaining near-baseline performance while significantly reducing computational costs. Consistent with this approach, computation was performed in `BFloat16` precision, a for-

[4] `https://github.com/huggingface/trl`
[5] `https://github.com/huggingface/transformers`

mat commonly paired with 4-bit quantisation, owing to its dynamic range and its robustness across training tasks (Kalamkar et al., 2019).

Adapter Configuration The trainable low-rank adapters were defined using the `LoraConfig` class from the `peft`[6] library (Mangrulkar et al., 2022). The concept originates from the LoRA method introduced by E. J. Hu et al. (2021), which demonstrated that large transformer updates can be approximated by two small low-rank matrices. QLoRA extends this approach to quantised models, allowing only these lightweight matrices to be trained while all pre-trained weights remain frozen (Parthasarathy et al., 2024).

In this study, the LoRA configuration was set with a rank of 64 and a scaling factor (α) of 16, following the approaches of Dettmers et al. (2023) and Li et al. (2024), both of whom fine-tuned LLMs using the QLoRA technique. Additionally, Dettmers et al. (2023) note that a LoRA dropout value of 0.05 is effective for smaller models, such as those with 7 and 13 billion parameters, and this setting was therefore adopted for this study. Additionally, adapters were applied to the model's principal attention (`q_proj`, `k_proj`, `v_proj`, `o_proj`) and feed-forward (`up_proj`, `down_proj`, `gate_proj`) projection layers. These configurations were employed to both word-level and sentence-level fine-tuning experiments.

4.4.1.2 | Optimisation and Learning Dynamics

Optimisation was performed using the `paged_adamw_32bit` optimiser across both the word-level and sentence-level models. Adam-based optimisers are widely adopted in LLM (Jin et al., 2023), with the AdamW variant commonly recommended to mitigate overfitting through decoupled weight decay. The paged variant of AdamW further improves memory efficiency and has been identified as particularly advantageous for fine-tuning large-scale models (Parthasarathy et al., 2024). Moreover, the choice to maintain optimiser states in 32-bit precision is supported by the findings of Alahmari et al. (2024), who reported that using the 32-bit version resulted in lower perplexity and more consistent outcomes compared with lower-precision alternatives, indicating improved stability and reproducibility.

The initial learning rate was set to 2×10^{-4}, following the findings of Parthasarathy et al. (2024), who identified this configuration as a suitable starting point for fine-tuning LLMs. Subsequently, an empirical comparison was conducted using a learning rate of 2×10^{-5}, based on the recommendations of Pareja et al. (2024), whose work demon-

[6] `https://github.com/huggingface/peft`

strated that this lower rate is also effective for fine-tuning smaller pre-trained models containing between 3 and 7 billion parameters.

In line with the recommendations of Pareja et al. (2024), who suggest that larger batch sizes can improve optimisation stability, this study adjusted the batch configuration according to the available hardware capacity. Therefore, the batch size was increased as far as GPU memory permitted to achieve efficient utilisation. For the word-level model, a batch size of 64 with a gradient accumulation of 32 was used, while the sentence-level model employed a batch size of 8 with a gradient accumulation of 4. Gradient accumulation was applied to emulate larger effective batch sizes by combining gradients from multiple smaller forward passes before performing a single optimisation step. This method ensured stable convergence while maintaining computational feasibility.

A cosine learning rate scheduler was adopted as it has become the standard approach in LLM training and fine-tuning. This strategy allows the learning rate to reach its peak shortly after the warm-up phase before gradually decreasing to around 10% of its maximum value. Such a schedule provides a practical balance between rapid early adaptation and stable late-stage convergence. Since the introduction of the LLMs, such as Generative Pre-trained Transformer (GPT) and LLaMA, the cosine schedule has remained the standard choice due to its consistent stability and effectiveness across LLM fine-tuning tasks.

Furthermore, the number of training epochs was set to 3, following the observations of Jin et al. (2023), who observed that fine-tuning LLMs typically requires only 2-3 epochs, with the best results often achieved between the second and third epoch. Based on this finding, the initial configuration employed 3 full epochs to allow sufficient learning while avoiding unnecessary overfitting. In addition, an early-stopping mechanism was implemented to terminate training automatically if no further improvement was observed in the evaluation loss, thereby ensuring computational efficiency and model generalisation.

The configuration parameters outlined in this section (4.4.1.2), were applied consistently across both the word-level and sentence-level models. For clarity and conciseness, Section 4.4.1.3 presents a summarised overview of all configurations, distinguishing between the initial fine-tuning settings and the subsequent empirical adjustments made during experimentation.

4.4.1.3 | Configurations Summary

This section provides an overview of the configurations used to fine-tune both the word-level and sentence-level models, highlighting the key settings applied throughout the training process. Table 4.9 summarises the setup for the word-level model, whereas Table 4.10 presents the corresponding configuration for the sentence-level model. Furthermore, the empirical comparison carried out with a different learning rate uses the same configurations presented in these tables. The only modification is the adjustment of the learning rate from 2×10^{-4} to 2×10^{-5} for both the word-level and sentence-level models.

Table 4.9: Fine-tuning configurations for word-level model. Source: Author.

Parameter	**Value**
`lora_r`	64
`lora_alpha`	16
`lora_dropout`	0.05
`learning_rate`	2×10^{-4}
`lr_scheduler_type`	cosine
`num_train_epochs`	3
`per_device_train_batch`	64
`gradient_accumulation_steps`	32
`per_device_eval_batch`	64
`optim`	paged_adamw_32bit
`max_steps`	-1
`eval_strategy`	steps
`eval_steps`	750
`early_stopping_patience`	5
`early_stopping_threshold`	0.01

Table 4.10: Fine-tuning configurations for sentence-level model. Source: Author.

Parameter	Value
lora_r	64
lora_alpha	16
lora_dropout	0.05
learning_rate	2×10^{-4}
lr_scheduler_type	cosine
num_train_epochs	3
per_device_train_batch	8
gradient_accumulation_steps	4
per_device_eval_batch	8
optim	paged_adamw_32bit
max_steps	-1
eval_strategy	steps
eval_steps	2000
early_stopping_patience	5
early_stopping_threshold	0.01

4.4.2 | Fine-Tuning

The LLaMA-3-8B-Instruct model and tokeniser were first retrieved from Meta's official Hugging Face repository[7]. Quantisation was then applied following the QLoRA configuration described in Section 4.4.1.1 to enable efficient fine-tuning. Subsequently, the `SFTTrainer` class from the `trl` library was initialised to perform supervised fine-tuning (SFT) on the quantised model using the training configurations described in Section 4.4.1.2. Throughout training, performance was monitored using Weights and

[7] https://huggingface.co/meta-llama

Biases (WandB)[8]. Once training was completed, the fine-tuned LoRA adapters were reloaded and merged back into the base model to produce a single optimised model. The final fine-tuned model was then saved and uploaded to a dedicated Hugging Face repository.

During inference, text generation was performed using a temperature of 0.5 and a top-p of 0.9. The temperature parameter controls the level of randomness in token selection, where lower values make outputs more deterministic and higher values increase variation. A temperature of 0.5 is low enough to ensure stable and consistent predictions, yet it still provides slightly more flexibility than a temperature of 0, which could make the model overly rigid and less accurate when multiple valid corrections exist.

The top-p parameter (nucleus sampling) restricts token selection to the most probable words until a cumulative probability of 0.9 is reached. This setting helps maintain fluent and natural output while avoiding unlikely tokens and aligns with common practice in LLM research (Nguyen et al., 2025).

4.4.3 | Discarded Experiments

Before conducting the final experiments outlined in Sections 4.4.1 and 4.4.2, several fine-tuning trials were performed using a smaller, still-developing version of the dataset along with various fine-tuning configurations.

The configurations of these early models were not consistently documented, making it difficult to reproduce the experiments reliably. Additionally, only accuracy values were recorded, and the lack of recorded configurations prevented the models from being reproduced and evaluated on the complete set of evaluation metrics used in the final analysis.

Furthermore, as the dataset grew and became more comprehensive, the results of these early experiments became less relevant, as they no longer reflected the conditions of the final dataset.

Consequently, the early models were discarded, and only the fully documented models trained on the final dataset presented in Section 4.3.4 are presented in this dissertation. A summary of the approximate dataset sizes and the hyperparameter settings used in some early experiments can be found in Appendix H.

[8] `https://wandb.ai/site`

4.5 | Evaluation Metrics

The evaluation strategies employed in this study differ between the two fine-tuned models, reflecting the distinct linguistic dimensions each addresses. The word-level model is evaluated based on its effectiveness in correcting isolated lexical errors, while the sentence-level model is assessed for its ability to produce contextually accurate and coherent text. This dual evaluation framework ensures that each model is measured using metrics appropriate to its respective functional focus and level of linguistic granularity.

4.5.1 | Word-Level Evaluation

At the word level, the model's performance is evaluated using a combination of detection and accuracy-based metrics that are widely employed in spelling correction research.

To assess the fine-tuned model's correction performance, the underlying counts of True Positives (TPs), False Positives (FPs), False Negatives (FNs), and True Negatives (TNs) are first obtained, where TP denote correctly corrected errors, FP indicate unnecessary or incorrect edits, FN represent undetected errors that remain uncorrected, and TN correspond to correctly unchanged inputs. These values form the basis for the computation of precision, recall, and the F1-score, which are used to measure how accurately the system identifies and modifies individual spelling errors. In this context, precision represents the proportion of changes proposed by the model that are valid, reflecting how carefully the system decides when to make an edit and how well it avoids introducing unnecessary or incorrect changes. Recall measures the proportion of actual errors in the dataset that the model successfully detects and modifies, showing how effectively it identifies existing mistakes. The F1-score provides the harmonic mean of precision and recall, offering a balanced view of the model's performance by combining its accuracy in making correct edits with its ability to capture all relevant errors. In addition, the $F_{0.5}$-score is calculated to place greater emphasis on precision, providing insight into how reliably the model avoids incorrect or unnecessary corrections, which is an important consideration for user-facing spell checkers, where reliability and trustworthiness are valued over extensive error coverage (Bijoy et al., 2025; Hládek et al., 2020; Näther, 2020).

To capture orthographic accuracy more directly, Character Error Rate (CER) is employed alongside these measures. CER quantifies the proportion of incorrect characters between the model's prediction and the gold-standard reference, providing a fine-

grained assessment of spelling quality. Its inclusion follows established practice in recent spelling and text-correction studies, which emphasise CER's sensitivity to subword deviations and minor orthographic inconsistencies often overlooked by word-level metrics. Furthermore, exact-match accuracy is calculated to represent the percentage of words corrected perfectly, offering a direct measure of the model's ability to generate fully accurate corrections (Niranjan et al., 2020; Salhab & Abu-Khzam, 2024).

4.5.2 | Sentence-Level Evaluation

The sentence-level model is evaluated using a set of complementary metrics designed to assess both the accuracy of the corrections and their contextual appropriateness within the sentence. This approach reflects the broader goal of sentence-level correction, which aims not only to fix spelling errors but also to ensure that each correction preserves the original meaning, grammatical integrity, and overall sentence clarity.

To assess the model's correction performance, the underlying counts of True Positives (TPs), False Positives (FPs), False Negatives (FNs), and True Negatives (TNs) are first obtained. At the sentence level, these represent sentences that were correctly corrected (TP), unnecessarily altered (FP), left uncorrected despite containing errors (FN), or correctly left unchanged (TN). These values form the basis for the computation of precision, recall, and the F-scores, which quantify how accurately the model identifies and applies corrections across complete sentences. The F1-score provides a balanced measure of overall correction accuracy, while the $F_{0.5}$-score places greater emphasis on precision, offering insight into the model's reliability and its ability to avoid unnecessary or incorrect changes (Arnardóttir et al., 2024).

To further evaluate contextual accuracy and how well corrections integrate into the surrounding text, several automatic metrics are employed, including Error Annotation Toolkit (ERRANT) (Bryant et al., 2017); Bilingual Evaluation Understudy (BLEU) (Papineni et al., 2002); Grammar Language Evaluation Understudy (GLEU) (Napoles et al., 2015); Word Error Rate (WER); and CER.

Together, these metrics form a comprehensive framework for evaluating sentence-level correction. Precision, recall, and F-scores provide interpretable measures of correction accuracy, while ERRANT, BLEU, and GLEU capture contextual and grammatical accuracy. In parallel, WER and CER assess surface-level form accuracy by quantifying deviations at the word and character levels (Goto et al., 2025; Keita et al., 2024; Niranjan et al., 2020; Phyu et al., 2024; Salhab & Abu-Khzam, 2024). This ensures that the model is evaluated both for its linguistic correctness and for the precision of its generated text.

4.6 | Limitations and Ethical Considerations

The development of the Maltese spell checker is subject to several constraints, primarily related to data availability, computational resources, and ethical considerations. Each of these factors presents challenges that will be addressed through targeted mitigation strategies.

4.6.1 | Data Constraints

A significant limitation is the lack of a publicly available dataset containing pairs of incorrect and corrected Maltese text. To address this, data augmentation techniques will be applied to generate synthetic spelling errors from correctly written Maltese text. This approach has been widely adopted in low-resource settings to simulate realistic error patterns and increase training data diversity (Marier et al., 2025). Source texts will be collected from verified and reliable Maltese corpora, and all generated data will be reviewed to maintain linguistic integrity and representativeness.

However, a limitation of this approach is that since the majority of the training data consists of synthetic errors, the model may still struggle to generalise effectively to real-world errors.

4.6.2 | Technical Constraints

Fine-tuning LLMs can be computationally intensive, often requiring GPUs with memory capacities beyond those available in consumer hardware. To address this, a dedicated server provided by the University of Malta and a cloud-based GPU solution, RunPod[9], were used to access the required computational resources on demand. Further details about the hardware are provided in Section 4.7.

4.6.3 | Time Constraints

Due to the time constraints associated with the research, it was not feasible to explore a broader range of fine-tuning configurations. The experiments were therefore restricted to a small number of parameter settings, which limits the extent to which the impact of alternative configurations could be evaluated. In particular, adjustments to LoRA hyperparameters, including the rank, the scaling factor and the dropout rate, were not investigated. These parameters can influence the expressive capacity of the adapter layers and may lead to different levels of model training complexity.

[9] `https://www.runpod.io/`

Similarly, other training components, including alternative learning-rate scheduler types, were not examined. The choice of scheduler can affect convergence behaviour and stability, and exploring different schedules could have yielded further insights into optimal training dynamics for the model.

Finally, the study did not incorporate multiple random seeds. Running experiments with varying seeds is important for assessing the robustness and reproducibility of results, as it helps distinguish between outcomes driven by genuine configuration effects and those arising from stochastic variation during training.

Collectively, these omissions constrain the breadth of the methodological investigation and mean that the reported results reflect only one plausible configuration pathway within the wider fine-tuning design space.

4.6.4 | Ethical Considerations

Ethical compliance is a key aspect of the project, particularly in relation to data collection and usage. Care will be taken from the begining to ensure that no sensitive, personal, or confidential information is collected or disclosed. All text data will originate from publicly available or openly licensed sources, and, where applicable, permissions were obtained to confirm that the material can be used for research purposes. Permissions can be reviewed in Appendix G.

4.7 | Hardware

Model fine-tuning was conducted on a dedicated Ubuntu-based, Command-Line Interface (CLI) server provided by the University of Malta. Both the word-level and sentence-level models were fine-tuned on an NVIDIA L40S GPU. This GPU is a high-performance data centre GPU based on the Ada Lovelace architecture. Designed for large-scale Artificial Intelligence (AI) workloads, the L40S features 48 Gigabyte (GB) of Graphics Double Data Rate 6 (GDDR6) memory, making it particularly well suited for fine-tuning large language models (NVIDIA Corporation, 2023). The fine-tuning processes were executed within Docker[10] containers to ensure server safety and preserve the fine-tuning environment. This approach also simplified dependency management and maintained reproducibility across experimental runs.

Inference and testing were carried out on RunPod a cloud-based GPU platform that provides access to a range of GPUs. The GPU used for inference and testing was the

[10] `https://www.docker.com/`

same model employed during fine-tuning, an NVIDIA L40S GPU, which was rented at a rate of $0.87 per hour. RunPod was selected primarily for its accessibility and simplified file management. Unlike a traditional Linux-based server that requires remote connection and manual file transfers, RunPod offers a direct web interface for managing, uploading, and retrieving files. This facilitated faster setup and more efficient handling of experimental data and model outputs.

5

Results and Evaluation

This chapter presents the evaluation of the proposed Maltese spell checker and reports the results obtained across the different experimental settings. The purpose of this chapter is to assess how effectively the fine-tuned model fulfils the research aim of creating a reliable Maltese spell checker by finetuning an open-source Large Language Model (LLM), specifically Meta's Large Language Model Meta AI (LLaMA) model, using a custom Maltese dataset. To do so, a series of evaluations were carried out on multiple test sets designed to examine performance at the word and sentence levels, and across both synthetic and real-world error data.

The chapter follows the sequence in which the experiments were conducted. Section 5.1 describes the evaluation setup, including the datasets used, the metrics applied, and the procedures followed to generate and analyse model predictions. Section 5.2 outlines the experimental design, explaining why each experiment was conducted and how the sequence of experiments was structured. Section 5.3 then presents the word-level results and sentence-level results. Finally, Section 5.4 provides a brief summary of the main findings, which are discussed in depth in Chapter 6.

5.1 | Evaluation Setup

This section outlines the setup used to test the fine-tuned LLM in its role as a Maltese spell checker. It introduces the datasets used in the evaluation, the metrics applied to measure performance, and the overall structure of the testing process. Together, these elements establish the foundation for the results presented in the following sections.

5.1.1 | Datasets

The datasets used for testing the word-level and sentence-level models were derived from the resources described in Section 4.3.4, with only the portions relevant for testing

included in this process. To ensure that testing remained independent from the training procedure, a held-out subset of each dataset was sampled exclusively for this purpose.

5.1.1.1 | Word-Level Test Sets

The word-level test set consisted of 10% of the word dataset described in Table 4.6. This subset was selected after the prompt creation process and amounted to 191,063 entries. The dataset consisted primarily of synthetically generated spelling errors, with each entry paired with its corresponding correct form. It was sampled using a fixed random seed and was fully excluded from the training process to ensure independence between training and testing. This test set was used to evaluate the model's ability to correct isolated words containing spelling mistakes.

Furthermore, 1,000 correctly written words were randomly selected from this 191,063-entry test set. These 1,000 words were used to examine how the model behaves when presented with error-free words, enabling the evaluation to capture any tendencies towards unnecessary modification or overcorrection.

5.1.1.2 | Sentence-Level Test Sets

The sentence-level test set consisted of 10% of the sentence dataset described in Table 4.7. This subset was selected after the prompt creation process and amounted to 19,211 entries. It was sampled using a fixed random seed and was fully excluded from the training process to ensure independence between training and testing. Each entry contains one or more synthetically generated errors paired with its corresponding corrected version. This test set was used to evaluate the model's ability to correct spelling mistakes within full-sentence contexts, where grammatical, syntactic, and semantic information may influence the expected correction.

In addition to the synthetic subset, a real-world sentence-level test set was prepared. This consisted of 50% of the collected real-world errors described in Section 4.3.1. Table 4.8 shows the sources and the items included in this dataset. These entries contain spelling errors produced by humans during a transcription exercise, as well as mistakes that occurred in genuine real-world scenarios. This test set was used to examine the model's ability to generalise beyond synthetic error patterns and to evaluate how it performs on authentic error data.

Furthermore, 1,000 correctly written sentences were randomly selected from the data sources presented in Table 4.8. These 1,000 sentences were used to examine how the model behaves when presented with error-free text, allowing the evaluation to capture any tendencies towards unnecessary modification or overcorrection.

5.1.2 | Evaluation Metrics

A set of evaluation metrics, as outlined in Section 4.5, was used to assess the performance of the fine-tuned model, and these can be grouped into two categories. The first category comprises correction-outcome metrics, which were applied to both the word-level and sentence-level evaluations. These metrics assess whether the model applied the correct correction to each input by assigning predictions to the classes True Positive (TP), False Positive (FP), False Negative (FN), and True Negative (TN). Based on these counts, several aggregate measures were derived, including precision, recall, the F1-score, and the F0.5-score. When correct inputs were used at either the word or sentence level, preservation accuracy and overcorrection rate were also calculated in order to quantify the model's behaviour when presented with error-free text.

The second group consists of sequence-level similarity and error metrics, which evaluate how closely the model's output resembles the reference correction at the character, and word level. These metrics include Character Error Rate (CER), which was used for both the word-level and sentence-level evaluations, and a set of sentence-level metrics, namely Word Error Rate (WER), the edit-based Error Annotation Toolkit (ERRANT) metric and n-gram overlap measures such as Bilingual Evaluation Understudy (BLEU) and Grammar Language Evaluation Understudy (GLEU). Together, these sequence-level metrics complement the correction-outcome metrics by capturing graded forms of similarity and deviation that may not be reflected in exact-match measures.

The following sections describe each metric in detail. Section 5.1.2.1 explains how TP, FP, FN, and TN were determined and how accuracy, precision, recall, the F1-score, the F0.5-score, preservation accuracy, and overcorrection rate were calculated. Section 5.1.2.2 then outlines the sequence-level metrics, describing the computation of CER, which was applied at both word and sentence level models, as well as the sentence-level metrics WER, BLEU, GLEU and ERRANT.

5.1.2.1 | Correction-Outcome Metrics

For each test item, the model output was evaluated by comparing it with both the original input and the reference correction. This comparison determined whether the model handled erroneous and error-free inputs appropriately. When the input contained an error, the evaluation assessed whether the model corrected it accurately, corrected it incorrectly, or failed to correct it. When the input was already correct, the evaluation assessed whether the model preserved the original form or introduced an unnecessary modification. Based on these conditions, each prediction was assigned to one of four

categories: TP, FP, FN, or TN. These categories form the basis for calculating precision, recall, the F1-score, the F0.5-score, preservation accuracy, and overcorrection rate.

The four possible correction outcomes are defined below, and illustrative examples of their application are provided in Table 5.1.

Possible correction outcomes:

- TP: The input contained an error and the model corrected it to the exact target form.
- FN: The input contained an error, but the model either failed to correct it or produced an incorrect correction.
- FP: The input was already correct, yet the model modified it unnecessarily.
- TN: The input was correct and the model correctly left it unchanged.

From these four categories, several performance measures were derived. Accuracy represents the proportion of predictions that match the reference correction and is calculated using Equation 5.1. Precision measures the share of items the system corrected that genuinely contained an error (Equation 5.2), while recall measures how many of the existing errors were successfully corrected (Equation 5.3). The F1-score, shown in Equation 5.4, is the harmonic mean of precision and recall. The F0.5-score, calculated using Equation 5.5, places greater weight on precision than recall and is therefore appropriate for spell checking tasks, where introducing incorrect corrections is less desirable than occasionally failing to correct an error (Abdelaal et al., 2024; Kobayashi et al., 2024; Ng et al., 2014).

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{5.1}$$

$$Precision = \frac{TP}{TP + FP} \tag{5.2}$$

$$Recall = \frac{TP}{TP + FN} \tag{5.3}$$

$$F_1 = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{5.4}$$

$$F_{0.5} = (1 + 0.5^2) \times \frac{Precision \times Recall}{0.5^2 \times Precision + Recall} \tag{5.5}$$

Table 5.1: Examples of correction outcome categories. Source: Author.

Case	Input	Target	Model Output	Explanation
TP	It-tifel **marret** l-iskola.	It-tifel **mar** l-iskola.	It-tifel **mar** l-iskola.	The input contained errors that were successfully identified and corrected by the model. This prediction is counted as TP.
FN	It-tifel **marret** l-iskola.	It-tifel **mar** l-iskola.	It-tifel **tmur** l-iskola.	The input contains an error, but the model fails to correct it. This prediction is counted as FN. The same category applies if the model does not provide a correction (e.g., *It-tifel **marret** l-iskola.*).
FP	It-tifel **mar** l-iskola.	It-tifel **mar** l-iskola.	It-tifel **tmur** l-iskola.	The input is already correct, but the model introduces an unnecessary change. This prediction is counted as FP.
TN	It-tifel **mar** l-iskola.	It-tifel **mar** l-iskola.	It-tifel **mar** l-iskola.	The input is correct and the model leaves it unchanged. This prediction is counted as TN.

Two additional metrics were used for evaluations involving correct inputs: preservation accuracy and overcorrection rate. Preservation accuracy quantifies how often the model correctly leaves error-free text unchanged, as shown in Equation 5.6. On the other hand, the overcorrection rate indicates how frequently the model unnecessarily alters correct inputs, as shown in Equation 5.7.

$$\textit{Preservation accuracy} = \frac{TN}{TN + FP} \tag{5.6}$$

$$\textit{Overcorrection rate} = \frac{FP}{TN + FP} \tag{5.7}$$

5.1.2.2 | Sequence-Level Metrics

Sequence-level metrics were used to assess how closely the model's output matched the reference correction at the levels of characters, words, and sentence structure. These metrics are particularly beneficial for sentence-level evaluation because they offer a more detailed view of the model's behaviour than exact-match measures. A sentence may not perfectly match the target but may still contain correct or partially correct segments, and such cases are not captured by accuracy-based metrics alone. Sequence-level metrics therefore allow the evaluation to recognise graded similarity between the model output and the reference, providing a more informative assessment of sentence-level performance.

Building on these motivations, several sequence-level metrics were applied in this study. CER was used for both the word-level and sentence-level evaluations, while the remaining metrics: WER, ERRANT, BLEU and GLEU, were used exclusively for sentence-level testing. Each metric captures a different aspect of similarity between the predicted and reference outputs, as described below.

CER quantifies the number of character-level edits, namely insertions, deletions, and substitutions, needed to transform the model's output into the target, normalised by the length of the target. This is computed using Equation 5.8, where S, D and I, represent the number of substitutions, deletions, and insertions respectively, and N is the number of characters in the target.

$$\textit{Error Rate} = \frac{S + D + I}{N} \tag{5.8}$$

WER applies the same formulation (Equation 5.8) at the word level. It measures the number of word-level substitutions, deletions, and insertions required to transform the predicted sentence into the reference. When WER is calculated, S, D and I now correspond to word-level edit operations, and N denotes the number of words in the reference sentence (Salhab & Abu-Khzam, 2024).

ERRANT was used as an edit-based evaluation metric tailored for Grammatical Error Correction (GEC). It aligns the predicted and target sentences, extracts edits, assigns edit types, and computes precision, recall, and an F0.5-score. This makes it possible to assess how well the model identified and corrected specific types of errors, rather than evaluating surface similarity alone. ERRANT uses the standard Fβ formulation shown in Equation 5.5, where precision and recall are computed over matched edits (Bryant et al., 2017). Importantly, the F0.5-score computed by ERRANT is conceptually different from the instance-level F0.5-score described in Section 5.1.2.1. ERRANT derives its score from edit-level precision and recall, whereas the earlier instance-level measure is

based on item-level outcomes. ERRANT therefore evaluates the quality and correctness of the edits produced, rather than the overall correctness of each prediction. Below is an example of how ERRANT is computed on an example sentence.

ERRANT computation example:

- Incorrect input: It-tifel marret l-iskola u kielet il-ħobż.
- Target: It-tifel mar l-iskola u kiel il-ħobż.
- Model output: It-tifel mar l-iskola u kielet il-ħobż.
- ERRANT computation:
  - Gold edits:
    1. "marret" → "mar"
    2. "kielet" → "kiel"
  - System edits:
    1. "marret" → "mar"
  - TP = 1 (correctly identified/corrected "marret → mar")
  - FN = 1 (missed "kielet → kiel")
  - FP = 0 (no incorrect changes introduced)
  - Precision = $1/(1+0) = 1.0$
  - Recall = $1/(1+1) = 0.5$
  - $F_{0.5} = 1.25 \times \frac{1.0 \times 0.5}{0.25 \times 1.0 + 0.5} = 0.833$

BLEU was included as an n-gram overlap metric to assess the surface-form similarity between the model's output and the target correction. It was computed using the SacreBLEU library (Post, 2018), which implements the standard formulation introduced by Papineni et al. (2002). BLEU measures how many n-grams in the model output also appear in the target by calculating modified n-gram precision, which limits the credit given for repeated words and prevents systems from artificially inflating their score by repeating common terms. In addition to this precision component, BLEU includes a brevity penalty, shown in Equation 5.10, which reduces the score when the model output is shorter than the reference. This prevents overly short hypotheses from receiving disproportionately high precision scores. The full BLEU formulation is given in Equation 5.9, where p_n denotes the modified precision for each n-gram order and w_n the

corresponding weight. In Equation 5.10, c represents the length of the candidate output and r the length of the target (Papineni et al., 2002). In the context of spell checking, BLEU provides an indication of how closely local word sequences in the model output align with those in the target. However, although the brevity penalty alleviates this effect, BLEU remains more sensitive to precision than recall, meaning that shorter outputs containing several matching n-grams may still receive relatively high scores. For this reason, BLEU is used alongside additional metrics in this study to provide a more balanced assessment of model performance.

$$BP \times \exp\left(\sum_{n=1}^{N} w_n \log p_n\right) \tag{5.9}$$

$$BP = \begin{cases} 1, & c > r \\ e^{1-(r/c)}, & c \leq r \end{cases} \tag{5.10}$$

Following the use of BLEU, GLEU was included as a complementary n-gram based similarity metric. It was computed using the implementation provided in the Natural Language Toolkit (NLTK) Python library (Bird et al., 2009), which follows the formulation described by Wu et al. (2016). Like BLEU, GLEU evaluates surface-level overlap between the model output and the target, but it does so by considering both n-gram precision and n-gram recall. For each sentence, all 1-to-4-gram subsequences are extracted from both texts, and the number of matching n-grams is used to compute recall, defined as the proportion of target n-grams found in the candidate, and precision, defined as the proportion of candidate n-grams found in the target. For each n-gram order, the GLEU value is taken as the minimum of these two quantities. The NLTK implementation then produces a corpus-level score by aggregating all matching n-grams and all total n-grams across the entire test dataset, computing precision and recall from these corpus-level counts, and taking the arithmetic mean of the four order-specific values. This approach yields a single score for the entire corpus and provides a balanced assessment of surface similarity that avoids disproportionately rewarding very short or very long outputs.

5.1.3 | Evaluation Procedure

This section outlines the procedure followed to evaluate the performance of both models across the different test settings. The aim was to ensure a consistent and reproducible workflow that enabled direct comparison between the word-level and sentence-level approaches.

The evaluation procedure involved feeding the test inputs to each model and comparing the resulting output with the corresponding target. For the word-level model, each token from the test sets described in Section 5.1.1.1 was processed independently, whereas the sentence-level model operated on complete sentences as described in Section 5.1.1.2.

For each model, the items in the respective test sets were formatted using the appropriate prompt template, with the output field left blank to allow the model to generate its prediction. Inputs were provided to the models one at a time. All predictions were generated using a temperature of 0.5 and a top-p value of 0.9.

Model inference was carried out on RunPod[1] using an NVIDIA L40s Graphics Processing Unit (GPU), after which the resulting predictions were saved as Comma Seperated Values (CSV) files. These files were subsequently parsed in Python within Google Colab[2] to compute the evaluation metrics.

Once predictions had been collected, three conditions were used to determine whether the model identified and corrected an error: whether the input contained an error, whether the model made a change, and whether the model produced the correct output. These conditions formed the basis for classifying each instance as a TP, FP, FN, or TN. These classifications enabled the calculation of precision, recall, and F-scores, following the definitions provided in Section 5.1.2.1. These metrics were computed across all relevant evaluation settings, namely the word-level reserved test set, the word-level correct-input test set, the sentence-level reserved test set, the sentence-level real-world error test set, and the sentence-level correct-input test set.

Sequence-level metrics were also applied. For the word-level model, CER was computed using the jiwer[3] library on the outputs and targets from the reserved test set containing spelling errors. The sentence-level model used the same CER procedure and, owing to the longer sequences, additionally employed WER, BLEU, and GLEU, following the methods described in Section 5.1.2.2. These metrics were applied to the sentence-level reserved test set and the real-world error dataset.

Finally, the ERRANT toolkit was used to obtain span-based edit annotations for the sentence-level predictions. For each input, model output, and target sentence, ERRANT generated edit counts which were aggregated into TPs, FPs, FNs, or TNs, and the corresponding F0.5-score. ERRANT evaluation was performed on both the sentence-level reserved test set and the real-world error dataset.

This systematic workflow ensured that evaluation remained fair, reproducible, and

[1] https://www.runpod.io/
[2] https://colab.research.google.com/
[3] https://pypi.org/project/jiwer/

consistent across all test settings, thereby providing a reliable basis for comparing the performance of both models under controlled and realistic conditions.

5.2 | Experimental Design

This section outlines the rationale behind the experimental settings adopted in this study. Whereas Section 5.1 describes how the evaluation was carried out, this section focuses on why particular experiments were conducted and the reasoning behind their design. The experiments were structured to investigate the behaviour of both models under a range of conditions, from controlled synthetic errors to real-world data and correctly written inputs.

5.2.1 | Word-Level Model Evaluation

The word-level evaluation was conducted first, as this represented the simpler of the two correction tasks. Assessing performance at the token level provided an initial indication of how effectively the model could recognise and correct isolated spelling errors without the added complexity of sentence structure or grammar. This stage also provided a baseline for assessing how well the model handled the different word forms found in Maltese.

5.2.2 | Sentence-Level Model Evaluation

Following the word-level experiments, the sentence-level model was evaluated to investigate how the model handled corrections within full grammatical contexts. Unlike the word-level setting, sentence-level correction requires the model to maintain fluency, grammatical accuracy, and semantic coherence while correcting spelling errors. This experiment therefore assessed how well the fine-tuned model handled corrections that are context-dependent.

5.2.3 | Learning Rate Comparison

Two experiments were carried out to compare two different learning rates found in the literature which are 2×10^4 and 2×10^5. The learning rate of 2×10^4 was selected as the primary configuration, as it was reported as an effective starting point for fine-tuning LLMs (Parthasarathy et al., 2024). The lower learning rate of 2×10^5 was also evaluated based on the findings of Pareja et al. (2024), who noted it to be a suitable optimisation

setting for fine-tuning smaller LLMs. This comparison was included to test whether a more conservative optimisation setting would improve stability or generalisation. The learning rate comparison was performed for both the word-level and sentence-level models.

5.2.4 | Evaluation on Real-World Errors

To assess how well the sentence-level model generalised beyond synthetic data, an additional experiment was conducted using a dataset containing real-world spelling errors. This test served two purposes. The first was to evaluate whether performance persisted when encountering naturally occurring errors that were not represented in the synthetic training data. The second was to identify any performance degradation and understand the types of real-world errors the model struggled to correct.

This evaluation was performed only at the sentence level, as no equivalent word-level dataset of real-world errors was available. A word-level dataset could not be derived from the sentence-level real-world data because some sentences contained real-word errors, where all words were valid Maltese words but used incorrectly in context. Additionally, in many cases the number of words in the incorrect sentence did not match the number of words in the corrected sentence, since some corrections required adding or removing words to produce a grammatically and contextually correct form. As a result, word-level pairs could not be generated automatically and would have required manual review, which was not feasible due to time constraints.

5.2.5 | Evaluation on Correct Inputs

A further experiment examined how both models behaved when presented with correctly written inputs. This was done to measure the extent of overcorrection, which occurs when a system makes unnecessary changes to text that is already correct. Since the reserved test sets were mainly composed of erroneous inputs, a separate sample of 1,000 correctly written words and sentences was derived at random from the unseen test datasets. This allowed the stability of both the word-level and sentence-level models to be assessed.

5.2.6 | Summary

Overall, the experimental design progressed from simpler to more complex evaluation settings. Starting with word-level correction enabled baseline assessment, while the subsequent sentence-level experiments explored context-dependent performance. The

learning rate comparison examined the effect of optimisation parameters, and the evaluations on real-world and correct-input datasets provided insight into model generalisation and reliability. Together, these experiments formed a coherent framework for analysing the strengths and limitations of the proposed spell checking models.

5.3 | Results

This section presents the results obtained from evaluating both the word-level and sentence-level models across the different test settings. For the word-level model, results are reported for the initial configuration and the empirical learning-rate comparison on both the reserved test set and the correct-input dataset. The sentence-level model is evaluated in the same way, with an additional evaluation on the real-world error dataset to examine its performance on naturally occurring spelling errors. The following sections (Section 5.3.1 and Section 5.3.2) present the results in a descriptive manner, as a detailed discussion will be presented in Chapter 6.

5.3.1 | Word-Level Model Results

Table 5.2 presents the results obtained by the model using the configurations specified in Table 4.9. The evaluation was carried out on the reserved 10% portion of the dataset described in Table 4.6, which consists primarily of incorrect-correct word pairs, as well as on an additional set of 1,000 randomly selected correct words from unseen test data, which were used as inputs to evaluate overcorrection. The notation Not Applicable (N/A) indicates that the corresponding metric was not calculated because it is not relevant or applicable to the evaluation setting for that test set.

Table 5.3 presents the results obtained by the empirical model. This model uses the configurations listed in Table 4.9, with the exception of the learning rate, which was reduced to 2×10^{-5}. The evaluation was performed on the same word-level test datasets that were used to produce the initial model results reported in Table 5.2. The notation N/A indicates that the corresponding metric was not computed because it is not relevant or applicable to the evaluation setting for that particular test set.

5.3.2 | Sentence-Level Model Results

Table 5.4 presents the results obtained by the sentence-level model using the configurations specified in Table 4.10. The evaluation was carried out on the reserved 10% of the sentence-level dataset described in Table 4.7, as well as on the reserved real-world

Table 5.2: Word-level model results using configurations outlined in Table 4.9. Source: Author.

Metric	**Result on 10% incorrect-correct pairs**	**Result on correct inputs**
TP	161,719	0
FP	0	668
FN	29,344	0
TN	0	332
Precision	1.0	N/A
Recall	0.846	N/A
F1-score	0.917	N/A
F0.5-score	0.965	N/A
Accuracy	84.64%	N/A
CER	2.79%	N/A
Preservation accuracy	N/A	33.2%
Overcorrection rate	N/A	66.8%

Note. N/A indicates that the metric is not applicable to a given test set and was therefore not computed.

error dataset described in Table 4.8. As indicated in Table 4.8, the real-world datasets were drawn from the studies of Debattista (2022) and Busuttil (2025). Specifically, 2,131 entries from Debattista (2022) and 1,294 entries from Busuttil (2025) were used as unseen test sets to evaluate the model's reliability in correcting naturally occurring human spelling errors, including rare or unconventional mistakes that could not be captured by the synthetic data augmentation process, which was based on linguistic studies of the most common error patterns. This makes real-world evaluation essential for assessing the model's performance on unexpected errors. In addition, an extra set of 1,000 correctly written sentences was drawn at random from unseen real-world test data to assess the model's tendency to overcorrect. The notation Not Applicable (N/A) indicates that the corresponding metric was not computed because it is not relevant or applicable

Table 5.3: Word-level model results using the empirical configuration with a reduced learning rate. Source: Author.

Metric	Result on 10% incorrect-correct pairs	Result on correct inputs
TP	113,877	0
FP	0	538
FN	77,186	0
TN	0	462
Precision	1.0	N/A
Recall	0.596	N/A
F1-score	0.747	N/A
F0.5-score	0.881	N/A
Accuracy	59.6%	N/A
CER	8.19%	N/A
Preservation accuracy	N/A	46.2%
Overcorrection rate	N/A	53.8%

Note. N/A indicates that the metric is not applicable to a given test set and was therefore not computed.

to the evaluation setting for that particular test set.

Table 5.5 presents the results obtained by the empirical sentence-level model. This model follows the configurations outlined in Table 4.10, with the exception of the learning rate, which was reduced to 2×10^{-5}. The evaluation was carried out on the same sentence-level test datasets used to produce the initial results reported in Table 5.4.

5.4 | Summary of Findings

This section summarises the main results from the word-level and sentence-level evaluations. Its purpose is to highlight the key performance patterns observed across the different learning-rate configurations and test settings. A detailed interpretation of these

findings is provided in Chapter 6.

For the word-level model, the configuration using the learning rate of 2×10^{-4} achieved the strongest performance on the test set containing incorrect-correct pairs, which consisted primarily of synthetically generated spelling errors. It produced higher recall, higher F-scores, better accuracy, and a lower CER than the model trained with the lower learning rate of 2×10^{-5}. When tested on correctly written inputs, both configurations introduced unnecessary changes, although the empirical model made fewer.

For the sentence-level model, the configuration trained with the learning rate of 2×10^{-4} again outperformed the version trained with 2×10^{-5}. It achieved higher recall and F-scores, and a substantially higher accuracy. It also showed better performance across all sequence-level metrics. In particular, it obtained higher BLEU and GLEU scores, together with lower CER and WER values, indicating stronger sentence-level correction quality. The initial configuration also achieved higher ERRANT precision, recall, and F0.5-score. Performance declined for both configurations when evaluated on the real-world error dataset, although the initial configuration continued to produce noticeably better results. On the correct-input dataset, both versions introduced unnecessary edits, with the empirical configuration showing a higher overcorrection rate.

Across both word-level and sentence-level tasks, the model trained with the learning rate of 2×10^{-4} consistently achieved the best overall results. These best-performing models correspond to the configurations listed in Table 4.9 for the word-level model and Table 4.10 for the sentence-level model. The discussion in Chapter 6 therefore focuses on these configurations, examining their behaviour across the different evaluation settings.

Table 5.4: Sentence-level model results using configurations outlined in Table 4.10. Source: Author.

Metric	Result on 10% incorrect-correct pairs	Result on real-world errors	Result on correct inputs
TP	18,298	957	0
FP	6	70	538
FN	891	2,466	0
TN	16	110	462
Precision	0.99	0.93	N/A
Recall	0.95	0.28	N/A
F1-score	0.97	0.43	N/A
F0.5-score	0.99	0.64	N/A
Accuracy	95.3%	29.6%	N/A
ERRANT precision	0.98	0.72	N/A
ERRANT recall	0.98	0.71	N/A
ERRANT F0.5-score	0.98	0.72	N/A
BLEU	99.21	79.27	N/A
GLEU	0.99	0.78	N/A
CER	0.46%	4.82%	N/A
WER	0.73%	13.03%	N/A
Preservation accuracy	N/A	N/A	46.2%
Overcorrection rate	N/A	N/A	53.8%

Note. N/A indicates that the metric is not applicable to a given test set and was therefore not computed.

Table 5.5: Sentence-level model results using the empirical configuration with a reduced learning rate. Source: Author.

Metric	**Result on 10% incorrect-correct pairs**	**Result on real-world errors**	**Result on correct inputs**
TP	12,372	476	0
FP	11	67	622
FN	6,817	2,947	0
TN	11	113	378
Precision	0.99	0.88	N/A
Recall	0.64	0.14	N/A
F1-score	0.78	0.24	N/A
F0.5-score	0.90	0.43	N/A
Accuracy	64.46%	16.35%	N/A
ERRANT precision	0.90	0.02	N/A
ERRANT recall	0.88	0.04	N/A
ERRANT F0.5-score	0.90	0.03	N/A
BLEU	95.20	69.68	N/A
GLEU	0.94	0.69	N/A
CER	1.32%	7.25%	N/A
WER	3.16%	18.61%	N/A
Preservation accuracy	N/A	N/A	37.8%
Overcorrection rate	N/A	N/A	62.2%

Note. N/A indicates that the metric is not applicable to a given test set and was therefore not computed.

6

Discussion

This chapter discusses the results presented in Chapter 5 and explains what worked well in the performance of the word-level and sentence-level models, what did not work as expected, and why these outcomes are likely to have occurred. Since the discussion focuses on the best-performing configuration for each model, the different test datasets provide the main source of variation in the analysis. Each dataset offers a distinct view of how the models behave under different conditions, allowing the chapter to examine their strengths and limitations from several perspectives. The discussion relates these behaviours to the characteristics of the training data, the fine-tuning decisions made, and the broader challenges of developing a spell checker for a low-resourced language.

Overall, the word-level and sentence-level models trained with the learning rate of 2×10^{-4} performed better those trained with 2×10^{-5}, and the discussion therefore focuses on the models trained with this higher learning rate. On the incorrect-correct pairs, both models achieved higher recall and F-scores, together with lower Character Error Rate (CER). The sentence-level model also achieved lower Word Error Rate (WER) and stronger ERRANT performance, as well as higher n-gram overlap scores (Bilingual Evaluation Understudy (BLEU) and Grammar Language Evaluation Understudy (GLEU)). However, its performance dropped on the real-world human error dataset, suggesting that the model struggled to generalise beyond the synthetic spelling errors used during training. In addition, both models introduced unnecessary edits when processing correctly written inputs. This suggests that the models sometimes applied correction patterns too broadly and attempted to modify text even when no error was present.

The remainder of this chapter examines these behaviours in detail. Section 6.1 discusses the word-level model, focusing on its performance on incorrect-correct pairs and its tendency to overcorrect on correct inputs. Section 6.2 provides the corresponding interpretation for the sentence-level model, considering its behaviour on incorrect-correct pairs, its challenges with real-world human errors, and its overcorrection tendencies. This is followed by Section 6.3, which compares the models developed in this study

with existing Maltese spell checking tools and previous research. Section 6.4 then reflects on the extent to which the methodology used in this dissertation may be applied to other low-resource languages. Finally, Section 6.5 summarises the main points of the chapter.

6.1 | Word-Level Model Discussion

The following section examines the behaviour of the best-performing word-level model, as determined by the evaluation metrics reported in Chapter 5. The analysis focuses on two aspects of the model's performance. The first is its ability to correct the incorrect-correct pairs, which consist mainly of synthetic spelling errors, and the second is its stability when handling correctly written inputs. Together, these perspectives highlight both the strengths and the limitations of the word-level approach, offering insight into the types of errors the model corrects reliably and the situations in which it struggles or produces unnecessary changes.

6.1.1 | Performance on Incorrect-Correct Pairs

The word-level model trained using the configurations specified in Table 4.9 performed strongly on the incorrect-correct pairs in the test dataset, which consisted mainly of synthetically generated spelling errors. The model achieved an accuracy of 84.64%. Although the test set was created using the same synthetic process as the training data, the individual items were unseen during fine-tuning. As a result, this accuracy indicates that the model learned the types of spelling errors represented in the training data effectively and that synthetic error generation provided a useful foundation for capturing common misspellings at word-level.

The model's high recall and F-scores show that it successfully corrected most of the misspelled tokens. Its low CER suggests that the remaining errors were typically minor deviations rather than completely incorrect replacements. Furthermore, the precision score of 1.0 (perfect score) indicates that every attempted correction was correct and that the model did not introduce any unnecessary edits. This outcome is expected because the test set contained only misspelled inputs, meaning that any modification the model made corresponded to a genuine correction and overcorrection was not possible. To address the fact that this test set included only spelling errors as inputs, an additional sample of 1,000 correctly written words was derived as described in Section 5.1.1.1, and the results of this evaluation are discussed in Section 6.1.2. Examples of model outputs that correctly matched the expected target word are shown in Table 6.1.

Table 6.1: Examples of incorrect inputs where the model produced the correct target word. Source: Author.

Input	Model Output	Target
nikkorrispontu	nikkorrispondu	nikkorrispondu
ma thazzqux	ma tħażżqux	ma tħażżqux
ja ninfaqax	ma ninfaqax	ma ninfaqax
tistokja	tistokkja	tistokkja
essensjali	essenzjali	essenzjali

However, several limitations also became evident. The recall score of 0.846 shows that the model did not correct all spelling errors. In total, it left 2,393 of the 191,063 incorrect inputs unchanged. This indicates that the model occasionally failed to detect an error or could not determine an appropriate correction, even when a spelling mistake was present. Examples of such cases are presented in Table 6.2. In addition to these missed corrections, the model also produced incorrect outputs that were not valid Maltese words, and examples of these invalid predictions are shown in Table 6.3.

Table 6.2: Examples of misspelled inputs that the model left unchanged. Source: Author.

Input	Model Output	Target
mieqaf	mieqaf	nieqaf
ntenneb	ntenneb	ndenneb
jinbartu	jinbartu	jinbardu
xxehru	xxehru	xehru
ma tifisniex	ma tifisniex	ma ttifisniex

A further limitation emerged when examining the cases where the prediction did not match the expected target. Out of the 29,344 mismatches, approximately 5,600 were still valid Maltese words. These were identified by comparing the model's outputs against a list of 169,618 correct Maltese word forms (lexicon) available for this study, although

Table 6.3: Examples of invalid model outputs. Source: Author.

Input	Model Output	Target
nieqża	nieqsah	nieqsa
nittret	niddret	pittret
jtma	jidma	jitma
perikolusi	perikoliżi	perikolużi
ma jagħfiċx	ma jagħfidx	ma jagħfiġx

the true number may be higher because the lexicon does not necessarily cover every valid word in the language. These outputs were not spelling mistakes but alternative inflections or orthographically acceptable variants. In these situations, a strict word-level evaluation penalises the model even when the predicted word is linguistically valid. This challenge is especially relevant for Maltese because the language has rich morphology that allows many possible surface forms which depend on context. Since the word-level model evaluates each token independently, it may output a linguistically valid form that differs from the target. Examples of such cases are presented in Table 6.4.

Table 6.4: Examples of valid Maltese words from model output that did not match the expected target. Source: Author.

Input	Model Output	Target
nimfluwenza	influwenza	ninfluwenza
riet	ried	rmiet
knaħħal	nkaħħal	jnaħħal
ġra	ġera	ġara
ma temmux	ma demmux	ma themmux

This limitation also reflects a constraint in the design of the word-level model developed in this study. The fine-tuning process required the model to produce a single correct output for each input. Allowing the model to generate multiple candidate corrections would be more flexible and user-friendly, as users could select the form they

need. It would also reduce the likelihood of penalising the model for producing a valid Maltese word that differs from the expected target, since the correct form would be more likely to appear among the generated options.

Overall, these findings suggest that the model corrects many spelling errors reliably, but its single-target design imposes constraints on performance. In addition, the absence of real-world error data meant that the model could not be evaluated under authentic conditions, leaving its behaviour in practical usage scenarios unresolved.

6.1.2 | Performance on Correct Inputs

The evaluation on correctly written words provides insight into how reliably the word-level model preserves error-free text, which is a crucial requirement for a practical spell checker. Unlike the incorrect-correct pairs, this dataset contained no spelling errors, so the expected behaviour is for the model to leave all inputs unchanged. However, the model modified 668 out of the 1,000 correctly written words, resulting in an overcorrection rate of 66.8%. This means that in more than two-thirds of the cases, the model attempted to fix words that were already correct. This behaviour highlights a key limitation of the word-level approach and may interfere with user workflow, as inaccuracies in the model's output can reduce users' trust in its suggestions. Examples of the model's performance on error-free inputs are presented in Table 6.5. Many of these overcorrections involve minor orthographic changes or shifts to alternative inflected forms, but they nevertheless represent undesirable behaviour in a spell checker.

Table 6.5: Examples of correct inputs that the model modified unnecessarily. Source: Author.

Input	Model Output	Target
ma insulentax	ma insulentaxx	ma insulentax
nibgħat	nibggħad	nibgħat
imxandra	imxandrah	imxandra
ma tittamawx	ma tiddamawx	ma tittamawx
ftiehmu	tfiehmu	ftiehmu

These outcomes indicate that the model struggles to distinguish between words that require correction and those that do not. This limitation arises from the fact that the

training data included very few cases in which a correct input was paired with an identical target, leaving the model with limited exposure to situations where no change is the appropriate response. Instead, it assumes that every input word is a misspelling that must be transformed. This limitation is expected from a design perspective, as the training data included very few correct input-target pairs, which meant the model had almost no opportunity to learn that some tokens should be preserved without modification.

From a practical standpoint, this overcorrection tendency poses a substantial challenge. A word-level spell checker must be conservative when making changes and should avoid altering content unless there is strong evidence of an error. The current model's behaviour suggests that additional training strategies could help mitigate this issue. Possible improvements include adding correct examples to the fine-tuning data so that the model learns that some inputs are already correct and should be left unchanged.

Taken together, the results on the correct-input dataset show that while the model is effective at correcting many types of spelling errors, its inability to reliably preserve correct words reduces its suitability as a standalone word-level spell checker.

6.2 | Sentence-Level Model Discussion

This section discusses the behaviour of the sentence-level model trained with the learning rate of 2×10^{-4}, which was identified in Chapter 5 as the best-performing configuration when compared to the other sentence-model trained on leanring rate of 2×10^{-5}. The analysis examines the model's performance across the three evaluation settings: the test dataset of incorrect-correct sentence pairs, the real-world error dataset, and the correct-input dataset. These settings provide complementary perspectives on how well the model generalises beyond the synthetic errors of the training data and how reliably it preserves or modifies sentences. The discussion highlights both the strengths and limitations observed in each case and offers explanations for why these behaviours occur.

6.2.1 | Performance on Incorrect-Correct Pairs

The sentence-level model trained using the configurations in Table 4.10 performed strongly on the incorrect-correct sentence pairs in the test dataset. Similar to the word-level setup, this dataset consisted mainly of synthetically generated spelling errors, but here the model also needed to maintain grammatical structure, fluency, and meaning while

correcting the input. The model achieved an accuracy of 95.3%, indicating that it was able to produce the correct corrected sentence in most cases, even though sentence-level correction introduces more complexity than isolated token correction.

The model also achieved a precision score of 0.99. Precision reflects the proportion of corrections made by the model that were correct, and such a high value indicates that the model introduced few incorrect edits. This score is understandable since the training data contained a large number of sentences with spelling errors, which could have biased the model toward assuming that most inputs should be changed. The incorrect-correct test dataset was also made up almost completely of inputs with spelling errors, which helps explain why precision was so high in this setting.

The recall score of 0.95 further indicates strong performance. Recall measures how many of the actual errors in the input the model successfully corrected. A recall of 0.95 shows that the model corrected almost all instances of spelling errors contained in this dataset. Together, the high precision and high recall demonstrate that the model not only corrected errors accurately but also corrected almost all of them.

The F1-score of 0.97 and the F0.5-score of 0.99 further demonstrate the model's strong and well-balanced performance. The F0.5-score places greater weight on precision, which is consistent with the model's tendency to apply correct edits while avoiding unnecessary changes.

Additional insights come from the sequence-level metrics. The model obtained a CER of 0.46% and a WER of 0.73% on the incorrect-correct pairs. These metrics measure the degree of difference between the model's output and the target when the two do not match. When the prediction differs from the target, CER and WER quantify how large that deviation is. The low values observed here therefore indicate that, in the cases where the model did not produce the expected target sentence, the differences were usually small. Most deviations involved minor character edits or single-word substitutions rather than substantial structural changes.

Furthermore, the high BLEU score of 99.21 and GLEU score of 0.99 demonstrate a near-perfect n-gram overlap with the target sentences, meaning the model's output closely followed the intended sentence structure and phrasing.

Finally, the ERRANT precision, recall and F0.5-score offer a more fine-grained assessment of edit quality. ERRANT checks both the correctness of an edit and whether it appears in the correct location. Each of the three ERRANT metrics reached 0.98, indicating that the model applied almost all required edits, rarely introduced incorrect edits, and consistently placed edits where the gold-standard correction required them. Together, these scores show that the model produced highly accurate and well-targeted corrections that closely followed the expected target sequence.

To support the interpretation of the quantitative results, Tables 6.6 and 6.7 provide illustrative examples from the incorrect-correct test set. Table 6.6 shows cases where the model produced the expected target sentence, supporting the strong performance indicated by the metrics. On the other hand, Table 6.7 shows examples where the output did not fully match the target, giving insight into the deviations observed. These examples complement the numerical findings and give a clearer picture of how the model behaves in practice.

Table 6.6: Examples of the sentence-level model's outputs on the incorrect-correct test dataset where its predictions matched the intended corrections. Source: Author.

Input	**Model Output**	**Target**
Firrigward talinfużjoni "te mediċinali veġetali talweraq tassalvja", ordnat ukoll lil SALUS sabiex tipprovdi ċerta informazzjoni.	Fir-rigward tal-infużjoni "te mediċinali veġetali tal-weraq tas-salvja", ordnat ukoll lil SALUS sabiex tipprovdi ċerta informazzjoni.	Fir-rigward tal-infużjoni "te mediċinali veġetali tal-weraq tas-salvja", ordnat ukoll lil SALUS sabiex tipprovdi ċerta informazzjoni.
Għax hi iżżomm mal-Franċiżi, ħej!	Għax hi żżomm mal-Franċiżi, ħej!	Għax hi żżomm mal-Franċiżi, ħej!
Għal ħafna żmien il-Malti kien jinkiteb balfabetti differenti.	Għal ħafna żmien il-Malti kien jinkiteb b'alfabetti differenti.	Għal ħafna żmien il-Malti kien jinkiteb b'alfabetti differenti.
Kll mn ken hmm dr megħu jmmira dn i-bejta.	Kull min kien hemm dar miegħu jammira din il-bejta.	Kull min kien hemm dar miegħu jammira din il-bejta.
Żorr, maledukat, ġġelled u ta' qattagħni.	Żorr, maledukat, ġellied u ta' qattagħni.	Żorr, maledukat, ġellied u ta' qattagħni
Ġisimna għandu bżonn ħafna ilma biex tiffunzjona tajjeb.	Ġisimna għandu bżonn ħafna ilma biex jiffunzjona tajjeb.	Ġisimna għandu bżonn ħafna ilma biex jiffunzjona tajjeb.

Overall, the sentence-level model performs well when the errors match the patterns seen during training, demonstrating strong control over both local edits and sentence-level structure. These results indicate that the model has learned the synthetic correction patterns to a high degree and that its performance is closely tied to the characteristics of the data on which it was trained. The implications of this reliance on synthetic error

Table 6.7: Examples of the sentence-level model's outputs on the incorrect-correct test dataset where its predictions did not match the intended corrections. Source: Author.

Input	Model Output	Target
Il-kapitali tar-Repubblika Ceka, Praga tinsab f'paniku hekk kif l-awtoritajiet qed jibzghu li d-dizastru bl-ilmijiet li hakem lill-pajjiz f'Awwissu ta' hdax -il sena ilu, jirrepeti ruhu.	Il-kapitali tar-Repubblika Ċeka, Praga tinsab f'paniku hekk kif l-awtoritajiet qed jibżgħu li d-**dizastru** bl-ilmijiet li ħakem lill-pajjiżf'Awwissu ta' ħdax-il sena ilu, jirrepeti ruħu.	Il-kapitali tar-Repubblika Ċeka, Praga tinsab f'paniku hekk kif l-awtoritajiet qed jibżgħu li d-diżastru bl-ilmijiet li ħakem lill-pajjiżf'Awwissu ta' ħdax-il sena ilu, jirrepeti ruħu.
Erba' mit-18-il każ.	**Erba'** mit-18-il każ.	Erbgħa mit-18-il każ.
Fjum l-Istrina bosta tfal kienu joħorġu fit-toroq jagħtu l-awguri għas-sena t-tajba lil min ikun għaddej.	**Fjum** l-Istrina bosta tfal kienu joħorġu fit-toroq jagħtu l-awguri għas-sena t-tajba lil min ikun għaddej.	F'jum l-Istrina bosta tfal kienu joħorġu fit-toroq jagħtu l-awguri għas-sena t-tajba lil min ikun għaddej.
Imma biex il-fruntiera tkun sigurà ma jistax ikun li tibqa' tigi kkontrollata minnhom.	Imma biex il-fruntiera tkun **sigura** ma jistax ikun li tibqa' tiġi kkontrollata minnhom	Imma biex il-fruntiera tkun sikura, ma jistax ikun li tibqa' tiġi kkontrollata minnhom.
U la titwemmen kulma tisma' sew jgħidulek: "Ja mazzun!"	U la **titwemmen** kulma tisma' sew jgħidulek: "Ja mazzun!"	U la temmen kulma tisma' sew jgħidulek: "Ja mazzun!"

Note. The **bold text** in the *Model Output* column highlights the parts where the model's prediction did not match the target.

patterns are examined further in the following sections (Section 6.2.2 and 6.2.3).

6.2.2 | Performance on Real-World Errors

The model's performance declined substantially when evaluated on the real-world error dataset. Unlike the incorrect-correct pairs discussed in Section 6.2.1, which consisted mainly of synthetically generated spelling errors, the real-world dataset contained naturally occurring mistakes produced by human writers. These errors were more diverse, less systematic than those created through controlled synthetic transformations.

The model achieved an accuracy of 29.6%, which is considerably lower than the 95.3% obtained on the synthetic incorrect-correct pairs. This means that fewer than one-third of the real-world sentences were corrected as expected. The large drop in accuracy indicates that the model struggled to provide the expected corrected output once the error patterns no longer matched the synthetic structure it had learned during fine-tuning.

The model's precision on real-world errors was 0.93. This value indicates that, when the model decided to apply a correction, it was generally correct in assuming that an error was present in the input. This relatively high precision is understandable because the training data contained many inputs with spelling errors, and the real-world dataset also consisted largely of misspelled sentences as inputs. As a result, the model may have developed a tendency during training to assume that most inputs required correction, which contributed to the high precision score.

In contrast, recall was considerably lower at 0.28. This indicates that only a small proportion of the model's outputs matched the expected corrected versions in the real-world dataset. A low recall score does not necessarily mean that every incorrect output was entirely wrong; many sentences contained multiple errors, and the model sometimes corrected only some of them. However, any missed or partially corrected errors still count as mismatches, which lowers the recall score and explains the substantial gap between the model's precision and recall. The F1-score of 0.43 and the F0.5-score of 0.64 further illustrate this imbalance between precision and recall.

The ERRANT metrics also show a clear reduction in performance. The ERRANT precision (0.72), recall (0.71), and F0.5-score (0.72) indicate that the model was able to apply some edits correctly but frequently missed others or placed them inaccurately. Since ERRANT awards partial credit when some edits are correct, these scores suggest that the model could still produce useful corrections in parts of the incorrectly written text. However, the overall decline compared with the synthetic test dataset confirms that real-world errors presented a greater challenge.

The CER and WER offer further insight into the model's behaviour on real-world data. The model achieved a CER of 4.82% and a WER of 13.03%, both noticeably higher than the values recorded on the incorrect-correct pairs, which were largely composed of synthetically generated errors. These scores indicate that, when the model's output did not match the target on real-world inputs, the differences tended to be larger and more substantial. Although these levels of divergence are not extreme, they can be sufficient enough to affect clarity, meaning, and overall reliability. Taken together, the higher CER and WER demonstrate that the model found it more difficult to produce outputs closely aligned with the expected corrections when the errors no longer followed the synthetic

patterns encountered during training.

Additional insight is provided by the n-gram overlap metrics. Across the full real-world test dataset, the model achieved a BLEU score of 79.27 and a GLEU score of 0.78. Although these values are lower than the near-perfect scores recorded on the synthetic incorrect-correct pairs, they still show that a substantial amount of n-gram overlap was preserved between the model outputs and the expected targets. This means that even when the model did not fully correct a sentence, many of its word or character sequences remained similar to the target. Compared with the larger declines observed in accuracy and recall, the BLEU and GLEU scores declined to a lesser extent. This suggests that while the model often failed to apply all necessary corrections, it still retained correct local phrasing and produced partial corrections rather than entirely incorrect sentences. These results are consistent with the ERRANT findings and further indicate that the model could produce outputs that were partially aligned with the expected corrections, despite the greater variation and unpredictability found in real-world errors.

To support the interpretation of these results, examples from the real-world test set are provided in Tables 6.8 and 6.9. Table 6.8 presents cases where the model produced outputs that matched the expected corrections. In contrast, Table 6.9 shows examples where the model's outputs did not align with the target. These examples illustrate different types of deviations: some outputs differ only slightly from the expected correction, while others diverge more substantially. Together, these qualitative examples provide additional insight into how the model behaves when handling naturally occurring errors.

Overall, the evaluation on real-world errors shows that the model's performance depends strongly on how closely the input resembles the synthetic patterns encountered during training. Although the model retained some useful behaviour, such as producing partial corrections, maintaining reasonable n-gram overlap as reflected in the BLEU and GLEU scores, and applying some edits correctly, it was unable to fully correct most sentences and often struggled with the more complex or irregular errors produced by human writers. The combination of low accuracy and recall, reduced ERRANT scores, and considerably higher CER and WER demonstrates that naturally occurring spelling errors were significantly more difficult for the model to handle. These results indicate that while the model learned effective correction strategies for controlled synthetic errors, its ability to generalise to real-world writing remains limited and would benefit from further refinement and exposure to more diverse linguistic data.

Table 6.8: Examples of real-world sentences where the model produced the expected corrected output. Source: Author.

Input	Model Output	Target
B'dan il-mod il-kazijiet ta' reazzjonijiet avversi li sehhew f'Malta kienu accessibli ghal kull studju Ewropew li rrikjeda l-investigazzjoni tar-riskju tal-prodott farmacewtiku kkoncernat.	B'dan il-mod il-każijiet ta' reazzjonijiet avversi li seħħew f'Malta kienu aċċessibbli għal kull studju Ewropew li rrikjeda l-investigazzjoni tar-riskju tal-prodott farmaċewtiku kkonċernat.	B'dan il-mod il-każijiet ta' reazzjonijiet avversi li seħħew f'Malta kienu aċċessibbli għal kull studju Ewropew li rrikjeda l-investigazzjoni tar-riskju tal-prodott farmaċewtiku kkonċernat.
Illum inhares lejn il-hajja jum wara jum.	Illum inħares lejn il-ħajja jum wara jum.	Illum inħares lejn il-ħajja jum wara jum.
l-akkużi jittrattaw perjodu ta' għaxar snin sas-sena li għaddiet.	L-akkużi jittrattaw perjodu ta' għaxar snin sas-sena li għaddiet.	L-akkużi jittrattaw perjodu ta' għaxar snin sas-sena li għaddiet.
Nuru mod u fir-rejalta nkunu mod ieħor	Nuru mod u fir-realtà nkunu mod ieħor.	Nuru mod u fir-realtà nkunu mod ieħor.
Skada il-hin ghal-mistoqsiejiet parlamentari	Skada l-ħin għall-mistoqsijiet parlamentari.	Skada l-ħin għall-mistoqsijiet parlamentari.

6.2.3 | Performance on Correct Inputs

The final evaluation examined how the sentence-level model behaved when provided with correctly written sentences.The model achieved a preservation accuracy of 46.2%, meaning that fewer than half of the correct sentences were left unchanged. The remaining 53.8% were modified unnecessarily, demonstrating a high level of overcorrection.

The results in Table 5.4 show that the model preserved 462 out of the 1,000 correct sentences, resulting in a preservation accuracy of 46.2%. The remaining 53.8% of sentences were altered unnecessarily, indicating a notable tendency to overcorrect. This means that more than half of the time, the model changed sentences that did not contain any errors.

This behaviour is consistent with what was observed in the word-level model, and users would likely encounter similar disruptions, inconsistencies, and unreliable outputs because correct inputs, which should remain unchanged, are instead modified.

Table 6.9: Examples of real-world sentences where the model output did not match the expected correction. Source: Author.

Input	Model Output	Target
aħallini nweiġbek	**aħallini wieġeb.**	Ħallini nwieġbek!
Min iooublikahom jidher li daħal fl-arkivji digitali tal-personalitajiet mingħajr permess.	Min **ioħroġlikhom** jidher li daħal fl-arkivji diġitali tal-personalitajiet mingħajr permess.	Min ippubblikahom jidher li daħal fl-arkivji diġitali tal-personalitajiet, mingħajr permess.
jigifiei bala percentagg ta xoghol x'sar?	Jiġifieri bħala **perċentagg** ta' xogħol x'sar?	Jiġifieri bħala persentaġġta' xogħol, x'sar?
Kif nistghu nkunu favur?	Kif nistgħu nkunu favur**?**	Kif nistgħu nkunu favur!?
Il-Gvern Malta ghamel hiltu kollha biex fn ma jaghmlux.	Il-Gvern **Malta** għamel ħiltu kollha biex **fn** ma jagħmlux.	Il-Gvern Malti għamel ħiltu kollha biex dan ma jagħmlux.

Note. The **bold text** in the *Model Output* column highlights the parts where the model's prediction did not match the target.

Like the word-level model, the sentence-level model was trained almost entirely on inputs that required correction, so it rarely saw examples where no changes were needed. With only a small number of correct sentences included during fine-tuning, the model developed a strong bias toward altering most inputs and therefore did not reliably learn when to preserve text that was already correct.

A closer look at the overcorrected outputs showed two recurring issues. Sometimes the model replaced correct words with unnecessary alternatives, and in other cases it rephrased parts of the sentence even though no correction was needed. These unnecessary changes make the system less reliable in practical use.

To illustrate this behaviour, Tables 6.10 and 6.11 provide examples from the evaluation. Table 6.10 shows instances where the model successfully preserved the input, demonstrating that it can correctly recognise when no correction is needed. Table 6.11 highlights cases where the model introduced unnecessary edits, showing the different types of overcorrection that occurred.

Overall, the findings from this test set show clearly that the sentence-level model tends to overcorrect when handling error-free text, which limits its reliability as a general-

Table 6.10: Examples of correctly written sentences that the sentence-level model preserved without applying any changes. Source: Author.

Input	Model Output	Target
Tinsiex li aħna naħdmu fuq bażi volontarja.	Tinsiex li aħna naħdmu fuq bażi volontarja.	Tinsiex li aħna naħdmu fuq bażi volontarja.
X'jista' jsir?	X'jista' jsir?	X'jista' jsir?
Il-mira taċ-ċentru hi li tagħmel ix-xjenza aċċessibbli għal kulħadd.	Il-mira taċ-ċentru hi li tagħmel ix-xjenza aċċessibbli għal kulħadd.	Il-mira taċ-ċentru hi li tagħmel ix-xjenza aċċessibbli għal kulħadd.
La 'l hemm u lanqas 'l hawn.	La 'l hemm u lanqas 'l hawn.	La 'l hemm u lanqas 'l hawn.
Huma żewġ inizjattivi paralleli, indipendenti minn xulxin.	Huma żewġ inizjattivi paralleli, indipendenti minn xulxin.	Huma żewġ inizjattivi paralleli, indipendenti minn xulxin.

purpose spell-checker. A more balanced training setup that includes a larger proportion of correct sentences would help the model learn when to abstain from making changes.

6.3 | Comparison with Prior Research and Existing Tools

This section critically examines how the models developed in this study compare with previous research on Maltese spell checking and with existing tools, namely the publicly available spelling.mt. The aim is to assess the relative strengths and limitations of the proposed models within the broader landscape of Maltese spell checking approaches.

6.3.1 | Comparison with Traditional and Statistical-Based Research

As outlined in Section 3.1, the earliest documented work on Maltese spell checking is that of Mangion (1999) and Mizzi (2000). However, these studies cannot be directly compared with the models developed in this dissertation because they employ fundamentally different evaluation methodologies. Mangion (1999) assessed MSpell using a classification-style approach, measuring how accurately the system distinguished correct words from misspellings and, for the errors it identified, the proportion for which a suitable correction was suggested. In contrast, Mizzi (2000) adopted an evaluation framework suitable for early statistical spell checking systems. Therefore, this approach

Table 6.11: Examples of correctly written sentences that the sentence-level model modified unnecessarily, illustrating different types of overcorrections. Source: Author.

Input	**Model Output**	**Target**
Kulħadd maħruq, kulħadd daħaq bina.	Kulħadd **maħruġ**, kulħadd **ġie ħa naħa**.	Kulħadd maħruq, kulħadd daħaq bina.
Ħallini nwieġbek!	Ħallini **nwieġebek**!	Ħallini nwieġbek!
Is-Sibt hu jum kruċjali.	Is-Sibt hu **dlam** kruċjali.	Is-Sibt hu jum kruċjali
Se nsaħħu s-sehem tagħna fl-Unjoni Ewropea.	Se **nisahhu** s-sehem tagħna fl-Unjoni Ewropea.	Se nsaħħu s-sehem tagħna fl-Unjoni Ewropea.
U x'aktarx jimblokkawlu kollox.	U x'aktarx **li** jimblokkawlu kollox.	U x'aktarx jimblokkawlu kollox.

Note. The **bold text** in the *Model Output* column highlights the parts where the model's prediction did not match the target.

differs from the metrics adopted in this research, which follow contemporary practices for neural spell checking and Grammatical Error Correction (GEC) as outlined in Section 5.1.2. Consequently, the results reported in that work cannot be directly or quantitatively compared with the evaluation measures employed in this study.

A more recent contribution is the spelling and grammar checking prototype developed by Buhagiar (2021), which combines edit-distance methods with a Maltese Soundex variant. Its evaluation focused on hit rate, measuring how often the correct word appeared as the top candidate in the system's suggestion list. This makes the assessment centred on candidate generation, rather than full correction as produced by the models in this dissertation. The system itself is not publicly accessible, and its evaluation design, test dataset, and scoring method differ entirely from those adopted here. For these reasons, a quantitative comparison with the models developed in this dissertation is not possible.

6.3.2 | Comparison with Publicly Available Tool

To our knowledge, the only publicly available Maltese spell checker is spelling.mt. Its reported hit rate of 48.8%, as measured by Buhagiar (2021), indicates how often the cor-

rect suggestion appears first in its ranked list of candidate words. Because spelling.mt outputs multiple candidate corrections rather than a single predicted form, this evaluation was based on ranking quality rather than exact-match correction. Consequently, the hit-rate metric used for spelling.mt is not directly comparable to the accuracy- and edit-based metrics adopted in this dissertation. Differences in evaluation datasets further limit the possibility of comparison, as the dataset used to assess the models developed in this study differs from that used by Buhagiar (2021) to evaluate spelling.mt.

Although it would have been possible to supply the dataset used in this dissertation to spelling.mt to determine how often the expected target word appears among its ranked candidates, this approach was not practical. Although the tool is open source and could, in principle, be set up locally, time constraints prevented doing so. Additionally, its manual web interface or text editor plugin does not support automated querying, and performing the evaluation manually by entering each input and extracting outputs individually would have required considerable time, making this approach infeasible.

Even if such an evaluation had been feasible, a meaningful comparison with the models developed in this study would still not have been possible. The sentence-level model cannot be compared at all, as spelling.mt does not handle real-word errors, which are a core feature of the sentence-level model. While the word-level model could, in theory, be compared to spelling.mt, fundamental differences in output formats present limitations. The word-level model produces a single corrected word, whereas spelling.mt provides a list of candidate corrections. As a result, direct comparison is problematic. For spelling.mt, the expected correction may appear second or third in the list and still be counted as successful, whereas the word-level model is evaluated on a single output, which is either correct or incorrect. A fair comparison would require the word-level model to generate a ranked list of candidate corrections and compare the quality of both lists. This could be achieved using a custom dataset in which each incorrect input is paired with multiple valid corrections. Since such a dataset was not available, this limitation prevents a fully systematic evaluation against spelling.mt.

6.3.3 | Comparison with Recent Deep Learning Research

Recent approaches provide the most meaningful point of comparison for the sentence-level model developed in this dissertation because they use similar architectures, rely on synthetic and authentic error data, and report performance using ERRANT. These studies are those of Debattista (2022), Oliveri (2025), and Busuttil (2025) (reported in Oliveri (2025)). All three evaluated their systems on real-world errors. Although the

datasets are not identical to the one used in this dissertation, they represent the same type of real-world scenario. For this reason, the comparison presented below is based primarily on the real-world error results.

The ERRANT F0.5-scores reported in the previous studies are as follows: Debattista (2022) achieved an F0.5 score of 31.84%, Oliveri (2025) reported a score of 33.40%, and Busuttil (2025) obtained a score of 34.66%.

These studies collectively represent all prior work on Maltese GEC undertaken prior this dissertation. Each of them employed encoder-decoder models trained on datasets that integrated both synthetic errors and naturally occurring mistakes.

The sentence-level model developed in this dissertation achieved an ERRANT F0.5-score of of 0.72 (72%) on the real-world error test set.The real-world test results offer a more appropriate basis for comparison because prior studies were also tested on naturally occurring errors. The real-world performance of the model presented in this dissertation therefore aligns with the evaluation scenario used by Debattista (2022), Oliveri (2025), and Busuttil (2025), allowing a cautious but meaningful comparison at the methodological level.

The results achieved by the sentence-level model developed during this research achieved a higher score than those reported in previous work studies. While differences in dataset composition prevent a strict one-to-one comparison, this result indicates that the model presented here is capable of applying correct and well-placed edits with substantially greater consistency than earlier Maltese systems. The improvement may be attributed to the larger underlying LLM, the instruction-tuned training method, and the model's broader representational capacity, as well as the influence of the tailored training data, which likely supported more effective learning.

Despite the strong real-world ERRANT performance, the model still showed difficulties similar to previous studies. Like earlier systems, it struggled with errors that did not resemble the synthetic patterns seen during training and demonstrated overcorrection tendencies. These limitations reflect broader constraints in Maltese GEC research, including limited real-world training data and the uncertainty surrounding the amount of Maltese linguistic knowledge contained in multilingual LLMs such as Large Language Model Meta AI (LLaMA).

Overall, the findings suggest that the sentence-level model performs competitively with other Maltese GEC systems and demonstrate that LLMs can serve as a viable approach to spell checking in low-resource languages such as Maltese.

6.4 | Transferability to Other Low-Resource Languages

The approach outlined in this dissertation can potentially be applied more broadly to other low-resource languages. Several components of the methodology, such as building a clean reference corpus, generating synthetic errors, and fine-tuning a multilingual LLM using Parameter-Efficient Fine-Tuning (PEFT) techniques, are not exclusive to Maltese. These components can be adapted to various linguistic contexts.

Clear and well-structured text is essential for helping the model recognise correct patterns. Additionally, synthetic errors can be used to create valuable training examples and increase the amount of data available to the model. Furthermore, fine-tuning a multilingual LLM can improve learning in low-data settings. With appropriate adjustments, this workflow could be used for other low-resource languages that lack extensive corpora.

However, each low-resource language has unique linguistic characteristics that affect how the method should be adapted. Factors such as differences in morphology, spelling systems, pronunciation, and writing conventions will influence the nature of the errors and how easily a model can learn to correct them. For example, languages with rich inflectional morphology or non-Latin scripts may require tailored error-generation strategies or specific pre-processing tools. Additionally, the availability and quality of text sources can vary. This variability impacts both the reference corpus and the synthetic errors generated from it.

Despite these variations, the promising results achieved with Maltese suggest that the general approach can be effective in low-resource settings. This indicates that the methodology adopted in this dissertation could provide a viable approach for developing spell checking tools for other low-resource languages.

6.5 | Summary

This chapter examined how the word-level and sentence-level models performed across the different evaluation settings and interpreted the factors that shaped these results. The analysis highlighted where the models were effective, where they struggled, and why these patterns emerged, reflecting the influence of synthetic training data, limited exposure to correct inputs, and the complexity of real-world errors.

Both models performed strongly on synthetic incorrect-correct pairs but showed reduced effectiveness on real-world mistakes and displayed overcorrection tendencies on error-free inputs. These findings underscore the importance of more balanced training

data and greater exposure to naturally occurring errors.

The chapter also positioned the results within the broader context of Maltese spell checking research. While early systems were not directly comparable, the sentence-level model achieved performance that aligns well with the more recent neural approaches for Maltese, indicating that LLMs can offer competitive behaviour even in low-resource settings.

Overall, the discussion linked the empirical findings to the research aim, identified key limitations, and showed how the models relate to existing work, providing a clear basis for the conclusions that follow. It also discussed that the methodology developed in this study could be adapted to other low-resource languages, provided that the specific characteristics of each language are taken into account.

7

Conclusion

This chapter summarises the main findings of the study and evaluates how effectively it met its aim of developing a Maltese spell checker through the fine-tuning of the Meta's Large Language Model Meta AI (LLaMA), Large Language Model (LLM). It revisits the research objectives, highlights the key results from both the word-level and sentence-level evaluations, identifies the principal limitations of the approach, and outlines several directions for future research. Taken together, these elements provide a final evaluation of the study's contribution and its relevance to the wider field of Maltese spell checking.

7.1 | Revisiting the Research Aim and Objectives

The aim of this dissertation was to investigate whether a reliable Maltese spell checker could be developed by fine-tuning an open-source LLM, specifically Meta's LLaMA model, using a custom Maltese dataset. As outlined in Chapter 1, several objectives were established to address this aim:

1. Data Collection and Pre-processing - Gather and prepare a comprehensive dataset of Maltese text, including common spelling mistakes and their corrections.
2. Fine-tuning of the LLM - Perform fine-tuning on the selected LLM using the custom Maltese dataset.
3. Evaluation - Evaluate the fine-tuned model using proper evaluation metrics.
4. Comparison with existing spell checkers - Evaluating the fine-tuned model against existing spell checkers to assess whether it outperforms or falls short of their performance.

The first objective, ***Data Collection and Pre-processing - Gather and prepare a comprehensive dataset of Maltese text, including common spelling mistakes and their cor-***

rections, was achieved by collecting correctly written Maltese text and generating synthetic errors informed by findings in the literature on common spelling patterns in Maltese writing. Furthermore, a real-world error dataset from previous studies was incorporated, to ensure that the evaluation considered naturally occurring errors in addition to synthetic ones. These processes are described in detail in Section 4.3.

The second objective, ***Fine-tuning of the LLM - Perform fine-tuning on the selected LLM using the custom Maltese dataset***, was fulfilled by applying the custom dataset to fine-tune the LLaMA model, as detailed in Section 4.4.

The third objective, ***Evaluation - Evaluate the fine-tuned model using proper evaluation metrics***, was met by applying contemporary metrics commonly used in modern spell-checking and Grammatical Error Correction (GEC) research. These metrics, selected based on the relevant literature, are described in Sections 4.5 and 5.1.2.

Lastly, the fourth objective, ***Comparison with existing spell checkers - Evaluating the fine-tuned model against existing spell checkers to assess whether it outperforms or falls short of their performance***, was addressed by comparing the models developed in this study with prior research on Maltese spell checking and with the publicly available tool spelling.mt, as outlined in Section 6.3. However, as discussed in the same section, not all previous work could be directly compared to this study due to differences in system design, intended purpose, and evaluation methodology, which resulted in incompatible metrics. Moreover, although recent studies employed similar evaluation metrics, the underlying test datasets were not identical. Consequently, the comparisons presented are indicative only, because the same metrics were compared, but these metrics were obtained on different test datasets than those used in the systems developed by Debattista (2022), Oliveri (2025), and Busuttil (2025). A rigorous, direct comparison would require identical test datasets, standardised metrics, and consistent task definitions for all systems. Since the datasets used in previous studies differ from those employed in this dissertation, the current results cannot establish which system performs better under equivalent conditions. Using the same datasets for all systems would be necessary to fairly determine their relative performance in comparable scenarios.

7.2 | Summary of Key Findings

The primary objective of this study was to develop and evaluate a Maltese spell checker by fine-tuning an LLM. This was pursued through two complementary models, a word-level model and a sentence-level model. Both models were evaluated on multiple datasets and using a range of metrics to assess their effectiveness across different

types of errors and input conditions.

7.2.1 | Word-Level Model

The word-level model performed strongly on the synthetic incorrect-correct pairs, achieving 84.64% accuracy, precision of 1.0, recall of 0.846, and a low Character Error Rate (CER), indicating that most predictions closely matched the intended targets. Around 5,600 model outputs that did not match the expected target were still valid Maltese word forms, showing that some apparent "errors" were due to morphological variation rather than incorrect correction.

On correctly written words, the model preserved 33.2% of inputs, indicating a degree of overcorrection linked to the overwhelmingly error-focused training data.

7.2.2 | Sentence-Level Model

The sentence-level model achieved very strong performance on synthetic errors, with 95.3% accuracy, precision of 0.99, recall of 0.95, and near-perfect scores for Bilingual Evaluation Understudy (BLEU) (99.21) and Grammar Language Evaluation Understudy (GLEU) (0.99). Furthermore the model achieved a CER of 0.46%, Word Error Rate (WER) of 0.73%, and Error Annotation Toolkit (ERRANT) F0.5-score of 0.98.

Performance declined when the model was evaluated on real-world errors. Accuracy fell to 29.6%, recall to 0.28, the Error Annotation Toolkit (ERRANT) $F_{0.5}$ score to 0.72, BLEU to 79.27, and GLEU to 0.78. These results indicate that the model found naturally occurring and more irregular errors considerably more challenging, particularly those that did not resemble the patterns underlying the synthetically generated errors. The full set of results is presented in Table 5.4.

When evaluated on correct inputs, the model preserved only 46.2% of error-free sentences and modified the remaining 53.8%, demonstrating a pronounced tendency to overcorrect, often replacing valid words or rephrasing text unnecessarily.

7.2.3 | Summary

Across both model types, performance was highest on synthetic errors, which followed the predictable patterns reflected in the training data. Real-world errors and correct-input tests revealed two consistent behaviours:

- Limited generalisation: Both models struggled with naturally occurring errors, which were more diverse and structurally irregular than synthetic ones.

- Overcorrection: Both models frequently altered inputs that were already correct, reflecting the limited exposure to error-free examples during fine-tuning.

These findings underline the importance of incorporating more realistic data and a balanced mixture of correct and incorrect examples in future training strategies.

7.3 | Limitations

The limitations discussed earlier in Section 4.6 shaped both the design and the outcomes of this study. This section summarises the constraints most relevant to interpreting the results, with emphasis on how they affected model performance.

7.3.1 | Data and Synthetic Error Generation

As outlined in Section 4.6, the absence of a real-world Maltese incorrect-correct dataset meant that the majority of the training data consisted of synthetically generated errors. While this approach is standard in low-resource settings, it inevitably restricts the range and diversity of error types the models are exposed to. This limitation was reflected in the evaluation, where both the word-level and sentence-level models performed well on synthetic error patterns but struggled with naturally occurring real-world mistakes, which were more varied and less predictable.

This finding highlights the need for substantially more real-world Maltese data, especially naturally occurring incorrect–correct pairs, to enable the models to learn a wider spectrum of error types.

Furthermore, because the training data contained very few correct examples as inputs (correct-correct pairs), the models did not develop a strong "no-change" behaviour. This contributed directly to the overcorrection observed in both models, especially at sentence level, where more than half of the correct inputs were unnecessarily modified.

Including a larger number of correct-correct pairs would have helped the models recognise when no edit is needed. In addition, using a broader and more diverse set of Maltese text would have given the models better exposure to the language and improved their ability to tell the difference between correct forms and genuine errors.

These limitations have important implications for real-world deployment. In addition to the high rate of overcorrection observed in both models, where more than half of the correct inputs were unnecessarily modified, the performance drop on the real-world test set for the sentence-level model indicates that it struggles to produce the exact target output when confronted with naturally occurring and less predictable error

patterns. Together, these findings suggest that users could encounter both unnecessary modifications and incorrect or missed corrections, which may disrupt workflow and reduce confidence in the system's reliability. This is particularly concerning because such a tool is expected to provide consistent and accurate feedback. Before deployment in public-facing applications, the models would therefore require further refinement, including increased exposure to correct-correct examples to reduce the bias toward correcting every input, a larger volume of authentic Maltese incorrect-correct pairs, and adjustments to fine-tuning parameters to make the model more conservative in applying corrections. Additionally, models which are pre-trained on a larger corpus of Maltese text could further improve its understanding of correct language use and help mitigate these limitations. Addressing these issues is essential to ensure that the system is not only capable of correcting errors but also dependable in preserving correct text and handling real-world variation.

7.3.2 | Constraints on Experimentation

As noted in Section 4.6, time constraints restricted the scope of experimentation. Only a narrow range of fine-tuning configurations was tested, and variations such as alternative Low-Rank Adaptation (LoRA) hyperparameters, learning-rate schedulers, or multiple training seeds were not explored. These constraints limit the breadth of the conclusions that can be drawn about optimal training setups for Maltese spell checking.

7.3.3 | Evaluation Constraints

Meaningful comparison with existing tools and previous studies is constrained by variations in evaluation methodologies. Systems such as spelling.mt and earlier approaches were tested using hit-rate measures or candidate-ranking procedures, whereas this evalution framework adopted for this study applies metrics commonly used in neural GEC. Moreover, the lack of a common test set or standardised evaluation framework limits the possibility of drawing direct, quantitative comparisons with more recent research on Maltese GEC.

7.3.4 | Deployment in Real-World Applications

The models developed in this study demonstrate promising results under experimental conditions, however, their real-world deployment presents practical limitations. The primary constraint stems from the computational requirements of the underlying LLM.

Such requirements make local integration into word processors impractical for typical consumer hardware.

Deployment via a remote Application Programming Interface (API) represents a technically feasible alternative. However, hosting a model of this scale necessitates substantial Graphics Processing Unit (GPU) infrastructure, resulting in high operational costs. Additionally, latency, scalability, and data privacy considerations must be addressed for real-world use. These constraints limit the immediate practicality of the system.

7.4 | Future Work

Several directions for further research emerge from this study.

- Evaluate a wider range of fine-tuning configurations: This work explored only a narrow set of hyperparameters. Testing alternative hyperparameter would provide a clearer understanding of how training choices affect performance and may lead to better-generalising models.
- Increase the amount and variety of real-world error data: Larger collections of authentic incorrect-correct pairs would help the models learn from naturally occurring error patterns rather than relying heavily on synthetic generation.
- Incorporate more correct Maltese sentences during training: A more balanced dataset containing additional correct-correct examples would help mitigate over-correction.
- Explore multi-candidate output generation: Allowing models to produce several candidate corrections could capture valid alternatives that single-output systems may miss, improving flexibility and evaluation fairness.
- Develop a hybrid spell-checking system: Integrating an LLM with traditional methods, such as dictionary lookup, edit-distance algorithms, may help refine outputs and reduce invalid corrections.
- Adopt or develop Maltese-specific language models: Continued pre-training or targeted adaptation of LLMs on Maltese corpora could improve linguistic coverage and address current limitations arising from limited pre-training exposure.

7.5 | Final Remarks

The models developed in this dissertation are not definitive solutions for Maltese spell checking. Instead, they demonstrate that fine-tuning an LLM offers a promising and viable approach for supporting low-resource languages. The results achieved, indicate that this technique has the potential to form the basis of an effective spell checking tool.

However, the study also highlights important limitations that must be addressed before such systems can be deployed in real-world contexts. The proposed directions for future work outline several practical steps for overcoming these constraints, including expanding real-world datasets, refining training configurations, and exploring Maltese-specific model adaptations. Pursuing these developments would provide deeper insights into the behaviour and capabilities of LLM-based spell checkers and could lead to more robust and reliable tools.

Overall, while this work represents an initial step, it contributes evidence that LLM fine-tuning can play a meaningful role in advancing language technology for low-resource settings. Continued research building on these foundations has the potential to produce solutions that more effectively meet the needs of Maltese language users.

A

Types of Spelling Errors in Maltese

Table A.1: Non-word and real-word errors using Maltese examples. Source: Author.

Error Type	Spelling Error	Correction	Explanation
Non-word	Qiegħed **niktevb** sentenza.	Qiegħed **nikteb** sentenza.	The word **niktevb** represents a non-word spelling error caused by an **insertion error**. This likely resulted from the accidental insertion of the adjacent letter **v**, possibly due to an unintentional keystroke. The resulting form, **niktevb**, does not exist in the Maltese language and is therefore classified as a non-word error.

(Continued on next page)

(Continued from previous page)

Error Type	Spelling Error	Correction	Explanation
Non-word	**Iktb** din is-sentenza.	**Ikteb** din is-sentenza.	The word **iktb** represents a non-word spelling error caused by an **omission error**. This likely occurred due to the accidental omission of the vowel **e** in the final syllable, resulting in the incorrect form **iktb** instead of **ikteb**. The word **iktb** does not exist in the Maltese language and therefore qualifies as a non-word error.
Non-word	Il-**jarozza** hija ta' lewn aħmar.	Il-**karozza** hija ta' lewn aħmar.	The word **jarozza** represents a non-word spelling error caused by a **substitution error**. This error likely arose from the accidental substitution of the letter **k** with the adjacent key **j** on the keyboard, producing the incorrect form **jarozza** in place of **karozza**. Since **jarozza** is not a valid word in the Maltese language, it is classified as a non-word error.

(Continued on next page)

(Continued from previous page)

Error Type	Spelling Error	Correction	Explanation
Non-word	Dun Karm Psaila **kitbe** l-Innu Malti.	Dun Karm Psaila **kiteb** l-Innu Malti.	The word **kitbe** constitutes a non-word spelling error resulting from a **transposition error**. This occurred due to the unintended reversal of the letters **b** and **e**, yielding the incorrect form **kitbe** instead of the correct **kiteb**. As **kitbe** is not a valid word in the Maltese language, it is classified as a non-word error.
Non-word	Il-belt **kappitali** ta' Malta hija il-Belt Valletta.	Il-belt **kapitali** ta' Malta hija il-Belt Valletta.	The word **kappitali** is a non-word spelling error caused by a **duplication error**. This occurred due to the unintended repetition of the letter **p**, resulting in the incorrect form **kappitali** instead of the correct **kapitali**. Since **kappitali** is not a valid word in the Maltese language, it is classified as a non-word error.

(Continued on next page)

(Continued from previous page)

Error Type	Spelling Error	Correction	Explanation
Non-word	Il-futbol jintlagħab b'**pallun**.	Il-futbol jintlagħab b'**ballun**.	The word **pallun** is a non-word spelling error resulting from a **phonological error**. This occurred due to the substitution of the letter **b** with **p**, likely caused by phonological confusion between the two sounds. The incorrect form **pallun** was produced instead of the correct **ballun**. As **pallun** is not a valid word in the Maltese language, it is classified as a non-word error.
Real-word	Sena fiha tnax-il **xagħar**.	Sena fiha tnax-il **xahar**.	The word **xagħar** is a real-word spelling error resulting from a **homophone error**. This occurred due to the incorrect use of **xagħar** (hair) in place of **xahar** (month), likely caused by confusion between the two similarly sounding words. Although both forms are valid in Maltese, only **xahar** is appropriate in this context. Therefore, the substitution leads to a real-word error with semantic implications.

(Continued on next page)

(Continued from previous page)

Error Type	Spelling Error	Correction	Explanation
Real-word	Il-kors **hija twila** sentejn.	Il-kors **huwa twil** sentejn.	By using **hija twila**, a real-word error is introduced, as the subject **kors** (course) is masculine, while **hija twila** is a feminine construction. The correct form should be **huwa twil** to match the grammatical gender of the subject.

B

Word-Level Level Error Types in Fine-Tuning Dataset

This table serves as a concise reference for the various error types addressed in this dissertation at the word-level. This table is based on the studies of Bezzina (2020), Blundell (2014), Dimech (2013), and Schembri (2016), which identify common spelling errors in Maltese writing.

Table B.1: Word-level error types in fine-tuning dataset. Source: Author.

Error Type	**Description**
Typographical errors	This includes: missing or extra letters, swapped letters, insertion of adjacent letters from the keyboard that shouldn't appear in the word, replacement with visually similar letters, and letter duplication.
Apostrophe and accent confusion	This involves mistakenly replacing an apostrophe with an accent mark or vice versa in a word.
Apostrophe and accent omission	This involves mistakenly omitting an apostrophe or an accent in a word.

(Continued on next page)

(Continued from previous page)

Error Type	**Description**
Phonological errors	Replacing letters with phonological similar letters, example using “b” instead of “p”, “t” instead of “d”, “ġ” instead of “ċ” or using “tx” instead of “ċċ”.
Single consonants instead of double	When a word which makes use of a double consonant is only written with, for example “kittieb” (correct) is written as “kitieb” (mistake).
Diacritics omission	Letters ċ, ġ, ħ and ż are replaced with the letters c, g, h and z respectively.
Letters “h” and “għ” are wrongly positioned or omitted	These letters are often written out of position example “tahgħa” (mistake) instead of “tagħha” (correct) or even left out from the word completely, example “kellima” (mistake) instead of “kellimha” (correct).
Replacing “h” with “ħ”	This error occurs when the letter “h” is replaced with ħ”, often due to their visual similarity. Additionally, when “h” appears near the end of a word, it can sometimes have a pronunciation similar to “ħ”.

C

Sentence-Level Error Types in Fine-Tuning Dataset

This table serves as a concise reference for the various error types addressed in this dissertation at the sentence-level. This table is based on the studies of Bezzina (2020), Blundell (2014), Dimech (2013), and Schembri (2016), which identify common spelling errors in Maltese writing.

Table C.1: Sentence-level error types in fine-tuning dataset. Source: Author.

Error Type	**Description**
Article omission	Instances where articles are improperly omitted.
Incorrect punctuation of articles	Errors involving the misuse of apostrophes in articles, which may need to be replaced with hyphens.
Incorrect use of “ie” in negative statements	Mistakes where the vowel “ie” is improperly used in negative statements.

(Continued on next page)

(Continued from previous page)

Error Type	Description
Article assimilation removal	Errors involving the elimination of correct article assimilation, where an article blends phonetically with the word that follows it. For example writing "iċ-ċavetta" (correct) as "il-ċavetta" (mistake); "id-dar" (correct) as "il-dar" (mistake); and "ix-xemx" (correct) as "il-xemx" (mistake).
Preposition word separation	Errors in separating the preposition word. For example writing "tal-" (correct) as "ta'l-" (mistake); and "mal-" (correct) as "ma'l-" (mistake).
Articles begin with a vowel after the previous word ends with a vowel	In Maltese, when a word ends with a vowel, the article of the following word should not begin with another vowel. This synthetic error reproduces such a spelling mistake by allowing both vowels to appear consecutively. For example, writing "jiktbu l-komponimenti" (correct) as "jiktbu il-komponimenti" (mistake) or "marru x-xogħol" (correct) as "marru ix-xogħol" is incorrect (mistake).

(Continued on next page)

(Continued from previous page)

Error Type	**Description**
Word starts with a vowel after a word ending with a vowel	In Maltese, when a word ends with a vowel, the following word should not begin with another vowel, except in specific cases where the initial "i" must always be retained (some examples are shown in Appendix D). This synthetic error reproduces such a spelling mistake by starting a word with a vowel immediately after the preceding word ends with one. For example, writing "għada nsiefer" (correct) as "għada insiefer" (mistake) or "il-kantanti jkantaw" (correct) as "il-kantanti ikantaw" (mistake).
Separation of words that should be written as one	For example writing "mhux" (correct) as "m'hux" (mistake) and "mhuwa" (correct) as "m'huwa" (mistake).
Incorrect use of prepositions	Mistakes involving the incorrect use of prepositions. For example writing "mill-" instead of "mil-" and "lil-" instead of "lill-".
Sentence capitalisation	Initial letter in a beginning of a sentence is not capitalised.

(Continued on next page)

(Continued from previous page)

Error Type	Description
Swap words that should start with the letter “i” to start with “j” and vice versa	In Maltese, a common spelling error involves using the letter “j” at the beginning of a word instead of “i”, or vice versa. The correct choice typically depends on the surrounding words in the sentence. This synthetic error replicates such orthographic confusion by interchanging “i” and “j” at the start of words to create realistic spelling mistakes. For instance, it is incorrect to write “biex idoqq” as “biex jdoqq”, and to write “jistgħu jweġġgħu” as “jistgħu iweġġgħu”.
Swap words that should start with the letter “u” to start with “w” and vice versa	In Maltese, a common spelling error involves using the letter “w” at the beginning of a word instead of “u”, or vice versa. The correct choice typically depends on the surrounding words in the sentence. This synthetic error replicates such orthographic confusion by interchanging “u” and “w” at the start of words to create realistic spelling mistakes. For instance, it is incorrect to write “li wħud” as “li uħud”, and to write “instigaw ukoll” as “instigaw wkoll”.

(Continued on next page)

(Continued from previous page)

Error Type	**Description**
Incorrect use of numeral form	This synthetic error represents cases where the wrong form of a numeral is used. In Maltese, certain numerals have multiple valid forms depending on grammatical context. This error simulates confusion between these forms by substituting one for the other, such as writing "erbgħa" instead of "erba"', or vice versa.
Verb and grammar errors	This synthetic error was designed to replicate grammatical mistakes such as the incorrect use of verb forms, including errors in tense or gender agreement. These variations were generated using data obtained from the **verb.mt** resource, which provided access to a wide range of Maltese verb conjugations. Examples of this type of synthetic error include writing "huwa torqod" instead of "huwa jorqod" (gender error); "hija jorqod" instead of "hija torqod" (gender error), "huma tkellem" instead of "huma tkellmu" (used singular instead of plural), and "ilbieraħ huwa jiekol" instead of "ilbieraħ huwa kiel" (tense error).

D

Words Where the Initial Vowel Must Be Retained

This appendix presents a list of words in which the initial vowel “i” must always be retained as part of the correct word form.

- idea
- ideat
- ideoloġija
- ideoloġija
- ideoloġikament
- interessa
- interessanti
- interess
- interessi
- interess
- interessi
- ingredjent
- ingredjenti
- insett
- insetti
- irresponsabbli
- irreparabbli
- immens
- immensament
- internazzjonali
- internazzjonalizzazzjoni
- insista
- Ingliż
- Ingliżi
- Ingliżata
- Ingliżati
- imperu
- imperi
- imperjali
- infern
- infernali
- illużjoni
- illużjonijiet
- illustri
- illustrazzjoni
- illegali
- illegalità
- illegalitajiet
- imbarazz
- imbark
- imbeċilli
- imbotta
- imprezzabbli
- infila
- imma
- immakulat
- immakulata
- immigrazzjoni
- immigrant
- immigranta
- immigranti
- imminenti
- impatt
- imparzjali
- imparzjalità
- impekkabbli
- impenn
- impenji
- impenjat
- impenjata

- impenjati
- impertinenti
- impetu
- impeti
- implika
- impliċitu
- importanti
- importanza
- importa
- importazzjoni
- impjega
- impjieg
- implimenta
- implimentazzjoni
- impona
- imponenti
- impunità
- imprenditur
- imprenditorjat
- impriża
- improbabbli
- improvizza
- improvizzazzjoni
- impruvja
- impronta
- inċensa
- inċens
- inċentivizza
- inċentiv
- inċentivi
- inċest
- inċident
- inċidenti
- inċidentalment
- Indja
- Indjan
- Indjana
- Indjani
- individwu
- individwi
- individwali
- individwalità
- indulġenza
- indulġenzi
- industrijalizza
- industrija
- industriji
- Infanzija
- indirizza
- infatti
- infezzjoni
- infiltra
- infiltrazzjoni
- infinit
- infinita
- infiniti
- infinità
- inflazzjoni
- influwenza
- influwenzi
- influwenza
- influwenzat
- influwenzata
- influwenzati
- informa
- informazzjoni
- informali
- informalità
- inġenerali
- inġinier
- inġiniera
- inġinerija
- inġust
- inġusta
- inġusti
- inġustizzja
- inġustizzji
- ingann
- inganni
- ingranaġġ
- inizjattiva
- inizjattivi
- inklużjoni
- inkluda
- inklussiv
- inkwiet
- inkwetat
- inkwetata
- inkwetati
- inkwistjoni
- insedjament
- insista

- insistenza
- intant
- intatt
- integra
- integrazzjoni
- interferixxa
- interferenza
- interferenzi
- interpreta
- interpretazzjoni
- interpretazzjonijiet
- interpretar
- interroga
- interrogazzjoni
- interrogazzjonijiet
- intervent
- interventi
- intervista
- intervisti
- intimu
- intima
- intimi
- intimità
- intrata
- intrati
- intrigu
- intriganti
- introduzzjoni
- introduzzjonijiet
- introvert
- introverżjoni
- invada
- invażjoni
- invista
- investiment
- invista
- investi
- invit
- inviti
- irregolari
- irregolarità
- irregolaritajiet
- irrispettivament
- ispirazzjoni
- ispirazzjonijiet
- ispirat
- ispirata
- ispirati
- istint
- istinti
- iżda

E

Synthetic Errors for Words

The following list includes all the synthetic errors applied at the word-level.

- Replace apostrophe with accent
- Remove a letter
- Swap letter positions
- Add extra letter
- Neighbouring letter mistake
- Replace letter with visually similar one
- Duplication of letter
- In the beginning of a word change “f” to “v”
- In the end of a word change “s” to “ż”
- Replace double consonant with single
- In the end of a word change “d” to “t”
- Anywhere in a word change “d” to “b”
- In the beginning of a word change “b” to “d”
- Anywhere in a word change “b” to “d”
- In the beginning of a word change “d” to “t”
- In the end of a word change “t” to “d”
- Anywhere in a word change “t” to “th”
- Append “w” for words ending with “u”
- Append “j” for words ending with “i”
- Anywhere in a word change “w” to “u”
- In the end of a word change “ż” to “s”
- Remove diacritics
- Anywhere in a word change “ż” to “s”
- Replace “ie” with “i”
- In the end of a word change “x” to “sh”
- Anywhere in a word change “x” to “sh”
- Replace accent with apostrophe
- Remove accent
- Anywhere in a word change “z” to “s”
- In the end of a word change “z” to “ts”
- In the end of a word change “ċ” to “k”
- In the end of a word change “ċċ” to “ġġ”
- Anywhere in a word change “ċċ” to “tx”
- In the end of a word change “ż” to “s”
- Anywhere in a word change “k” to “ck”
- Anywhere in a word change “k” to “c”
- In the beginning of a word change “t” to “d”
- In the end of a word change “g” to “k”
- Anywhere in a word change “g” to “c”
- Anywhere in a word change “tt” to “d”
- In the beginning of a word change “x” to “sh”

- Move h before its original position
- Move h after its original position
- Remove h from word
- Remove għfrom word
- Move “għ” before its original position
- Move “għ” after its original position
- In the end of a word change “ġ” to “ċ”
- Anywhere in a word change “ġ” to “j”
- Anywhere in a word change “tx” to “tc”
- Anywhere in a word change “tx” to “tċ”
- Anywhere in a word change “tx” to “ċ”
- In the end of a word change “p” to “b”
- Anywhere in a word change “mp” to “np”
- In the end of a word change “ġġ” to “ċċ”
- Anywhere in a word change “mb” to “nb”
- Remove apostrophe
- Anywhere in a word change “qf” to “fq”
- Anywhere in a word change “dt” to “tt”
- Anywhere in a word change “dt” to “d”
- Anywhere in a word change “dt” to “t”
- Anywhere in a word change “ln” to “nn”
- Anywhere in a word change “bq” to “qb”
- Replace “għ” with other letters
- Anywhere in a word change “bk” to “kp”
- Anywhere in a word change “bk” to “kb”
- Replace “h” with “ħ”
- Anywhere in a word change “mċ” to “nċ”
- Anywhere in a word change “ds” to “ts”
- Anywhere in a word change “ndn” to “nn”
- Anywhere in a word change “sx” to “xx”
- Anywhere in a word change “nb” to “mb”
- Anywhere in a word change “dx” to “c”
- Anywhere in a word change “dx” to “dċ”
- Anywhere in a word change “dx” to “ċ”
- Anywhere in a word change “nf” to “mf”
- Anywhere in a word change “lln” to “nn”
- Anywhere in a word change “ttx” to “ċċ”
- Anywhere in a word change “tts” to “zz”
- Anywhere in a word change “np” to “mp”
- Anywhere in a word change “ssx” to “ss”
- Anywhere in a word change “żżx” to “sx”

F

Synthetic Errors for Sentences

The following list includes all the synthetic errors applied at the sentence-level. Additionally, the synthetic errors referenced in Appendix E were also used in the creation of sentence-level errors, enabling examples where sentences contain word-level spelling mistakes.

- Convert the first word to lowercase
- Swap first letter “i” with “j” or vice-versa - For example writing “ikunu” instead of “jkunu” or vice-versa, because the correct form depends on the preceding word.
- Swap first letter “u” with “w” or vice-versa - For example writing “urew” instead of “wrew” or vice-versa, because the correct form depends on the preceding word.
- Vowel continuity - In Maltese, when a word ends with a vowel, the following word should not begin with another vowel, except in specific cases where the initial “i” must always be retained (some examples are shown in Appendix D). This synthetic error reproduces such a spelling mistake by starting a word with a vowel immediately after the preceding word ends with one.
- Article vowel continuity - Maltese, when a word ends with a vowel, the article of the following word should not begin with another vowel. This synthetic error reproduces such a spelling mistake by allowing both vowels to appear consecutively.
- Remove article assimilation - Errors involving the elimination of correct article assimilation, where an article blends phonetically with the word that follows it. For example writing “iċ-ċavetta” (correct) as “il-ċavetta” (mistake); “id-dar” (correct) as “il-dar” (mistake); and “ix-xemx” (correct) as “il-xemx” (mistake).
- Add “ie” in negative statements - Replacing “i” with “ie” in negative statements.

- Replace hyphen with nothing - This would result in the removal of any hyphens from articles, leading to words being concatenated without separation.

- Replace hyphen with a space - This would result in the removal of any hyphens from articles, leading to words being separated only with a space.

- Remove articles from words.

- Replace article apostrophe with a hyphen - This involves replacing an apostrophe in articles with a hyphen.

- Generate verb grammar errors -This operation creates intentional verb grammar mistakes, such as subject-verb agreement errors or tense issues.

- Incorrect use of prepositions - Mistakes involving the incorrect use of prepositions. For example writing “mill-” instead of “mil-” and “lil-” instead of “lill-”.

- Preposition word separation - Errors in separating the preposition word. For example writing “tal-” (correct) as “ta’l-” (mistake); and “mal-” (correct) as “ma’l-” (mistake).

- Separate combined words -This refers to splitting compound or combined words into individual components.

- Wrong use of numerical wording - This involves replacing numerical words to wrong forms example using “erbgħa” instead of “erba” or vice-versa as this depends on context.

- Wrong hyphen position for numerical words - In Maltese, numerical words have the article hyphen written before the article, for example “dsatax -il sena”. This synthetic error introduces spelling errors by inserting “dsatax il-sena” (mistake) instead of “dsatax -il sena” (correct).

G

Consent Forms for Data Collection and Processing

This appendix contains the signed consent form authorising the use of data for this dissertation. The document confirms that permission was obtained from the relevant parties to utilise the data for research purposes in accordance with ethical standards.

G.0.1 | L-Ilsien Malti Għal Qalbi

This subsection contains the signed consent form authorising the use of data from the publication *"L-Ilsien Malti Għal Qalbi"* for this dissertation. The consent form is presented on the following page.

Continued on next page

L-Università ta' Malta

Request for permission to access and use data

18th November 2024

Dear Prof Michael Spagnol,

My name is Owen Fava and I am a student at the University of Malta, presently reading for a Master of Science in Artificial Intelligence. I am conducting a research study for my dissertation titled Advancing Automated Maltese Spell Checking Using Deep Learning. to develop a Maltese spell checker by fine-tuning a Large Language Model (LLM). This project is being conducted under the supervision of Prof. Alexiei Dingli.

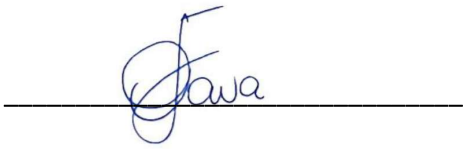
I am hereby seeking your permission to use the *"L-Ilsien Malti Għal Qalbi"* publication for data collection purposes. My data collection methods will involve automatic/manual extraction of Maltese words and sentences from the mentioned data source to build a corpus of correctly written text.

Participation will be entirely voluntary and participants will be free to withdraw at any point, without any repercussions. Data collected will be used to generate synthetic errors, producing an incorrect-correct dataset for fine-tuning the LLM.

Should you require further information, please do not hesitate to contact me; my contact details are provided below.

Thank you for your kind consideration of this request.

Sincerely,



Owen Fava
owen.fava.23.um.edu.mt

L-Università ta' Malta

Authorisation to access and use data

I hereby grant permission for the collection and use of the data as described above.

Name of authoriser: Michael Spagnol

Signature: 

Date: 18th November 2024

G.0.2 | Curia-Related Court Case Documents

This subsection includes the signed consent form granting permission to use the publicly available Curia-related court case documents for the purposes of this dissertation. The consent form is provided on the following page.

Continued on next page

L-Università ta' Malta

Request for permission to access and use data

3rd July 2025

Dear Robert Camenzuli,

My name is Owen Fava and I am a student at the University of Malta, presently reading for a Master of Science in Artificial Intelligence. I am conducting a research study for my dissertation titled Advancing Automated Maltese Spell Checking Using Deep Learning. to develop a Maltese spell checker by fine-tuning a Large Language Model (LLM). This project is being conducted under the supervision of Prof. Alexiei Dingli.

I am hereby seeking your permission to use the documents which are related to *"Kawżi mill-Qorti tal-Ġustizzja"* for data collection purposes. My data collection methods will involve automatic/manual extraction of Maltese words and sentences from the mentioned data source to build a corpus of correctly written text.

Participation will be entirely voluntary and participants will be free to withdraw at any point, without any repercussions. Data collected will be used to generate synthetic errors, producing an incorrect-correct dataset for fine-tuning the LLM.

Should you require further information, please do not hesitate to contact me; my contact details are provided below.

Thank you for your kind consideration of this request.

Sincerely,

Owen Fava
owen.fava.23.um.edu.mt

L-Università ta' Malta

Authorisation to access and use data

I hereby grant permission for the collection and use of the data as described above.

Name of authoriser: JOSEPH IZZO CLARKE

Signature: 

Date: 3rd July 2025

G.0.3 | verb.mt

This subsection contains the signed consent form granting permission to extract data from the website *verb.mt* (`https://verb.mt/`) and to make use of the collected data for the purposes of this dissertation. The consent form is presented on the following page.

Continued on next page

L-Università ta' Malta

Owen Fava <owen.fava.23@um.edu.mt>

verb.mt

Reuben Degiorgio <kellimna@ghidhabilmalti.mt> 7 March 2025 at 19:57
To: Owen Fava <owen.fava.23@um.edu.mt>
Cc: Prof Alexiei Dingli <alexiei.dingli@um.edu.mt>

Għażiż Owen,

Tista' tuża d-data li hemm fuq verb.mt, għalija mhi problema xejn. Jekk tista' tagħmel webscraping aħjar, għax m'għandix dokument wieħed lest li jiġbor l-informazzjoni kollha f'daqqa. Biex tkun taf, il-konjugazzjonijiet għamilthom kollha jiena u minkejja li qgħadt attent ħafna, jifdal dak il-ftit ċans li xi ħaġa tkun ħarbitli, bħal xi żball tat-tipa. Ma ntużawx algoritmi jew AI fit-tiswir ta' verb.mt.

Jekk tista' ssemmi lil verb.mt fit-teżi tiegħek inkun nafhulek. Nixtieqlek kull suċċess bl-istudji tiegħek u grazzi li qed taħdem fuq xi ħaġa tant importanti; ċekkjatur pubbliku huwa għodda li għandna bżonn wisq bħalissa!

Saħħiet,
Reuben

---- On Fri, 07 Mar 2025 16:37:17 +0100 **Owen Fava <owen.fava.23@um.edu.mt>** wrote ---
[Quoted text hidden]

L-Università ta' Malta

Owen Fava <owen.fava.23@um.edu.mt>

MSc Maltese Spell Checker - Data Consent Form

Reuben Degiorgio <kellimna@ghidhabilmalti.mt> 1 November 2025 at 08:59
To: Owen Fava <owen.fava.23@um.edu.mt>
Cc: Prof Alexiei Dingli <alexiei.dingli@um.edu.mt>

Għażiż Owen,

Jidher li ma nistax niffirma d-dokument mingħajr ma nħallas, użajt "signed" minflok - nispera li aċċettabbli hekk ukoll.

Din l-email isservi wkoll ta' prova li qed nagħti kunsens.

Tislijiet,
Reuben Degiorgio

---- On Wed, 29 Oct 2025 08:00:00 +0100 **Owen Fava <owen.fava.23@um.edu.mt>** wrote ---
[Quoted text hidden]

verb_mt-consent-form.pdf
138K

L-Università ta' Malta

Request for permission to access and use data

7th March 2025

Dear Reuben Degiorgio,

My name is Owen Fava and I am a student at the University of Malta, presently reading for a Master of Science in Artificial Intelligence. I am conducting a research study for my dissertation titled Advancing Automated Maltese Spell Checking Using Deep Learning. to develop a Maltese spell checker by fine-tuning a Large Language Model (LLM). This project is being conducted under the supervision of Prof. Alexiei Dingli.

I am hereby seeking your permission to use the data from verb.mt (https://verb.mt/) for data collection purposes. My data collection methods will involve using a web-scraper to extract all Maltese verb conjugates from the mentioned data source to build a corpus of correctly written text. A web scraper is a tool or script that automatically extracts data from websites by simulating human browsing and parsing the HTML content.

Participation will be entirely voluntary and participants will be free to withdraw at any point, without any repercussions. Data collected will be used to generate synthetic errors, producing an incorrect-correct dataset for fine-tuning the LLM.

Should you require further information, please do not hesitate to contact me; my contact details are provided below.

Thank you for your kind consideration of this request.

Sincerely,

Owen Fava
owen.fava.23.um.edu.mt

Authorisation to access and use data

I hereby grant permission for the collection and use of the data as described above.

Name of authoriser: Reuben Degiorgio

Signature: (signed)

Date: 7th March 2025

G.0.4 | Sagħtar

This subsection includes the signed consent form granting permission to use data from *Sagħtar* publications for this dissertation. The consent form is presented on the following page.

Continued on next page

L-Università ta' Malta

Request for permission to access and use data

12th March 2025

Dear Christopher Giordano,

My name is Owen Fava and I am a student at the University of Malta, presently reading for a Master of Science in Artificial Intelligence. I am conducting a research study for my dissertation titled Advancing Automated Maltese Spell Checking Using Deep Learning. to develop a Maltese spell checker by fine-tuning a Large Language Model (LLM). This project is being conducted under the supervision of Prof. Alexiei Dingli.

I am hereby seeking your permission to use the *Sagħtar* publications for data collection purposes. My data collection methods will involve automatic/manual extraction of Maltese words and sentences from the mentioned data source to build a corpus of correctly written text.

Participation will be entirely voluntary and participants will be free to withdraw at any point, without any repercussions. Data collected will be used to generate synthetic errors, producing an incorrect-correct dataset for fine-tuning the LLM.

Should you require further information, please do not hesitate to contact me; my contact details are provided below.

Thank you for your kind consideration of this request.

Sincerely,

Owen Fava
owen.fava.23.um.edu.mt

L-Università ta' Malta

Authorisation to access and use data

I hereby grant permission for the collection and use of the data as described above.

Name of authoriser: Chris Giordano

Signature: 

Date: 12th March 2025

G.0.5 | Additional Sources

This subsection outlines the additional data sources used in this dissertation and clarifies their approval and licensing status.

The sources listed in this appendix did not require formal approval for access or use. No specific licence has been attributed to these sources. Some materials were publicly accessible at the time of retrieval and may be used for research purposes. Access to the data did not require institutional permission, ethical clearance beyond standard academic research requirements, or paid licensing agreements.

In addition, some of the data sources were originally created, compiled, or maintained for purposes unrelated to this dissertation but have been used here for academic research and analysis.

Furthermore, certain datasets or materials were specifically compiled, or organised for the purpose of this dissertation. Where applicable, such materials were derived from publicly available information and structured solely to support the analytical objectives of this research.

All data were obtained from publicly available repositories, institutional websites, open databases, or other accessible platforms, as detailed below. The sources were accessed directly from their original publication platforms or official distributors.

The following is a list of the data sources and the locations from which they were obtained.

- IDS Maltese Dictionary – Sourced from: `https://ids.clld.org/`
- Correction of newspaper articles – Sent by Maltese department.
- Subset of manually corrected errors from MLRS Korpus Malti – Curated specifically for this dissertation.
- Qari tal-Provi (Debattista, 2022) – Sourced from: `https://github.com/AaronDebattista09/ICS5200/tree/main/QariTalProvi_Batch_1/Curated`
- Human collected error data (Busuttil, 2025) – Sourced from: `https://github.com/OFed22/maltesegec/tree/master/data/authentic/raw/busuttil`

These sources were used solely for academic and research purposes. No restricted, confidential, or proprietary data was accessed or collected in the process.

H

Discarded Experiments

This appendix provides a brief overview of some of the early fine-tuning experiments described in Section 4.4.3, which were conducted during the initial stages of model development and later discarded. It includes two summary tables: one for the word-level models (Table H.1) and one for the sentence-level models (Table H.2).

Table H.1: Overview of discarded word-level models developed during the early experimental phase. Source: Author.

Experiment	Approx. Dataset Size	Recorded Hyperparameters	Accuracy
1	Training: 922,473 Evaluation: 115,309 Test: Evaluation: 115,309	`learning_rate`: 2×10^{-4} `num_train_epochs`: 10 `per_device_train_batch`: 64 `per_device_eval_batch`: 128 `gradient_accumulation_steps`: 16	83%

Table H.2: Overview of discarded sentence-level models developed during the early experimental phase. Source: Author.

Experiment	Approx. Dataset Size	Recorded Hyperparameters	Accuracy
1	Training: 35,096 Evaluation: 4,387 Test: Evaluation: 4,387	`learning_rate`: 1×10^{-4} `num_train_epochs`: 7 `per_device_train_batch`: 8 `gradient_accumulation_steps`: 3	96.4%
2	Training: 35,096 Evaluation: 4,387 Test: Evaluation: 4,387	`r`: 8 `lora_alpha`: 16 `learning_rate`: 2×10^{-5} `num_train_epochs`: 4 `per_device_train_batch`: 8 `per_device_eval_batch`: 8 `gradient_accumulation_steps`: 3	53.32%
3	Training: 35,096 Evaluation: 4,387 Test: Evaluation: 4,387	`r`: 16 `lora_alpha`: 32 `learning_rate`: 2×10^{-5} `num_train_epochs`: 7 `per_device_train_batch`: 8 `per_device_eval_batch`: 8 `gradient_accumulation_steps`: 3	63%

References

Abdelaal, A., Medhat, A. A., Elsayad, M., Foad, M., Khaled, S., Tamer, A., & Medhat, W. (2024). Text correction for modern standard Arabic [6th International Conference on AI in Computational Linguistics]. *Procedia Computer Science*, *244*, 371–377. https://doi.org/https://doi.org/10.1016/j.procs.2024.10.211

Abdulrahman, R., & Hassani, H. (2022, June). A language model for spell checking of educational texts in Kurdish (Sorani). In M. Melero, S. Sakti, & C. Soria (Eds.), *Proceedings of the 1st annual meeting of the elra/isca special interest group on under-resourced languages* (pp. 189–198). European Language Resources Association. https://aclanthology.org/2022.sigul-1.25/

Ahmadi, S. (2021). Hunspell for Sorani Kurdish spell checking and morphological analysis. https://arxiv.org/abs/2109.06374

Ahmadzade, A., & Malekzadeh, S. (2021). Spell correction for Azerbaijani language using deep neural networks. https://doi.org/10.48550/arxiv.2102.03218

Ahmed, F., De Luca, E. W., & Nürnberger, A. (2009). Revised n-gram based automatic spelling correction tool to improve retrieval effectiveness. *Polibits*, *40*, 39–48. https://doi.org/10.17562/pb-40-6

Alahmari, S., Hall, L., Mouton, P., & Goldgof, D. (2024). Repeatability of fine-tuning large language models illustrated using QLoRA. *IEEE Access*, *Pp*, 1–1. https://doi.org/10.1109/access.2024.3470850

Allkivi-Metsoja, K., & Kippar, J. (2023, May). Spelling correction for Estonian learner language. In Alumäe, Tanel and Fishel, Mark (Ed.), *Proceedings of the 24th Nordic Conference on Computational Linguistics (NoDaLiDa)* (pp. 782–788). University of Tartu Library. https://aclanthology.org/2023.nodalida-1.79/

Ambili, T. R., PanchamiK, S., & Subash, N. (2016). Automatic error detection and correction in Malayalam. *International Journal For Science Technology And Engineering*, *3*, 92–96. https://api.semanticscholar.org/CorpusID:55975711

Arnardóttir, Þ., Ingólfsdóttir, S. L., Símonarson, H. B., Einarsson, H., Ingason, A. K., & Þorsteinsson, V. (2024, May). Beyond error categories: A contextual approach of evaluating emerging spell and grammar checkers. In M. Melero, S. Sakti, & C. Soria (Eds.), *Proceedings of the 3rd Annual Meeting of the Special Interest Group on Under-resourced Languages @ LREC-COLING 2024* (pp. 45–52). ELRA; ICCL. https://aclanthology.org/2024.sigul-1.6/

Arumugam, R., & Shanmugamani, R. (2018). *Hands-on natural language processing with Python: A practical guide to applying deep learning architectures to your NLP applications*. Packt Publishing Ltd.

Azzopardi-Alexander, M., & Borg, A. (2013, April). *Maltese* (0th ed.). Routledge. https://doi.org/10.4324/9780203192603

Bassil, Y. (2012). Parallel spell-checking algorithm based on Yahoo! N-grams dataset. https://doi.org/10.48550/arxiv.1204.0184

Bento, L. M. D. S., Sá, V. G. P. D., & Szwarcfiter, J. L. (2015). Some illustrative examples on the use of hash tables. *Pesquisa Operacional*, *35*(2), 423–437. https://doi.org/10.1590/0101-7438.2015.035.02.0423

Bezzina, J. (2020). Analiżi tal-iżbalji ortografiċi fil-komponimenti tal-primarja. [Thesis, University of Malta].

Bijoy, M. H., Hossain, N., Islam, S., & Shatabda, S. (2025). A transformer-based spelling error correction framework for Bangla and resource scarce Indic languages. *Computer Speech & Language*, *89*, 101703. https://doi.org/10.1016/j.csl.2024.101703

Bird, S., Klein, E., & Loper, E. (2009). *Natural Language Processing with Python*. O'Reilly.

Blasi, D., Anastasopoulos, A., & Neubig, G. (2022, May). Systematic inequalities in language technology performance across the world's languages. In S. Muresan, P. Nakov, & A. Villavicencio (Eds.), *Proceedings of the 60th Annual Meeting of the*

Association for Computational Linguistics (Volume 1: Long Papers) (pp. 5486–5505). Association for Computational Linguistics. https://doi.org/10.18653/v1/2022.acl-long.376

Blundell, D. (2014). Analiżi tal-iżbalji ortografiċi fil-bulettini tal-aħbarijiet televiżivi. [Thesis, University of Malta].

Boroujeni, M., Seddighin, M., & Seddighin, S. (2020). Improved algorithms for edit distance and LCS: Beyond worst case. In *Proceedings of the 2020 acm-siam symposium on discrete algorithms (soda)* (pp. 1601–1620). https://doi.org/10.1137/1.9781611975994.99

Bryant, C., Felice, M., & Briscoe, T. (2017, July). Automatic annotation and evaluation of error types for grammatical error correction. In R. Barzilay & M.-Y. Kan (Eds.), *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (pp. 793–805). Association for Computational Linguistics. https://doi.org/10.18653/v1/P17-1074

Bryant, C., Yuan, Z., Qorib, M. R., Cao, H., Ng, H. T., & Briscoe, T. (2023). Grammatical error correction: A survey of the state of the art. *Computational Linguistics*, *49*(3), 643–701. https://doi.org/10.1162/coli_a_00478

Buhagiar, M. (2021). Grammar and spell-checking techniques for Maltese. [Thesis, University of Malta].

Busuttil, A. (2025). Spell checking for the Maltese language. [Thesis, University of Malta].

Casha, R. (2004). *Spelling MT.* Retrieved October 1, 2024, from https://spelling.mt/check.html

Chaabi, Y., & Ataa Allah, F. (2022). Amazigh spell checker using Damerau-Levenshtein algorithm and n-gram. *Journal of King Saud University - Computer and Information Sciences*, *34*(8), 6116–6124. https://doi.org/10.1016/j.jksuci.2021.07.015

Chai, Y., Jin, L., Feng, S., & Xin, Z. (2024). Evolution and advancements in deep learning models for natural language processing. *Applied and Computational Engineering*, *77*(1), 144–149. https://doi.org/10.54254/2755-2721/77/20240674

Chang, Y., Wang, X., Wang, J., Wu, Y., Yang, L., Zhu, K., Chen, H., Yi, X., Wang, C., Wang, Y., Ye, W., Zhang, Y., Chang, Y., Yu, P. S., Yang, Q., & Xie, X. (2023). A survey on evaluation of large language models. https://doi.org/10.48550/arxiv.2307.03109

Cheekatimalla, L. B. (2025). Statistical modelling for natural language processing: Techniques, foundations and applications. *Ijarcce*, *14*(4), 251–258. https://doi.org/10.17148/ijarcce.2025.14433

Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation. https://doi.org/10.48550/arxiv.1406.1078

Chol David, M., & Balagadde Ssali, R. (2022). Prototyping NLP non-word detection system for Dinka using dictionary lookup approach. *International Journal on Natural Language Computing*, *11*(3), 31–41. https://doi.org/10.5121/ijnlc.2022.11304

Chung, J., Gulcehre, C., Cho, K., & Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. https://doi.org/10.48550/arxiv.1412.3555

Cissé, T. I., & Sadat, F. (2023). Automatic spell checker and correction for under-represented spoken languages: Case study on Wolof. *Proceedings of the Fourth workshop on Resources for African Indigenous Languages (RAIL 2023)*, 1–10. https://doi.org/10.18653/v1/2023.rail-1.1

Cissé, T. I., & Sadat, F. (2024, May). Advancing language diversity and inclusion: Towards a neural network-based spell checker and correction for Wolof. In R. Mabuya, M. Matfunjwa, M. Setaka, & M. van Zaanen (Eds.), *Proceedings of the Fifth Workshop on Resources for African Indigenous Languages @ LREC-COLING 2024* (pp. 140–151). ELRA; ICCL. https://aclanthology.org/2024.rail-1.16/

Damerau, F. J. (1964). A technique for computer detection and correction of spelling errors. *Communications of the ACM*, *7*(3), 171–176. https://doi.org/10.1145/363958.363994

de Araújo, S. S., Bezerra, B. L. D., de Sousa Neto, A. F., & Zanchettin, C. (2024). A proposal for post-OCR spelling correction using language models. *Latinx in AI@ NeurIPS 2024*.

Debattista, A. (2022). *Error-checking for Maltese: Deep learning for a low-resource scenario.* [Master's thesis, University of Malta].

Dettmers, T., Pagnoni, A., Holtzman, A., & Zettlemoyer, L. (2023). QLoRA: Efficient finetuning of quantized LLMs. *arXiv preprint arXiv:2305.14314*.

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019, June). BERT: Pre-training of deep bidirectional transformers for language understanding. In J. Burstein, C. Doran, & T. Solorio (Eds.), *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)* (pp. 4171–4186). Association for Computational Linguistics. https://doi.org/10.18653/v1/N19-1423

Dimech, D. (2013). Analiżi tal-iżbalji ortografiċi li saru fl-eżami tal-matrikola tal-Malti fl-2011. [Thesis, University of Malta].

Dixit, V., Dethe, S., & Joshi, R. K. (2005). Design and implementation of a morphology-based spellchecker for Marathi, and Indian language. *Archives Of Control Science*, *15*(3), 301.

Dong, G., Yuan, H., Lu, K., Li, C., Xue, M., Liu, D., Wang, W., Yuan, Z., Zhou, C., & Zhou, J. (2024). How abilities in large language models are affected by supervised fine-tuning data composition. https://arxiv.org/abs/2310.05492

El Atawy, S. M., & Abd ElGhany, A. (2018). Automatic spelling correction based on n-gram model. *International Journal of Computer Applications*, *182*(11), 5–9. https://doi.org/10.5120/ijca2018917724

Fabri, R. (2010). Maltese. *Revue belge de philologie et d'histoire*, *88*(3), 791–816. https://doi.org/10.3406/rbph.2010.7804

Flores, L. J. Y., & Radev, D. (2022). Look ma, only 400 samples! Revisiting the effectiveness of automatic n-gram rule generation for spelling normalization in Filipino. https://doi.org/10.48550/arxiv.2210.02675

Fredkin, E. (1960). Trie memory. *Communications of the ACM*, *3*(9), 490–499. https://doi.org/10.1145/367390.367400

Friendly, F. (2019). Jaro–Winkler distance improvement for approximate string search using indexing data for multiuser application. *Journal of Physics: Conference Series*, *1361*(1), 012080. https://doi.org/10.1088/1742-6596/1361/1/012080

Gamo, S. L., Gure, A. T., & Getechaw, B. D. (2024). Deep learning based model for a spell checker of Wolaita language. *2024 International Conference on Information and Communication Technology for Development for Africa (ICT4DA)*, 160–165. https://doi.org/10.1109/ict4da62874.2024.10777263

Gamu, D. T., & Woldeyohannis, M. M. (2023). Morphology-based spell checker for Dawurootsuwa language (S. Hussain, Ed.). *Scientific Programming*, *2023*, 1–10. https://doi.org/10.1155/2023/7625785

Gauci, P. (2024). Defying the monolingual mindset - Maltese as a language of identity and belonging. In S. Caruana (Ed.), *Malta and Italy: Language, Linguistics, Film and Literature. Writings in honour of Joseph M. Brincat* (pp. 149–163). Midsea Books Ltd.

Gers, F., & Schmidhuber, J. (2000). Recurrent nets that time and count. *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium*, *3*, 189–194 vol.3. https://doi.org/10.1109/ijcnn.2000.861302

Gers, F., Schmidhuber, J., & Cummins, F. (1999). Learning to forget: Continual prediction with LSTM. *1999 Ninth International Conference on Artificial Neural Networks ICANN 99. (Conf. Publ. No. 470)*, *2*, 850–855 vol.2. https://doi.org/10.1049/cp:19991218

Gezmu, A. M., Nürnberger, A., & Seyoum, B. E. (2018, May). Portable spelling corrector for a less-resourced language: Amharic. In Calzolari, Nicoletta and Choukri, Khalid and Cieri, Christopher and Declerck, Thierry and Goggi, Sara and Hasida, Koiti and Isahara, Hitoshi and Maegaard, Bente and Mariani, Joseph and Mazo, Hélène and Moreno, Asuncion and Odijk, Jan and Piperidis, Stelios and Tokunaga, Takenobu (Ed.), *Proceedings of the Eleventh International Conference on Lan-*

guage Resources and Evaluation (LREC 2018). European Language Resources Association (ELRA). https://aclanthology.org/L18-1651/

Gondaliya, Y., Kalariya, P., Panchal, B. Y., & Nayak, A. (2022). A rule-based grammar and spell checking. *SAMRIDDHI: A Journal of Physical Sciences, Engineering and Technology*, *14*(01), 48–54. https://doi.org/10.18090/samriddhi.v14i01.8

Goto, T., Sakai, Y., & Watanabe, T. (2025). GEC-Metrics: A unified library for grammatical error correction evaluation. https://arxiv.org/abs/2505.19388

Grattafiori, A., Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Mathur, A., Schelten, A., Vaughan, A., Yang, A., Fan, A., Goyal, A., Hartshorn, A., Yang, A., Mitra, A., Sravankumar, A., Korenev, A., Hinsvark, A., … Ma, Z. (2024). The LLaMA 3 herd of models. https://arxiv.org/abs/2407.21783

Graves, A., & Schmidhuber, J. (2005). Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Networks*, *18*(5–6), 602–610. https://doi.org/10.1016/j.neunet.2005.06.042

Greff, K., Srivastava, R. K., Koutnik, J., Steunebrink, B. R., & Schmidhuber, J. (2017). LSTM: A search space odyssey. *IEEE Transactions on Neural Networks and Learning Systems*, *28*(10), 2222–2232. https://doi.org/10.1109/tnnls.2016.2582924

Guo, Z., & Jin, P. (2020). SVM method and integrating pinyin and dictionary method for Chinese spelling errors detection. *2020 Prognostics and Health Management Conference (PHM-Besançon)*, 320–323. https://doi.org/10.1109/PHM-Besancon49106.2020.00062

Han, D., & Chang, B. (2013, October). A maximum entropy approach to Chinese spelling check. In L.-C. Yu, Y.-H. Tseng, J. Zhu, & F. Ren (Eds.), *Proceedings of the Seventh SIGHAN Workshop on Chinese Language Processing* (pp. 74–78). Asian Federation of Natural Language Processing. https://aclanthology.org/W13-4413/

Han, X., Zhang, Z., Ding, N., Gu, Y., Liu, X., Huo, Y., Qiu, J., Yao, Y., Zhang, A., Zhang, L., Han, W., Huang, M., Jin, Q., Lan, Y., Liu, Y., Liu, Z., Lu, Z., Qiu, X., Song, R., … Zhu, J. (2021). Pre-trained models: Past, present and future. *AI Open*, *2*, 225–250. https://doi.org/https://doi.org/10.1016/j.aiopen.2021.08.002

Hladek, D., Stas, J., & Juhar, J. (2013). Unsupervised spelling correction for Slovak. *Advances in Electrical and Electronic Engineering*, *11*(5), 392–397. https://doi.org/10.15598/aeee.v11i5.898

Hládek, D., Staš, J., & Pleva, M. (2020). Survey of automatic spelling correction. *Electronics*, *9*(10), 1670. https://doi.org/10.3390/electronics9101670

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, *9*(8), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

Hossain, N., Islam, S., & Huda, M. N. (2021). Development of Bangla spell and grammar checkers: Resource creation and evaluation. *IEEE Access*, *9*, 141079–141097. https://doi.org/10.1109/access.2021.3119627

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., & Chen, W. (2021). LoRA: Low-rank adaptation of large language models. https://arxiv.org/abs/2106.09685

Hu, Y., Zuo, X., Zhou, Y., Peng, X., Huang, J., Keloth, V. K., Zhang, V. J., Weng, R.-L., Chen, Q., Jiang, X., Roberts, K. E., & Xu, H. (2025). Information extraction from clinical notes: Are we ready to switch to large language models? https://arxiv.org/abs/2411.10020

Huang, W., Ma, X., Qin, H., Zheng, X., Lv, C., Chen, H., Luo, J., Qi, X., Liu, X., & Magno, M. (2024). How good are low-bit quantized LLaMA3 models? an empirical study. *CoRR*, *abs/2404.14047*. https://doi.org/10.48550/arXiv.2404.14047

Huang, W., Zheng, X., Ma, X., Qin, H., Lv, C., Chen, H., Luo, J., Qi, X., Liu, X., & Magno, M. (2024). An empirical study of LLaMA3 quantization: From LLMs to MLLMs. *Visual Intelligence*, *2*(1), 36. https://doi.org/10.1007/s44267-024-00070-x

Iqbal, S., Anwar, M. W., Bajwa, U. I., & Rehman, Z. (2013, October). Urdu spell checking: Reverse edit distance approach. In P. Bhattacharyya & M. G. A. Malik (Eds.), *Proceedings of the 4th Workshop on South and Southeast Asian Natural Language Processing* (pp. 58–65). Asian Federation of Natural Language Processing. https://aclanthology.org/W13-4707/

Jin, H., Wei, W., Wang, X., Zhang, W., & Wu, Y. (2023). Rethinking learning rate tuning in the era of large language models. https://arxiv.org/abs/2309.08859

Joshi, P., Santy, S., Budhiraja, A., Bali, K., & Choudhury, M. (2020, July). The state and fate of linguistic diversity and inclusion in the NLP world. In D. Jurafsky, J. Chai, N. Schluter, & J. Tetreault (Eds.), *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics* (pp. 6282–6293). Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.acl-main.560

Jurafsky, D., & Martin, J. H. (2025). *Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition with language models* (3rd) [Online manuscript released January 12, 2025]. https://web.stanford.edu/~jurafsky/slp3/

Kaalep, H.-J., Pirinen, F., & Moshagen, S. (2022). You can't suggest that?!: Comparisons and improvements of speller error models. *Nordlyd*, *46*(1). https://doi.org/10.7557/12.6349

Kalamkar, D., Mudigere, D., Mellempudi, N., Das, D., Banerjee, K., Avancha, S., Vooturi, D. T., Jammalamadaka, N., Huang, J., Yuen, H., Yang, J., Park, J., Heinecke, A., Georganas, E., Srinivasan, S., Kundu, A., Smelyanskiy, M., Kaul, B., & Dubey, P. (2019). A study of bfloat16 for deep learning training. https://arxiv.org/abs/1905.12322

Kamayani, M., Reinanda, R., Simbolon, S., Soleh, M., & Purwarianti, A. (2011). Application of document spelling checker for Bahasa Indonesia. *ICACSIS*.

Kapelles, D., Andriopoulos, A., & Koutsomitropoulos, D. (2025). Finetuning LLMs for grammatical error correction in English and Greek texts. *Proceedings of the 18th ACM International Conference on PErvasive Technologies Related to Assistive Environments*, 292–297. https://doi.org/10.1145/3733155.3733222

Kasmaiee, S., Kasmaiee, S., & Homayounpour, M. (2023). Correcting spelling mistakes in Persian texts with rules and deep learning methods. *Scientific Reports*, *13*(1), 19945. https://doi.org/10.1038/s41598-023-47295-2

Keita, M. K., Homan, C., Zampieri, M., Bremang, A., Alfari, H. A., Ibrahim, E. A., & Owusu, D. (2024). Grammatical error correction for low-resource languages: The case of Zarma. https://doi.org/10.48550/arxiv.2410.15539

Kernighan, M. D., Church, K. W., & Gale, W. A. (1990). A spelling correction program based on a noisy channel model. *COLING 1990 Volume 2: Papers presented to the 13th International Conference on Computational Linguistics*. https://aclanthology.org/C90-2036/

Khairul Islam, M. I., Meem, R. I., Abul Kasem, F. B., Rakshit, A., & Habib, M. T. (2019). Bangla spell checking and correction using edit distance. *2019 1st International Conference on Advances in Science, Engineering and Robotics Technology (ICASERT)*, 1–4. https://doi.org/10.1109/icasert.2019.8934536

Khalid, M., Yousaf, M. M., & Sadiq, M. U. (2022). Toward efficient similarity search under edit distance on hybrid architectures. *Information*, *13*(10), 452. https://doi.org/10.3390/info13100452

Klemen, M., Božič, M., Holdt, Š. A., & Robnik-Šikonja, M. (2024). Neural spell-checker: Beyond words with synthetic data generation. In *Text, Speech, and Dialogue* (pp. 85–96). Springer Nature Switzerland. https://doi.org/10.1007/978-3-031-70563-2_7

Kobayashi, M., Mita, M., & Komachi, M. (2024). Revisiting meta-evaluation for grammatical error correction. *Transactions of the Association for Computational Linguistics*, *12*, 837–855. https://doi.org/10.1162/tacl_a_00676

Kokong, D. A. R. S., Irmawati, B., & Dwiyansaputra, R. (2024). Spelling error correction in Indonesian using Damerau-Levenshtein distance dan n-gram. *Jurnal Teknologi Informasi, Komputer, dan Aplikasinya (JTIKA)*, *6*(1), 257–263. https://doi.org/10.29303/jtika.v6i1.169

Kukich, K. (1992). Techniques for automatically correcting words in text. *ACM Computing Surveys*, *24*(4), 377–439. https://doi.org/10.1145/146370.146380

Lee, J. (2025). Quantization-based jailbreaking vulnerability analysis: A study on performance and safety of the LLaMA3-8b-instruct model. *IEEE Access*, *13*, 136524–136535. https://doi.org/10.1109/access.2025.3594287

Levenshtein, V. I., et al. (1966). Binary codes capable of correcting deletions, insertions, and reversals. *Soviet physics doklady*, *10*(8), 707–710.

Li, Y., Song, L., & Hou, H. (2024, November). LoRAN: Improved low-rank adaptation by a non-linear transformation. In Y. Al-Onaizan, M. Bansal, & Y.-N. Chen (Eds.), *Findings of the Association for Computational Linguistics: EMNLP 2024* (pp. 3134–3143). Association for Computational Linguistics. https://doi.org/10.18653/v1/2024.findings-emnlp.177

Lin, A. (2019). Binary search algorithm. *WikiJournal of Science*, *2*(1), 5. https://doi.org/10.15347/wjs/2019.005

Liu, X., Cheng, F., Duh, K., & Matsumoto, Y. (2015). A hybrid ranking approach to Chinese spelling check. *ACM Transactions on Asian and Low-Resource Language Information Processing*, *14*(4), 1–17. https://doi.org/10.1145/2822264

Liyanapathirana, U., Gunasinghe, K., & Dias, G. (2021). SinSpell: A comprehensive spelling checker for Sinhala. https://doi.org/10.48550/arxiv.2107.02983

Lubis, A. R., Lase, Y. Y., Rahman, D. A., & Witarsyah, D. (2023). Improving spell checker performance for Bahasa Indonesia using text preprocessing techniques with deep learning models. *Ingénierie des systèmes d information*, *28*(5), 1335–1342. https://doi.org/10.18280/isi.280522

Lucas, C., & Čéplö, S. (2020). Maltese. In C. Lucas & S. Manfredi (Eds.), *Arabic and Contact–Induced Change* (pp. 265–302). Language Science Press. https://doi.org/10.5281/zenodo.3744525

Luitel, N., Bekoju, N., Sah, A. K., & Shakya, S. (2024). Contextual spelling correction with language model for low-resource setting. *2024 International Conference on Inventive Computation Technologies (ICICT)*, 582–589. https://doi.org/10.1109/icict60155.2024.10544712

Magueresse, A., Carles, V., & Heetderks, E. (2020). Low-resource languages: A review of past work and future challenges. https://arxiv.org/abs/2006.07264

Mangion, G. K. (1999). Spelling correction for Maltese. [Thesis, University of Malta].

Mangrulkar, S., Gugger, S., Debut, L., Belkada, Y., Paul, S., & Bossan, B. (2022). PEFT: State-of-the-art parameter-efficient fine-tuning methods.

Marier, S. M., Chen, X., Zhu, L., & Kong, X. (2025). Grammatical error correction for low-resource languages: A review of challenges, strategies, computational and future directions. *PeerJ Computer Science*, *11*, e3044. https://doi.org/10.7717/peerj-cs.3044

Markov, A. A. (1913). Essai d'une recherche statistique sur le texte du roman "Eugene Onegin" illustrant la liaison des épreuves en chaîne ["Example of a statistical investigation of the text of 'Eugene Onegin' illustrating the dependence between samples in chain"] [English translation by Morris Halle, 1956]. *Izvestia Imperatorskoi Akademii Nauk (Bulletin de l'Académie Impériale des Sciences de St.-Pétersbourg)*, *7*, 153–162.

Martynov, N., Baushenko, M., Kozlova, A., Kolomeytseva, K., Abramov, A., & Fenogenova, A. (2024, March). A methodology for generative spelling correction via natural spelling errors emulation across multiple domains and languages. In Y. Graham & M. Purver (Eds.), *Findings of the Association for Computational Linguistics: EACL 2024* (pp. 138–155). Association for Computational Linguistics. https://aclanthology.org/2024.findings-eacl.10/

Mati, D. N., Hamiti, M., Selimi, B., & Ajdari, J. (2021). Building spell-check dictionary for low-resource language by comparing word usage. *2021 44th International Convention on Information, Communication and Electronic Technology (MIPRO)*, 229–236. https://doi.org/10.23919/mipro52101.2021.9597183

Mengliev, D. B., Barakhnin, V. B., & Ibragimov, B. B. (2023). Rule-based syntactic analysis for Uzbek language: An alternative approach to overcome data scarcity and enhance interpretability. *2023 IEEE 24th International Conference of Young Professionals in Electron Devices and Materials (EDM)*, 1910–1915. https://doi.org/10.1109/edm58354.2023.10225235

Meta AI. (2024). Meta LLaMA 3 community license agreement. [Accessed: 21 October 2025]. https://www.llama.com/llama3/license/

Micallef, K., Gatt, A., Tanti, M., van der Plas, L., & Borg, C. (2022, July). Pre-training data quality and quantity for a low-resource language: New corpus and BERT

models for Maltese. In C. Cherry, A. Fan, G. Foster, G. (Haffari, S. Khadivi, N. (Peng, X. Ren, E. Shareghi, & S. Swayamdipta (Eds.), *Proceedings of the Third Workshop on Deep Learning for Low-Resource Natural Language Processing* (pp. 90–101). Association for Computational Linguistics. https://doi.org/10.18653/v1/2022.deeplo-1.10

Mikolov, T., Sutskever, I., Chen, K., Corrado, G., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. https://arxiv.org/abs/1310.4546

Mishra, R., & Kaur, N. (2013). A survey of spelling error detection and correction techniques. *International Journal of Computer Trends and Technology*, *4*(3), 372–374.

Mitton, R. (1996). *English spelling and the computer*. Longman.

Mizzi, R. (2000). The development of a statistical spell checker for Maltese. [Thesis, University of Malta].

Mohapatra, Y., Mishra, A. K., & Mishra, A. K. (2013). Spell checker for OCR. *International Journal of Computer Science and Information Technologies*, *4*(1), 91–97.

Mukazhanov, N., Alibiyeva, Z., Yerimbetova, A., Kassymova, A., & Alibiyeva, N. (2023). Development of an augmented Damerau–Levenshtein method for correcting spelling errors in Kazakh texts. *Eastern-European Journal of Enterprise Technologies*, *5*(2 (125)), 23–33. https://doi.org/10.15587/1729-4061.2023.289187

Napoles, C., Sakaguchi, K., Post, M., & Tetreault, J. (2015, July). Ground truth for grammatical error correction metrics. In C. Zong & M. Strube (Eds.), *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)* (pp. 588–593). Association for Computational Linguistics. https://doi.org/10.3115/v1/P15-2097

Napoles, C., Sakaguchi, K., & Tetreault, J. (2017). JFLEG: A fluency corpus and benchmark for grammatical error correction. https://arxiv.org/abs/1702.04066

Näther, M. (2020, May). An in-depth comparison of 14 spelling correction tools on a common benchmark. In N. Calzolari, F. Béchet, P. Blache, K. Choukri, C. Cieri,

T. Declerck, S. Goggi, H. Isahara, B. Maegaard, J. Mariani, H. Mazo, A. Moreno, J. Odijk, & S. Piperidis (Eds.), *Proceedings of the Twelfth Language Resources and Evaluation Conference* (pp. 1849–1857). European Language Resources Association. https://aclanthology.org/2020.lrec-1.228/

Naveed, H., Khan, A. U., Qiu, S., Saqib, M., Anwar, S., Usman, M., Akhtar, N., Barnes, N., & Mian, A. (2023). A comprehensive overview of large language models. https://doi.org/10.48550/arxiv.2307.06435

Ng, H. T., Wu, S. M., Briscoe, T., Hadiwinoto, C., Susanto, R. H., & Bryant, C. (2014, June). The CoNLL-2014 shared task on grammatical error correction. In H. T. Ng, S. M. Wu, T. Briscoe, C. Hadiwinoto, R. H. Susanto, & C. Bryant (Eds.), *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task* (pp. 1–14). Association for Computational Linguistics. https://doi.org/10.3115/v1/W14-1701

Nguyen, M. N., Baker, A., Neo, C., Roush, A., Kirsch, A., & Shwartz-Ziv, R. (2025). Turning up the heat: Min-p sampling for creative and coherent LLM outputs. https://arxiv.org/abs/2407.01082

Nicholas, G., & Bhatia, A. (2023). Lost in translation: Large language models in non-English content analysis. *arXiv (Cornell University)*. https://doi.org/10.48550/arxiv.2306.07377

Niranjan, A., Shaik, M. A. B., & Verma, K. (2020). Hierarchical attention transformer architecture for syntactic spell correction. https://arxiv.org/abs/2005.04876

Nistor, S. C., Moca, M., Moldovan, D., Oprean, D. B., & Nistor, R. L. (2021). Building a Twitter sentiment analysis system with recurrent neural networks. *Sensors*, *21*(7), 2266. https://doi.org/10.3390/s21072266

Nurhasanah, J. A., Susantiningdyah, H., Saputra, I., Prayoga, I. R. A., & Utomo, M. C. C. (2025). The grammar correction: A comparison of T5, LLAMA and ChatGPT. *Innovative Informatics and Artificial Intelligence Research*, *1*(1), 26–34. https://doi.org/10.35718/iiair.v1i1.1308

NVIDIA Corporation. (2023). *NVIDIA L40S Tensor Core GPU*. Retrieved November 4, 2025, from https://www.nvidia.com/en-us/data-center/l40s/

Olayiwola, A., Olayiwola, D., & Oyedeji, A. (2023). Development of an automatic grammar checker for Yorùbá word processing using government and binding theory. *Expert Systems with Applications*, *236*, 121351. https://doi.org/10.1016/j.eswa.2023.121351

Oliveri, F. (2025). Maltese spelling and error correction [Thesis, University of Malta].

Oluwaseyi, E., B, A. O., B, A. L., M, O. O., Abiodun, O., O, B. G., Badeji-Ajisafe, B., A, A. K., & A, S. O. (2024). Automatic spelling corrector for Yorùbá language using edit distance and n-gram language models. *2024 International Conference on Science, Engineering and Business for Driving Sustainable Development Goals (SEB4SDG)*, 1–6. https://doi.org/10.1109/seb4sdg60871.2024.10630121

Pal, A., & Mustafi, A. (2020). Vartani Spellcheck – automatic context-sensitive spelling correction of OCR-generated Hindi text using BERT and Levenshtein distance. https://doi.org/10.48550/arxiv.2012.07652

Papineni, K., Roukos, S., Ward, T., & Zhu, W.-J. (2002, July). BLEU: A method for automatic evaluation of machine translation. In P. Isabelle, E. Charniak, & D. Lin (Eds.), *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics* (pp. 311–318). Association for Computational Linguistics. https://doi.org/10.3115/1073083.1073135

Parankusham, K., Rizk, R., & Santosh, K. (2025). LakotaBERT: A transformer-based model for low resource lakota language. https://arxiv.org/abs/2503.18212

Pareja, A., Nayak, N. S., Wang, H., Killamsetty, K., Sudalairaj, S., Zhao, W., Han, S., Bhandwaldar, A., Xu, G., Xu, K., Han, L., Inglis, L., & Srivastava, A. (2024). Unveiling the secret recipe: A guide for supervised fine-tuning small LLMs. https://arxiv.org/abs/2412.13337

Parthasarathy, V. B., Zafar, A., Khan, A., & Shahid, A. (2024). The ultimate guide to fine-tuning LLMs from basics to breakthroughs: An exhaustive review of tech-

nologies, research, best practices, applied research challenges and opportunities. https://arxiv.org/abs/2408.13296

Patil, K. T., Bhavsar, R. P., & Pawar, B. V. (2021). Spelling checking and error corrector system for Marathi language text using minimum edit distance algorithm. In M. Singh, V. Tyagi, P. K. Gupta, J. Flusser, T. Ören, & V. R. Sonawane (Eds.), *Advances in Computing and Data Sciences* (pp. 102–111, Vol. 1440). Springer International Publishing. https://doi.org/10.1007/978-3-030-81462-5_10

Penteado, M. C., & Perez, F. (2023). Evaluating GPT-3.5 and GPT-4 on grammatical error correction for Brazilian Portuguese. https://arxiv.org/abs/2306.15788

Perumal, T., Mustapha, N., Mohamed, R., & Shiri, F. M. (2024). A comprehensive overview and comparative analysis on deep learning models. *Journal on Artificial Intelligence*, *6*(1), 301–360. https://doi.org/10.32604/jai.2024.054314

Peterson, J. L. (1980). Computer programs for detecting and correcting spelling errors. *Communications of the ACM*, *23*(12), 676–687. https://doi.org/10.1145/359038.359041

Phyu, E. T., Thu, Y. K., Chanlekha, H., Funakoshi, K., & Supnithi, T. (2024). Exploring the impact of error type features integration on transformer-based Myanmar spelling correction. *2024 21st International Joint Conference on Computer Science and Software Engineering (JCSSE)*, 365–372. https://doi.org/10.1109/jcsse61278.2024.10613711

Pirinen, T. A., & Lindén, K. (2014). State-of-the-art in weighted finite-state spell-checking. In A. Gelbukh (Ed.), *Computational Linguistics and Intelligent Text Processing* (pp. 519–532, Vol. 8404). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-54903-8_43

Post, M. (2018). A call for clarity in reporting BLEU scores. *Proceedings of the Third Conference on Machine Translation: Research Papers*, 186–191. https://www.aclweb.org/anthology/W18-6319

Radford, A., & Narasimhan, K. (2018). Improving language understanding by generative pre-training. https://api.semanticscholar.org/CorpusID:49313245

Rai, S. I., Maqsood, M., Hanif, B., Adam, M. A., Arslan, M., Shafiq, H., & Sijawal, M. (2024). Computational linguistics at the crossroads: A comprehensive review of NLP advancements. *World Journal of Advanced Engineering Technology and Sciences*, *11*(2), 578–591. https://doi.org/10.30574/wjaets.2024.11.2.0146

Rajan, A., Kalita, N., & Salgaonkar, A. (2025). Spell checker for low-resource Konkani language. https://doi.org/10.21203/rs.3.rs-6288011/v1

Randhawa, E., & Saroa, E. (2014). Study of spell checking techniques and available spell checkers in regional languages: A survey. *International Journal For Technological Research In Engineering*, *2*(3), 148–151.

Ransing, R., Dhamaskar, M. A., Rajpurohit, A., Dhoke, A., & Dalvi, S. (2024). Survey of pseudonymization, abstractive summarization & spell checker for Hindi and Marathi. https://doi.org/10.48550/arxiv.2412.18163

Rao, G. U. M., Kulkarni, A. P., & Chahoud, I. (2012). Telugu spell checker. https://api.semanticscholar.org/CorpusID:59652566

Remse, M., Mesgar, M., & Strube, M. (2016). Feature-rich error detection in scientific writing using logistic regression. *Proceedings of the 11th Workshop on Innovative Use of NLP for Building Educational Applications*, 162–171. https://doi.org/10.18653/v1/W16-0518

Ren, M., Cao, B., Lin, H., Liu, C., Han, X., Zeng, K., Wan, G., Cai, X., & Sun, L. (2024). Learning or self-aligning? Rethinking instruction fine-tuning. https://arxiv.org/abs/2402.18243

Reznik, Y. A. (2005). Analysis of a class of tries with adaptive multi-digit branching. In F. Dehne, A. López-Ortiz, & J.-R. Sack (Eds.), *Algorithms and Data Structures* (pp. 61–72, Vol. 3608). Springer Berlin Heidelberg. https://doi.org/10.1007/11534273_7

Rosner, M., & Borg, C. (2023). Language report Maltese. In G. Rehm & A. Way (Eds.), *European Language Equality* (pp. 183–186). Springer International Publishing. https://doi.org/10.1007/978-3-031-28819-7_27

Sakuntharaj, R., & Mahesan, S. (2018). Detecting and correcting real-word errors in Tamil sentences. *Ruhuna Journal of Science*, *9*(2), 150. https://doi.org/10.4038/rjs.v9i2.43

Sakuntharaj, R. (2022). Suggestion generation for erroneous words in Tamil documents using n-gram technique. *Asian Journal of Research in Computer Science*, 45–54. https://doi.org/10.9734/ajrcos/2022/v13i330317

Sakuntharaj, R., & Mahesan, S. (2021). Missing word detection and correction based on context of Tamil sentences using n-grams. *2021 10th International Conference on Information and Automation for Sustainability (ICIAfS)*, 42–47. https://doi.org/10.1109/ICIAfS52090.2021.9606025

Salhab, M., & Abu-Khzam, F. (2024). AraSpell: A deep learning approach for Arabic spelling correction. https://arxiv.org/abs/2405.06981

Salifou, L., & Naroua, H. (2014, July). Design and implementation of a spell checker for Hausa language (Étude et conception d'un correcteur orthographique pour la langue Haoussa) [in french]. In M. Mangeot & F. Sadat (Eds.), *TALN-RECITAL 2014 Workshop TALAf 2014: Traitement Automatique des Langues Africaines (TALAf 2014: African Language Processing)* (pp. 147–158). Association pour le Traitement Automatique des Langues. https://aclanthology.org/W14-6506/

Samsudin, N., & Hassen, H. R. (2023). Transformer-encoder generated context-aware embeddings for spell correction. In N. El Gayar, E. Trentin, M. Ravanelli, & H. Abbas (Eds.), *Artificial Neural Networks in Pattern Recognition* (pp. 129–139, Vol. 13739). Springer International Publishing. https://doi.org/10.1007/978-3-031-20650-4_11

Santoso, P., Yuliawati, P., Shalahuddin, R., & Wibawa, A. P. (2019). Damerau Levenshtein distance for Indonesian spelling correction. *Jurnal Informatika*, *13*(2), 11. https://doi.org/10.26555/jifo.v13i2.a15698

Sarkar, D. (2016). *Text Analytics with Python*. Apress. https://doi.org/10.1007/978-1-4842-2388-8

Schembri, C. M. (2016). Analiżi tal-iżbalji ortografiċi li saru fl-eżami tal-matrikola tal-Malti fil-livell avvanzat fl-2015. [Thesis, University of Malta].

Shannon, C. E. (1951). Prediction and entropy of printed English. *Bell System Technical Journal*, *30*(1), 50–64. https://doi.org/10.1002/j.1538-7305.1951.tb01366.x

Sharma, A., & Jain, P. (2013). Hindi spell checker. *Indian Institute of Technology Kanpur*.

Sheil, B. A. (1978). Median split trees: A fast lookup technique for frequently occuring keys. *Communications of the ACM*, *21*(11), 947–958. https://doi.org/10.1145/359642.359653

Shen, G., Tan, Q., Zhang, H., Zeng, P., & Xu, J. (2018). Deep learning with gated recurrent unit networks for financial sequence predictions. *Procedia Computer Science*, *131*, 895–903. https://doi.org/10.1016/j.procs.2018.04.298

Sheykholeslam, M. H., Minaei-Bidgoli, B., & Juzi, H. (2012). A framework for spelling correction in Persian language using noisy channel model. *Lrec*, 706–710.

Shiri, F. M., Perumal, T., Mustapha, N., & Mohamed, R. (2023). A comprehensive overview and comparative analysis on deep learning models: CNN, RNN, LSTM, GRU. https://doi.org/10.48550/arxiv.2305.17473

Singh, S., & Singh, S. (2020). Systematic review of spell-checkers for highly inflectional languages. *Artificial Intelligence Review*, *53*(6), 4051–4092. https://doi.org/10.1007/s10462-019-09787-4

Singh, S., & Singh, S. (2021). HINDIA: A deep-learning-based model for spell-checking of Hindi language. *Neural Computing and Applications*, *33*(8), 3825–3840. https://doi.org/10.1007/s00521-020-05207-9

Singh, S. P., Kumar, A., Singh, L., Bhargava, M., Goyal, K., & Sharma, B. (2016). Frequency based spell checking and rule based grammar checking. *2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT)*, 4435–4439. https://doi.org/10.1109/iceeot.2016.7755557

Soleh, M. Y., & Purwarianti, A. (2011). A non word error spell checker for Indonesian using morphologically analyzer and HMM. *Proceedings of the 2011 International Conference on Electrical Engineering and Informatics*, 1–6. https://doi.org/10.1109/iceei.2011.6021514

Sooraj, S., Manjusha, K., Anand Kumar, M., & Soman, K. (2018). Deep learning based spell checker for Malayalam language. *Journal of Intelligent & Fuzzy Systems*, *34*(3), 1427–1434. https://doi.org/10.3233/jifs-169438

Spagnol, M., Comrie, B., & Bibiko, H.-J. (2020, November). *Maltese IDS wordlist by Michael Spagnol and Bernard Comrie* (Version v1.0). Zenodo. https://doi.org/10.5281/zenodo.4280596

Stymne, S., Pettersson, E., Megyesi, B., & Palmér, A. (2017, May). Annotating errors in student texts: First experiences and experiments. In E. Volodina, G. Grigonytė, I. Pilán, K. N. Björkenstam, & L. Borin (Eds.), *Proceedings of the Joint Workshop on NLP for Computer Assisted Language Learning and NLP for Language Acquisition* (pp. 47–60). LiU Electronic Press. https://aclanthology.org/W17-0306/

Suresh, S. K., & P, S. (2024). Towards smaller, faster decoder-only transformers: Architectural variants and their implications. https://doi.org/10.48550/arxiv.2404.14462

Tarniceriu, A., Rimoldi, B., & Dillenbourg, P. (2015). HMM-based error correction mechanism for five-key chording keyboards. *2015 International Symposium on Signals, Circuits and Systems (ISSCS)*, 1–4. https://doi.org/10.1109/isscs.2015.7203984

Telenti, A., Auli, M., Hie, B. L., Maher, C., Saria, S., & Ioannidis, J. P. A. (2024). Large language models for science and medicine. *European Journal of Clinical Investigation*, *54*(6), e14183. https://doi.org/10.1111/eci.14183

Tesfaye, D. (2011). A rule-based Afan Oromo grammar checker. *International Journal of Advanced Computer Science and Applications*, *2*(8). https://doi.org/10.14569/ijacsa.2011.020823

Thapa, S., Pradhan, A., Kayastha, D., Lawaju, A., Bahadur Rana, P., & Basnet, M. (2025). Prasta Nepali: A transformer based approach for automated Nepali grammar (Byakaran) error detection and correction. *2025 International Conference on Inventive Computation Technologies (ICICT)*, 1327–1334. https://doi.org/10.1109/icict64420.2025.11004703

Tolegenova, A. (2024). Automatic error correction: Evaluating performance of spell checker tools. *Suleyman Demirel University Bulletin Natural and Technical Sciences*, *58*(1), 15–21. https://doi.org/10.47344/sdubnts.v58i1.690

Tolentino, H. D., Matters, M. D., Walop, W., Law, B., Tong, W., Liu, F., Fontelo, P., Kohl, K., & Payne, D. C. (2007). A UMLS-based spell checker for natural language processing in vaccine safety. *BMC Medical Informatics and Decision Making*, *7*(1), 3. https://doi.org/10.1186/1472-6947-7-3

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention is all you need. https://doi.org/10.48550/arxiv.1706.03762

von Werra, L., Belkada, Y., Tunstall, L., Beeching, E., Thrush, T., Lambert, N., Huang, S., Rasul, K., & Gallouédec, Q. (2020). TRL: Transformer Reinforcement Learning.

Wang, P., Jia, Z., & Zhao, H. (2014). Grammatical error detection and correction using a single maximum entropy model. *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task*, 74–82. https://doi.org/10.3115/v1/W14-1710

Wang, S., & Jiang, J. (2015). Learning natural language inference with LSTM. https://doi.org/10.48550/arxiv.1512.08849

Weerawardhena, S., Kassianik, P., Nelson, B., Saglam, B., Vellore, A., Priyanshu, A., Vijay, S., Aufiero, M., Goldblatt, A., Burch, F., Li, E., He, J., Kedia, D., Oshiba, K., Yang, Z., Singer, Y., & Karbasi, A. (2025). LLaMA-3.1-FoundationAI-SecurityLLM-8B-Instruct technical report. https://arxiv.org/abs/2508.01059

Wei, M., & Liu, X. (2024, July). *ClinMED-LLaMA3 a large model of clinical medicine based on LLaMA3*. https://doi.org/10.21203/rs.3.rs-4703651/v1

Wojcicki, P., & Zientarski, T. (2024). Polish word recognition based on n-gram methods. *IEEE Access*, *12*, 49817–49825. https://doi.org/10.1109/access.2024.3385113

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., Von Platen, P., Ma, C., Jernite, Y.,

Plu, J., Xu, C., Le Scao, T., Gugger, S., … Rush, A. (2020). Transformers: State-of-the-art natural language processing. *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, 38–45. https://doi.org/10.18653/v1/2020.emnlp-demos.6

Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., Klingner, J., Shah, A., Johnson, M., Liu, X., Kaiser, Ł., Gouws, S., Kato, Y., Kudo, T., Kazawa, H., … Dean, J. (2016). Google's neural machine translation system: Bridging the gap between human and machine translation. https://arxiv.org/abs/1609.08144

Xia, Y., Kim, J., Chen, Y., Ye, H., Kundu, S., Hao, C., & Talati, N. (2024). Understanding the performance and estimating the cost of LLM fine-tuning. https://doi.org/10.48550/arxiv.2408.04693

Yang, Z., Pilian, H., Wei, X., & Mu, L. (2006, October). *Discriminative reranking for spelling correction* [Doctoral dissertation, Waseda University]. https://waseda.repo.nii.ac.jp/records/28956

Zammit, M. R. (2017). Maltese. In G. Kanarakis (Ed.), *The Legacy of the Greek Language* (pp. 635–658). Peridot International Publications.

Żammit, J. (2023). Harnessing the power of ChatGPT for mastering the Maltese language: A journey of breaking barriers and charting new paths. In A. Adadi & S. Motahhir (Eds.), *Machine Intelligence for Smart Applications* (pp. 161–178, Vol. 1105). Springer Nature Switzerland. https://doi.org/10.1007/978-3-031-37454-8_8

Zhang, S., Dong, L., Li, X., Zhang, S., Sun, X., Wang, S., Li, J., Hu, R., Zhang, T., Wu, F., & Wang, G. (2025). Instruction tuning for large language models: A survey. https://arxiv.org/abs/2308.10792

Zhang, X., Rajabi, N., Duh, K., & Koehn, P. (2023, December). Machine translation with large language models: Prompting, few-shot learning, and fine-tuning with QLoRA. In P. Koehn, B. Haddow, T. Kocmi, & C. Monz (Eds.), *Proceedings of the Eighth Conference on Machine Translation* (pp. 468–481). Association for Computational Linguistics. https://doi.org/10.18653/v1/2023.wmt-1.43

Zhang, X., Mao, R., & Cambria, E. (2023). A survey on syntactic processing techniques. *Artificial Intelligence Review*, *56*(6), 5645–5728. https://doi.org/10.1007/s10462-022-10300-7

Zhao, C., & Sahni, S. (2019). String correction using the Damerau-Levenshtein distance. *BMC Bioinformatics*, *20*(S11), 277. https://doi.org/10.1186/s12859-019-2819-0

Zhao, C., & Sahni, S. (2020). Linear space string correction algorithm using the Damerau-Levenshtein distance. *BMC Bioinformatics*, *21*(S1), 4. https://doi.org/10.1186/s12859-019-3184-8

Zhao, J., Liu, H., Bao, Z., Bai, X., Li, S., & Lin, Z. (2017, December). N-gram model for Chinese grammatical error diagnosis. In Y.-H. Tseng, H.-H. Chen, L.-H. Lee, & L.-C. Yu (Eds.), *Proceedings of the 4th Workshop on Natural Language Processing Techniques for Educational Applications (NLPTEA 2017)* (pp. 39–44). Asian Federation of Natural Language Processing. https://aclanthology.org/W17-5907/

Zhu, Q., Ma, X., & Li, X. (2019). Statistical learning for semantic parsing: A survey. *Big Data Mining and Analytics*, *2*(4), 217–239. https://doi.org/10.26599/bdma.2019.9020011