applied sciences

MDPI

Article

Towards the Natural Language Processing as Spelling Correction for Offline Handwritten Text Recognition Systems

Arthur Flor de Sousa Neto [1], Byron Leite Dantas Bezerra [1,*] and Alejandro Héctor Toselli [2]

[1] Escola Politécnica de Pernambuco, Universidade de Pernambuco, Recife 50720-001, Brazil; afsn@ecomp.poli.br

[2] PRHLT Research Center, Universitat Politécnica de Valéncia, 46022 Valencia, Spain; ahector@prhlt.upv.es

* Correspondence: byronleite@ecomp.poli.br; Tel.: +55-81-3184-7548

Received: 16 September 2020; Accepted: 27 October 2020; Published: 31 October 2020

Abstract: The increasing portability of physical manuscripts to the digital environment makes it common for systems to offer automatic mechanisms for offline Handwritten Text Recognition (HTR). However, several scenarios and writing variations bring challenges in recognition accuracy, and, to minimize this problem, optical models can be used with language models to assist in decoding text. Thus, with the aim of improving results, dictionaries of characters and words are generated from the dataset and linguistic restrictions are created in the recognition process. In this way, this work proposes the use of spelling correction techniques for text post-processing to achieve better results and eliminate the linguistic dependence between the optical model and the decoding stage. In addition, an encoder–decoder neural network architecture in conjunction with a training methodology are developed and presented to achieve the goal of spelling correction. To demonstrate the effectiveness of this new approach, we conducted an experiment on five datasets of text lines, widely known in the field of HTR, three state-of-the-art Optical Models for text recognition and eight spelling correction techniques, among traditional statistics and current approaches of neural networks in the field of Natural Language Processing (NLP). Finally, our proposed spelling correction model is analyzed statistically through HTR system metrics, reaching an average sentence correction of 54% higher than the state-of-the-art method of decoding in the tested datasets.

Keywords: deep learning; offline handwritten text recognition; natural language processing; encoder–decoder model; spelling correction

1. Introduction

Writing is an essential communication and documentation tool worldwide. Currently, in the digital age, the integration of physical manuscripts in the technological environment is becoming common, in which machines can understand the text of scanned images through the process of handwriting recognition and represent them in the digital context for later use [1]. Historical manuscripts [2], medical prescriptions [3], documents [4], and general forms [5] are some scenarios that require manual effort to digitize and transcribe content into the digital environment through Optical Character Recognition (OCR) technologies [6].

In this field, OCR systems have two categories: (i) online, in which the input information is obtained in real time through sensors; and (ii) offline, which obtains data from static scenarios, as in the case of images [7]. Still, in the offline category, there is recognition of printed text and manuscript [8]. Unlike the printed text scenario, offline Handwritten Text Recognition (HTR) is more complex to achieve its goal, as for the same writer there are numerous variations in a single sentence [8].

Fortunately, HTR systems have evolved considerably since the use of the Hidden Markov Model (HMM) for text recognition [2,9–11]. Currently, with the use of Deep Neural Networks (Deep Learning), it is possible to more assertively perform the recognition process at different levels of text segmentation that is, character [12], word [13,14], line [15] and even the paragraph [16] levels. However, in scenarios with an unrestricted dictionary, they still do not achieve satisfactory results [6] and, to minimize this problem, it is common to perform text decoding in conjunction with post-processing using Natural Language Processing (NLP) techniques [17], specifically the statistical approach [18].

In this context, tools such as Stanford Research Institute Language Model (SRILM) [19] and Kaldi [20] have gained space in the last years, by performing text decoding through a language model. In fact, these two tools have become the most commonly used currently in HTR systems and the results of optical model are improved in this step [2,15,21]. In addition, through statistical methods, it is essential to create and use a structured character dictionary based on the dataset used or external corpora [18]. Thus, text decoding in an HTR system becomes restricted to this dictionary, which, in turn, has correlation to the dataset, causing a limitation in its application in new text scenarios, especially in multi-language systems.

On the other hand, fields of study in NLP, such as Machine Translation [22,23] and Grammatical Error Correction [24,25], which work with text processing, classification, and correction, have brought promising results with neural networks approaches in recent years [26–28]. The application of encoder–decoder models, also known as Sequence to sequence [26,29], has grown considerably in the field of NLP for applications that require huge linguistic knowledge and, many times, statistical approaches have limitations of linguistic context [29]. In addition, these models were extended with the Attention mechanism [30,31], achieving even better results, and, more recently, presented models based entirely on Attention [22,32,33].

Therefore, the main goal of this work is to apply alternative spelling correction techniques in the post-processing of an HTR system (at line level and free segmentation level), in order to obtain competitive results to the traditional method of decoding and disconnect the stages of the recognition process. In other words, to enable the HTR system to integrate with any post-processing approach, regardless of dictionary between both systems.

In this paper, we apply and analyze applications of techniques with a focus on spelling correction, varying between statistical approaches [2,18,34] and the most recent with neural networks in the field of linguistics, such as Sequence to sequence with Attention mechanism [30,31] and Transformer [22] models. For a better analysis in different data scenarios, we used five datasets in the experiment: Bentham [35]; Department of Computer Science and Applied Mathematics (Institut für Informatik und Angewandte Mathematik, IAM) [36]; Recognition and Indexing of Handwritten Documents and Faxes (Reconnaissance et Indexation de données Manuscrites et de fac similÉS, RIMES) [37]; Saint Gall [38]; and Washington [39]. In addition, we also use three optical models as HTR system: Bluche [40]; Puigcerver [15]; and the proposed model. In this way, we created a wide variety of combinations between datasets and applied techniques, creating various analysis workflows. An open source (https://github.com/arthurflor23/handwritten-text-recognition, https://github.com/arthurflor23/spelling-correction) implementation for the reproducibility of results is also provided upon request.

The remaining of this article is organized as follows: in Section 2, the process of the offline Handwritten Text Recognition system is detailed. Then, in the Section 3, the spelling correction process is presented through statistical and neural networks approaches. In Section 4, the overall best HTR system produced in our analysis is presented, detailing the optical model and the proposed encoder–decoder model for spelling correction. In Section 5, the methodology and experimental setup are explained. In Section 6, the results obtained from the experiment in each dataset and technique are presented. In Section 7, the results are interpreted and discussed. Finally, Section 8 presents the conclusions that summarize the article.

2. Offline Handwritten Text Recognition

Offline Handwritten Text Recognition (HTR) has evolved in the past few decades for two reasons: (i) the use of training and recognition concepts and techniques previously developed in the field of Automatic Speech Recognition (ASR); and (ii) the growing number of publicly available datasets for training and testing. In this way, the optical models in HTR systems are generally associated with language models, usually at the character or word level, to make the text recognition plausible [2,41].

The most traditional approaches to HTR are based on N-gram language models (statistical approach) and Hidden Markov Model (HMM) optical modeling with gaussian mixture emission distributions [42], most recently improved with emission probabilities by multilayer perceptrons [21]. However, notable improvements in HTR recognition accuracy have been achieved through artificial neural networks as optical models, specifically the Convolutional and Recurrent Neural Network (CRNN), becoming the current state-of-the-art [2,15,40].

This current approach, detailed in Figure 1, uses images as input data and has final recognized texts as output data. Then, the defined steps are: (i) image preprocessing and normalization; (ii) image features extraction through Convolutional Neural Network (CNN); (iii) selected features propagation over the sequence with a Recurrent Neural Network (RNN); and (iv) the loss function calculation (to conduct model training) through the Connectionist Temporal Classification (CTC) [43], combined to the traditional language model (dictionary included) and HMM to achieve even better accuracy through the text decoding with linguistic knowledge built from dataset [2,44,45].

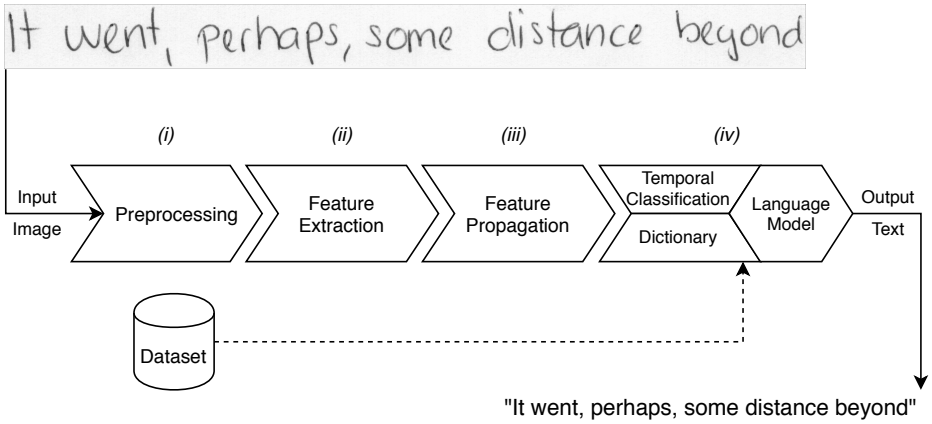

Figure 1. Workflow of the offline handwritten text recognition system.

2.1. Image Preprocessing

The cursive style varies depending on the writer and has unique characteristics from one person to another. Not only that, but external factors also directly influence the way of writing, such as the writing instrument used, the emotional state of the writer, and the time available [6]. This great variability leads to the application of image processing techniques to normalize these irregularities [8,21]. In these techniques, contrast correction, slant correction and image standardization are commonly used [21,46].

The first strategy, contrast correction, is adopted to reduce lighting irregularities in the image, such as light peaks and excessive shadows. Thus, techniques such as illumination compensation are efficient for this scenario, since they also remove background and foreground noises through lighting [47]. The second, slant correction or dislant, aims to normalize the vertical inclination of the characters. The method detects the slant angle of the writing on the vertical axis and then applies a geometric image transformation to adjust the angle detected [48]. The last strategy, standardization (or Z-score normalization), aims to rescale the image features (pixels) to result in an image with pixels with mean and variance values equal to 0 and 1, respectively [49].

2.2. Optical Model

The main component of the HTR system, the optical model, is represented by the steps following preprocessing. It is responsible for observing an image, understanding the context, and decoding it in text [21]. Currently, using Convolutional Recurrent Neural Networks (CRNN) with Connectionist Temporal Classification (CTC), the state-of-the-art optical model is subdivided into three steps [2,49].

In the first step, the input image, already preprocessed, feeds the CNN layers to extract features.This process can be done through the traditional CNN layers [2,15] or using the Gated mechanism [40,50], in which it aims to extract even more relevant information.

The second step uses the sequence of features extracted within RNN layers to map and generate a matrix containing character probabilities for each time-step (positions in the sequence). In this process, Bidirectional Long Short-Term Memory (BLSTM) and Bidirectional Gated Recurrent Unit (BGRU) layers are generally used to better handle large text sentences and their contexts [15].

Finally, in the third step, the probability matrix has two processing ways through the CTC: (i) calculate the loss function with the loss algorithm CTC in order to adjust the weights for learning the model (training phase); and text decoding (ii) using an CTC decoding algorithm (classification phase) [43,49].

2.3. Text Decoding with Linguistic Knowledge

We can perform text recognition of images for digital medium only using the character predictions generated by the optical model, specifically by CTC decoding [44]. However, from a linguistic perspective, we have an initial understanding of the context and what results the HTR system should return, which means in a set of predefined linguistic context about the texts to be decoded [18].

A conventional approach to modeling this contextual information is through the language model and its dictionary (vocabulary). This component is responsible for defining the set of words that the system can recognize and the words that have not been defined in this set are considered out-of-vocabulary [51]. Thus, depending on the context, the size of the dictionary can vary between small (dozens), medium (hundreds), and large (thousands) of defined words, in the latter case, it is considered an unrestricted text recognition [51].

Statistical approaches, specifically N-gram models, remains conventional to HTR systems [52–54]. The N-gram language model consists of a probabilistic distribution of characters or words, in which the probability of a term depends on the previous terms. The N-gram model follows a Markovian assumption to make it possible approximate the probability of a term, given the previous $n-1$ terms [55].

Currently, a common decoder in HTR systems, optical model output, and language model, are integrated by means of the Hidden Markov Model (HMM), which disregards the CTC decoding step and takes into account only the RNN output [2,21,56,57]. The role of HMM in a decoding process is to recover the most likely sequence of states that produce a given observed sequence [58]. Regarding the HTR context, the HMM retrieves the most likely sequence of words, given the sequence of the observed characteristic vector [21,57].

3. Spelling Correction

A spelling checker system, in most cases, checks the context of surrounding words, to only then present the possible error and its suggestions for correction. A spelling corrector goes one step further and automatically chooses the most likely word [59]. The spelling corrector has three approaches to deal with sentence correction: (i) uses a word frequency dictionary and works by entering one word of the sentence at a time [60]; (ii) uses a statistical language model and certain linguistic rules to learn the probability distribution of words [61]; and (iii) uses neural networks to build deep linguistic knowledge [62].

The correction steps are defined in: (i) preprocessing of the text in order to normalize the sentences; (ii) error detection through the frequency dictionary, language model or neural network. In addition, if there is no error in the sentence, the next steps are ignored; (iii) generation of candidate words for correction from the detected error, considering the correction with the most likely word in the candidate list [63]. Figure 2 shows the workflow of the spelling correction system.

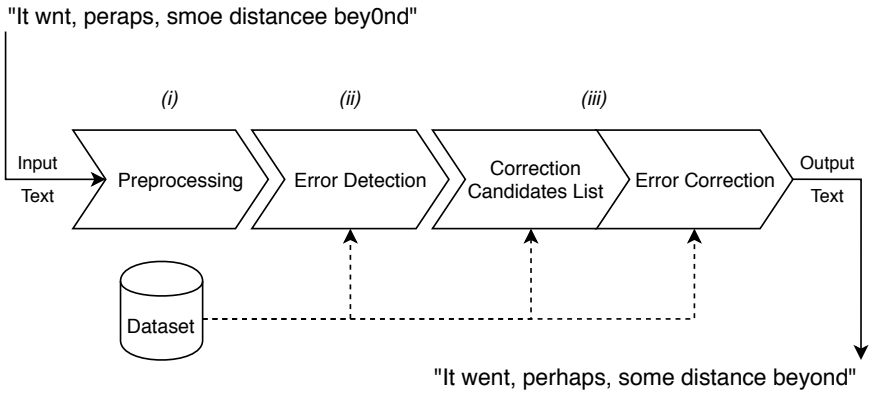

Figure 2. Workflow of the spelling correction system.

3.1. Text Preprocessing

Initially, to normalize a text, it is important to define what counts as a valid word (token) within a computer-readable text or speech collection (corpus, plural corpora) [64]. Tokens can count (or not) punctuation marks, depending on the final application [59]. Among the text preprocessing methods, we can highlight: (i) tokenization, task of breaking a sentence into words or characters; (ii) filtering, task to remove words of little information (stopwords); (iii) lemmatization, the task of grouping several inflected forms of the same word so that they can be analyzed as a single token; and (iv) stemming, the task of obtaining the stem of derived words, also as a way of analyzing it as a single token [65].

However, for most NLP applications, we need to preserve as much information as possible from the text, so only tokenization is used [59]. In this context, the well-known Penn Treebank Tokenization standard [66] is commonly used for corpora analysis of datasets. This standard separates out clitics ("doesn't" becomes "does" plus "n't"), keeps hyphenated words together, and separates out all punctuation.

Therefore, text preprocessing, or normalization, is the most important step in the spelling correction process for standardizing text input. This step influences the entire performance of the system, since any error in the tokenization, for example, can be equivalent to an unsuccessful spelling error detection and correction, or create noise in the construction of linguistic knowledge in a language model or neural network [59,67].

3.2. Error Detection

The spelling error detection process consists of verifying, given an input text, if it is considered a valid index or not. In a way, efficient techniques have been developed to detect the various types of spelling errors; however, the two most known techniques for this task are *N*-gram analysis and dictionary lookup, which are most used by text recognition systems and spell checkers, respectively [68,69].

The first approach, *N*-grams, are sub-strings of tokens with *N*-sequences. The detection process works by counting each *N*-gram probability of the input text and looking it up in a pre-built table of *N*-gram statistics. If a non-existent or rare *N*-gram is found, the token is flagged as a misspelling, otherwise not [70]. The second approach, dictionary, is structured from a given corpus and group words to avoid repetition (called word-types or types), and it is used to check each word in the input text for its presence in the dictionary. If that word is present in the dictionary, it is a correct word;

otherwise, it is flagged as a misspelling [59]. Furthermore, the frequency of words can also be integrated into the raw dictionary to increase vocabulary knowledge and create ranges for candidate words based on the value of occurrences. In this way, word-types are ordered by frequency of occurrence [71,72].

3.3. Error Correction

After detecting a spelling error in a text, the correction process consists of generating a correction candidates list and ranking them. The construction of this ranked list usually invokes some measure of probabilistic estimate between the incorrect sequence and the possible candidates [68,69].

In this context, the most studied algorithms for spelling correction are: (i) rule-based techniques [73], for which the candidate generation process consists of applying all the rules applicable to an incorrect sequence and keeping all valid words; (ii) similarity keys [74], which maps all character strings to a key, in which strings with similar spellings have similar keys; (iii) edit distance [75], which considers the three editing operations (insertion, exclusion, substitution) for one string to be converted into the other, and then to rank candidates based on the minimum number of operations performed; (iv) N–gram with probabilistic techniques (language model) [76,77], represent that a given character will be followed by another, in which they can be estimated by collecting N-gram frequency statistics in a large text corpus [69]; and finally, (v) neural networks [78], the most recent approach.

3.4. Encoder–Decoder Neural Network Model

The encoder–decoder model, also known as Sequence to sequence (Seq2seq) [26,29], is an approach of Neural Networks that consists of two sequential Recurrent Neural Networks (RNNs). The first RNN is responsible for encoding an input of variable-length symbol sequences into a fixed-length representation, while the second RNN decodes this fixed-length representation into another variable-length symbol sequences.

A potential problem with the traditional Seq2seq approach is that a neural network needs to be able to compress all the information needed from an input sentence to decode it later. This can make it difficult for the model to process very long sentences, especially those that are longer than training sentences [30]. According to [30], Attention mechanism allows the model to learn how to generate a context vector that best represents the input sentence at each stage of decoding; this means that the model learns to focus and pay more attention to the relevant parts of the input sequence.

Bahdanau's Attention (additive style) [30], and Luong's Attention (multiplicative style) [31] are considered the major approaches in the Seq2seq models. Both mechanisms follow the same application, slightly changing the score vectors calculation:

$$score(h_t, h_s) = \begin{cases} h_t^\top W_a h_s & \text{Multiplicative} \\ v_a^\top tanh(W_a h_t + U_a h_s) & \text{Additive} \end{cases} \quad (1)$$

where h_t is the encoder output, h_s is the hidden state and v_a, W_a, U_a are weights' matrices. Thus, after the encoder produces hidden states of each element in the input sequence, the steps are defined: (i) the alignment scores (attention weights) are calculated between the hidden state of the previous decoder and the hidden states of each encoder, where T_x denotes the length of source:

$$\alpha_{ts} = \frac{exp(score(h_t, h_s))}{\sum_{k=1}^{T_x} exp(score(h_t, h_k))} \quad (2)$$

(ii) the context vector is calculated by multiplying the alignment scores by their respective hidden states of the encoder:

$$c_t = \sum_s \alpha_{ts} h_s \quad (3)$$

(iii) the context vector is concatenated with the previous decoder output (attention vector), feeding the decoder at the current time-step to produce a new output:

$$a_t = f(c_t, h_t) = tanh(W_c[c_t; h_t]) \tag{4}$$

These steps repeat themselves for each decoder time-step until the decoding process ends ("<EOS>" special character or maximum length).

Most recently, a mechanism, called Multi-Head Attention, was proposed in a model architecture composed entirely of attention layers, the Transformer model [22], which makes use of Scaled Dot-Product Attention to relate different positions of a single sentence and calculate a representation. In this way, the Attention mechanism has the function of mapping vectors of a query and a set of key-value pairs, in an output.

The output is calculated as a weighted sum of the values, where the weight assigned to each value is calculated by a query compatibility function with the corresponding key. Then, the input data consist of queries and keys of dimension d_k and values of dimension d_v. To obtain the weights of the values, a softmax function is applied to the dot product of the query with all the keys, divided each by $\sqrt{d_k}$. The output matrix is defined as bellow:

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{5}$$

where the attention function computes a set of queries simultaneously, packed into a Q matrix. The keys and values are also packed into K and V matrices, respectively. Then, the Multi-Head Attention allows for jointly attending information from different representation subspaces in different positions through a attention heads, where W_i^Q, W_i^K, W_i^V are weights matrices:

$$head_i = Attention(QW_i^Q, KW_i^K, VW_i^V) \tag{6a}$$

$$MultiHead(Q, K, V) = Concat(head_1, ..., head_h)W^O \tag{6b}$$

Therefore, this mechanism allows the weights to be calculated in parallel in the model training instead of the traditional encoder–decoder, being the current approach in the field of linguistics [22,32,33]. However, it tends to be slower in the inference phase, since the decoding process is performed through the projection of subspaces; this means that a new context vector is calculated for each iteration over time-steps [22].

4. The Handwritten Text Recognition Proposed System

In this section, an overview of the proposed model architectures is described. At first, the optical model is detailed according to the offline Handwritten Text Recognition system. Then, the encoder–decoder model is detailed according to the Spelling Correction system.

4.1. Optical Model

The optical model architecture, responsible for recognizing the text from the images, is composed of the convolutional block followed by the recurrent block. Then, the proposed model uses gated mechanism presented by [50] in the convolutional layers, and make use of recent deep learning approaches.

The convolutional block is composed by: (i) 3×3 convolution with gated to extract 16 features; (ii) 3×3 convolution with gated to extract 32 features; (iii) 2×4 convolution with gated to extract 40 features; (iv) 3×3 convolution with gated to extract 48 features; (v) 2×4 convolution with gated to extract 56 features; and (vi) 3×3 convolution to extract 64 features. For all traditional convolutional layers, batch renormalization [79] is applied, and, in the last three with gated mechanisms, dropout (rate of 0.2) is applied. As an activator and initializer, we use He uniform and Parametric Rectified

Linear Unit (PReLU) [80], respectively. Finally, the recurrent blocks contain two BGRU (128 hidden units per GRU) with dropout (rate of 0.5) per GRU [52], alternated by a dense layer. Figure 3 shows the overview of proposed optical model architecture.

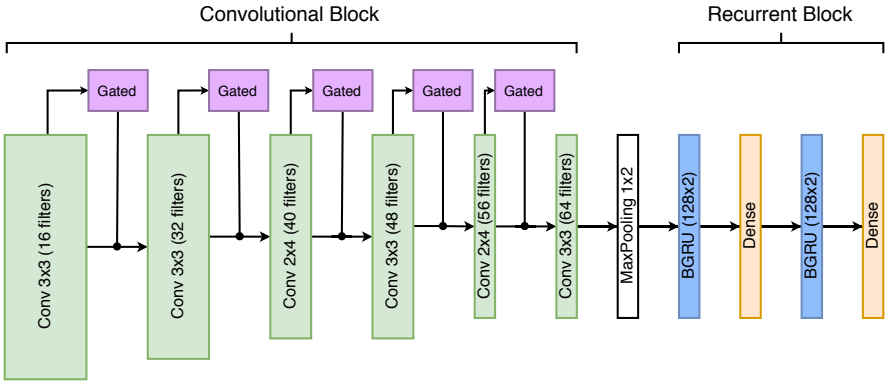

Figure 3. Proposed optical model architecture.

4.2. Spelling Corrector

The spelling corrector, responsible for correcting the output texts of the optical model, is composed by encoder–decoder model architecture, specifically Sequence to sequence (Seq2seq) approach. Then, the proposed Seq2seq architecture uses GRU layers, wherein the Encoder is one bidirectional layer, and the Decoder is another GRU with twice the number of hidden units (to match the Encoder units).

The Attention mechanism is applied with the output of the Encoder and Decoder. In addition, each GRU uses dropout with a rate of 0.2 [81], and a normalization layer is also applied with the output of the Attention mechanism and Decoder [22,82]. Figure 4 shows the overview of proposed Seq2seq architecture.

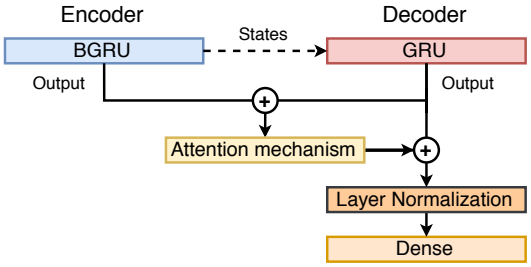

Figure 4. Proposed encoder–decoder architecture using Bidirectional Gated Recurrent Unit (BGRU) as encoder input and Gated Recurrent Unit (GRU) as decoder input.

5. Materials and Methods

To perform the experiment, two independent systems are implemented: (i) Handwritten Text Recognition (HTR), in which it receives an image as input and returns the recognized text; and (ii) the Spelling Corrector, which receives a text as input and returns another text with the corrections. Finally, the correction system acts as post-processing in the HTR system.

The datasets adopted were built with a focus on the text recognition process, but are applied in both systems in this work. For the HTR system, the partitioning methodology (training, validation and testing) follows that defined by each set. It is worth mentioning that the accentuation marks are disregarded for the training, in order to reduce the number of character variations for the optical model and maintain the accentuation corrections in the post-processing stage.

For the Spelling Corrector, the texts (ground truth) of the sets are used to generate multi-grams [18]; this means generating new lines of text from sequential combinations of the original sentences. In this

way, it is possible to considerably increase data and make it compatible with statistical and neural network approaches. In addition, the data are partitioned into training (90%) and validation (10%) for better representation in training, since the test partition will be the same used in the HTR system.

In the spelling correction model training, artificial spelling errors (text noises) are generated through editing operations (transposition, substitution, insertion, and deletion) with error variation between 5% to 10% of the sentence [83]. These text noises are generated randomly at each model iteration (each epoch) for the training partition. On the other hand, the noise is generated only once in the validation partition. Finally, Figure 5 shows the proposed system workflow with the integration in a post-processing step.

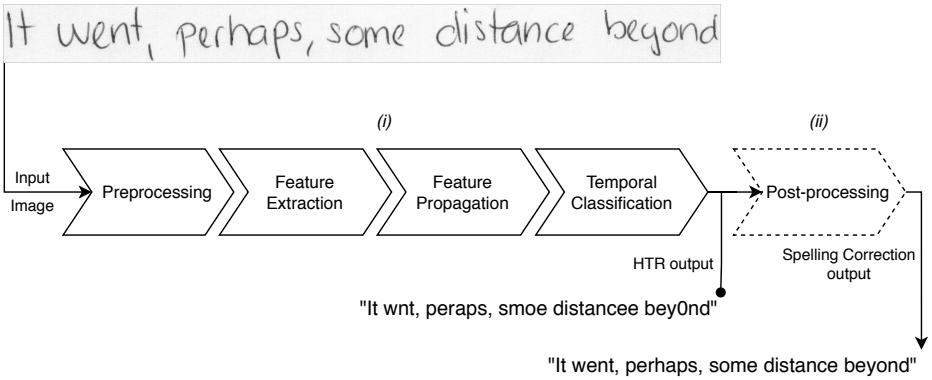

Figure 5. Proposed system workflow.

5.1. Metrics and Evaluation

The most common evaluation metrics in HTR systems are adopted: (i) Character Error Rate (CER); (ii) Word Error Rate (WER); and (iii) Sequence Error Rate (SER) [2,84]. CER is defined as the minimum number of editing operations; at the character level, a word must match its respective ground truth. WER and SER are defined in the same way, but at the word and sentence level.

For the statistical evaluation, we used an analysis equivalent to that presented by [15]. It involves two well-known statistical measures: (i) the confidence interval, to visualize the distribution of the results in the different sentences; and (ii) the p-value, to perform hypothesis tests of superiority or non-inferiority of our proposal in relation to the baseline.

In this context, to calculate the confidence interval, we use the non-parametric bootstrapping test [85], as it is not necessary to assume about the normality of the data distribution. In relation to the p-value calculation, we use the Wilcoxon signed-rank test [86] with a significance of 5%, meaning that the null hypothesis assumes that our approach performs at least 5% worse than the traditional one. Thus, the null and alternative hypotheses in superiority (left) and non-inferiority (right) tests are:

$$\begin{cases} H_0 : e_1 = e_2 \\ H_a : e_1 \neq e_2 \end{cases} \qquad \begin{cases} H_0 : e_1 - e_2 \geq \Delta \\ H_a : e_1 - e_2 < \Delta \end{cases} \tag{7}$$

where e_1 is the error rate (character, word, sequence) achieved by non-traditional spelling correction approaches in the HTR system, e_2 is the achieved by the traditional language model process, and $\Delta = 0.05 \times e_2$. In both scenarios, the spelling correction is analyzed over several recognized texts. Thus, all models of neural networks are executed twenty times [87] and the average among the results of the executions is taken into account.

5.2. Datasets

The experiment is performed on five datasets widely used for unrestricted handwriting recognition: Bentham [35]; IAM [36]; RIMES [37]; Saint Gall [38]; and Washington [39]. In addition,

the performance is also analyzed in a universal dataset composed for all datasets mentioned, called in this work All in One.

For the HTR system, all datasets follow their own partitioning methodology, defined in three subsets (training, validation, and testing). For the Spelling Corrector, the formation of new data (multi-grams) is considered for the training and validation subsets, while the test subset remains the same. In this way, the training subset is used by the learning algorithm to adjust the network parameters (weights). At this stage, training progress is assessed through the validation subset. Finally, the final model is evaluated using the test subset.

5.2.1. Bentham

The Bentham database [35] is a collection of manuscripts written by English philosopher Jeremy Bentham (1748–1832). This set of historical data has about 11,500 lines of text, where the images are in gray scale and have a dark background with noise in the text.

For the HTR system, partitioning consists of 9195 lines of text for training, 1415 for validation, and 860 for testing. For the Spelling Corrector scenario, partitioning is 264,678 for training and 29,408 for validation. In addition, the set has a distribution of 85 unique characters and 9048 unique words. Table 1 summarizes the data distribution in both scenarios.

Table 1. Data distribution from Bentham database.

System	Sentence Length		Average Tokens/Sentence		Partitioning			
	Minimum	Maximum	Chars	Words	Training	Validation	Testing	Total
Handwritten Text Recognition	2	105	48	8	9195	1415	860	11,470
Spelling Correction	2	105	28	5	264,678	29,408	860	294,946

5.2.2. IAM (Institut für Informatik und Angewandte Mathematik)

The Department of Computer Science and Applied Mathematics (Institut für Informatik und Angewandte Mathematik, IAM) database [36] contains 1539 handwritten English scanned text pages in gray scale. This set was built from 657 different writers and has about 9000 lines of text, extracted from these scanned pages. In addition, the images have a clear background and some have a darkening around the words.

This dataset was organized to ensure a line recognition task independent of the writer, which means that each writer's handwriting is found only in a single subset. Then, image partitioning for HTR system is 6161 for training, 900 for validation, and 1861 for testing. For Spelling Corrector, text partitioning is 141,312 for training and 15,701 for validation. Unique characters and words are 78 and 9087, respectively. Table 2 summarizes the data distribution.

Table 2. Data distribution from IAM (Institut für Informatik und Angewandte Mathematik) database.

System	Sentence Length		Average Tokens/Sentence		Partitioning			
	Minimum	Maximum	Chars	Words	Training	Validation	Testing	Total
Handwritten Text Recognition	20	81	44	7	6161	900	1861	8922
Spelling Correction	3	81	23	4	141,312	15,701	1861	158,874

5.2.3. RIMES (Reconnaissance et Indexation de Données Manuscrites et de fac SimilÉS)

The Recognition and Indexing of Handwritten Documents and Faxes (Reconnaissance et Indexation de données Manuscrites et de fac similÉS, RIMES) database [37] is a collection of over 12,000 text lines written in French language by several writers from 5600 handwritten mails. The challenge in this set is to deal with the recognition of several accented characters, since the images have a clear background and more readable writing.

The partitioning for the HTR system consists of 10,193 lines of text for training, 1133 for validation, and 778 for testing. As for the Spelling Corrector, 171,376 is for training and 19,041 for validation. Among the datasets, it has the largest list of unique characters, due to the accented characters, totaling 95 characters and 6358 unique words. Table 3 summarizes the data distribution.

Table 3. Data distribution from RIMES (Reconnaissance et Indexation de données Manuscrites et de fac similÉS) database.

System	Sentence Length		Average Tokens/Sentence		Partitioning			
	Minimum	Maximum	Chars	Words	Training	Validation	Testing	Total
Handwritten Text Recognition	2	110	47	7	10,193	1133	778	12,104
Spelling Correction	2	110	30	5	171,376	19,041	778	191,195

5.2.4. Saint Gall

The Saint Gall database [38] brings historical manuscripts in Latin from the 9th century of a single writer. This set has no punctuation or accentuation marks, having only 48 and 6000 unique characters and words, respectively. In addition, the images are already binarized and normalized.

This dataset is the second, among the others that has a lack of data (1410 in total), and brings the challenge of overfitting in model training. Thus, the partitioning of images for recognition is 468 for training, 235 for validation, and 707 for testing. For spelling correction, it is 42,283 for training and 4698 for validation. Table 4 summarizes the data distribution.

Table 4. Data distribution from Saint Gall database.

System	Sentence Length		Average Tokens/Sentence		Partitioning			
	Minimum	Maximum	Chars	Words	Training	Validation	Testing	Total
Handwritten Text Recognition	8	74	56	8	468	235	707	1410
Spelling Correction	3	74	26	4	42,283	4698	707	47,688

5.2.5. Washington

The Washington database [39] was built from George Washington papers at the Library of Congress in English language from the 18th century. This set of historical manuscripts brings two writers and transcriptions at line-level. Like Saint Gall, it has even less data (total of 656), increasing the overfitting scenario. In addition, your images are also binarized and normalized.

This set also has 68 unique characters and only 1189 unique words. Thus, for the HTR system, the data are divided into 325 for training, 168 for validation, and 163 for testing. As for the Spelling Corrector, there are only 12,933 for training and 1437 validation. Table 5 summarizes the data distribution.

Table 5. Data distribution from Washington database.

System	Sentence Length		Average Tokens/Sentence		Partitioning			
	Minimum	Maximum	Chars	Words	Training	Validation	Testing	Total
Handwritten Text Recognition	4	62	42	7	325	168	163	656
Spelling Correction	3	62	22	4	12,933	1437	163	14,533

5.2.6. All in One

Finally, the approach that combines all the other datasets is presented. This combination respects the partitioning adopted by each dataset; thus, the HTR system has a distribution of 26,342 samples for training, 3851 for validation, and 4369 for testing. As for the Spelling Corrector, there are 632,582 and

70,285 for training and validation, respectively. Furthermore, it has 98 unique characters and 26,690 unique words. Table 6 summarizes the data distribution.

Table 6. Data distribution from All in One database.

System	Sentence Length		Average Tokens/Sentence		Partitioning			
	Minimum	Maximum	Chars	Words	Training	Validation	Testing	Total
Handwritten Text Recognition	2	110	47	8	26,342	3851	4369	34,562
Spelling Correction	2	110	27	4	632,582	70,285	4369	707,236

5.3. Preprocessing

To minimize variations in data (images), we use image preprocessing: (i) illumination compensation [47]; (ii) slant correction (deslant) [48]; and (iii) standardization with color inversion. Thus, it is possible to keep the images with few background and noise among the datasets.

The Penn Treebank Tokenization standard [66] process was applied to text data, both for the HTR and Spelling Correction, which involves spacing between punctuation marks and joining parts of the hyphenated words together. In addition, we also separate the clitics terms, thus, optical model and corrector have their learning processes standardized in the same text pattern.

5.4. Optical Model

To compose the optical model, we adopted three architectures: two of the current state-of-the-art and one proposed for this work (presented in Section 4). In this way, we can analyze different texts' recognition methods as inputs to the Spelling Corrector. The optical model is composed of the convolutional block followed by the recurrent block; in addition, we added the Vanilla Beam Search [88] algorithm as CTC decoding ($beam\_width = 10$), with the exception of the traditional approach, which uses Hidden Markov Model with the language model for decoding.

The first architecture, presented by [40], is a small model that uses its own gated mechanism approach in the convolutional layers. Thus, the convolutional block is composed of: (i) 3×3 convolution to extract 8 features; (ii) 2×4 convolution with gated to extract 16 features; (iii) 3×3 convolution with gated to extract 32 features; (iv) 2×4 convolution with gated to extract 64 features; and (v) 3×3 convolution to extract 128 features. In addition, Hyperbolic Tangent function (tanh) and Glorot uniform [89] is used as activator and initializer, respectively. Likewise, the recurrent block contains two BLSTM (128 hidden units per LSTM) alternated by dense layer. Figure 6 shows the overview of Bluche optical model architecture.

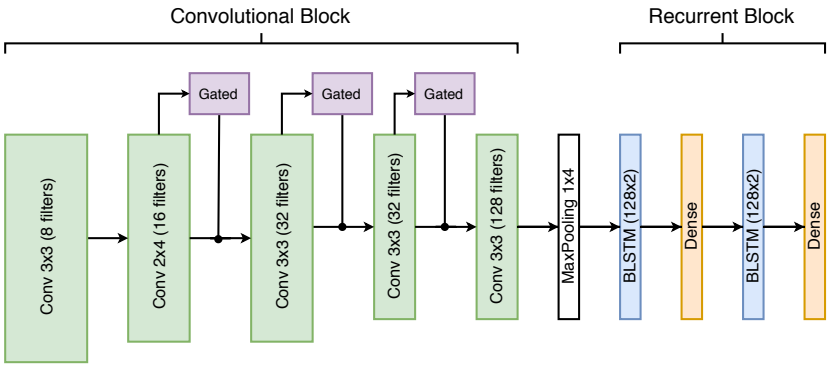

Figure 6. Bluche architecture overview.

The second architecture, presented by [15], is a big model that uses only traditional convolutional layers and several bidirectional recurrent layers; thus, its performance is equivalent to models that use multidimensional recurrent layers [54]. The convolutional block is composed of layers with

3×3 convolution with number of features extraction following the order of $16n$ (16, 32, 48, 64, 80), and MaxPooling to reduce data in the first three layers. In addition, dropout is applied (rate of 0.2) in the last three layers to avoid overfitting [90], and Batch Normalization [91] is used in all convolutional layers to normalize the inputs of nonlinear activation functions. Glorot uniform [89] and Leaky Rectifier Linear Units (LeakyReLU) [92] are used as activator and initializer, respectively. The recurrent block, contains five BLSTM layers (256 hidden units per LSTM) with a dropout (rate of 0.5) per LSTM [52]. Finally, dropout (rate of 0.5) is applied before the last dense layer. Figure 7 shows the overview of Puigcerver optical model architecture.

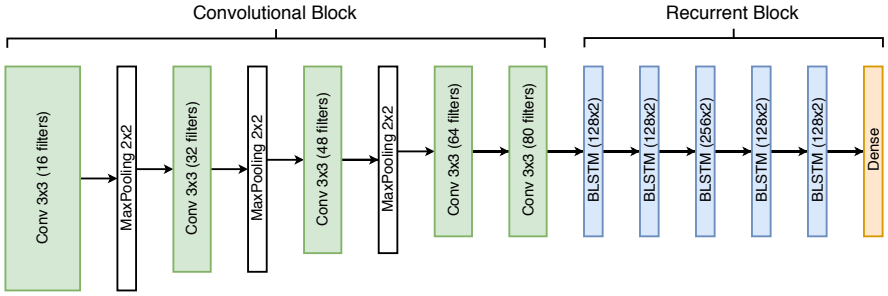

Figure 7. Puigcerver architecture overview.

The training and classification phase is applied in the same way for the three optical models, which we trained each one to minimize the validation loss value of the CTC function. We used RMSprop optimizer [93] with a learning rate of 0.001 and mini-batches of 16 image samples per step. Reduce Learning Rate on Plateau (factor of 0.2) and Early Stopping mechanisms also are applied after 15 and 20 epochs, respectively, without improvement in the validation loss value.

Furthermore, the optical models have input size of $1024 \times 128 \times 1$ (Height $\times$ Width $\times$ Channel) and maximum length text sentence set to 128. Finally, the CTC uses the 95 printable characters from ASCII (letters, digits, punctuation marks) to decode the final text; this means that accented characters are converted to non-accented characters (normalized).

5.5. Spelling Corrector

Spelling correction in the HTR system has two ways of implementation. The first one incorporates a decoding step with the language model to correct possible errors generated by the optical model, whose input is a CTC character probability matrix. The second uses raw text as an input to correct the potential errors. Thus, we use these two implementation scenarios, in which the second approach is used as a proposal. In addition, the traditional method in the HTR system, uses the first implementation approach through HMM combined with the language model [2,15,21].

In the HMM with language model approach, the RNN block serves as an emitter of a probability model, which converts it into scaled pseudo-probability estimates to build the HMMs and perform the decoding. The pipeline consists of three weighted finite-state transducers [57]: (i) the representation of the HMM built through left-to-right topology; (ii) Lexicon, a vocabulary list built from character vocabulary of the optical model; and (iii) Grammar, a N-gram language model built on the corpus (dataset), providing probabilities in relation to most likely sequences of length N. Finally, the decoder receives the input and searches for the best transcription using Vanilla Beam Search algorithm ($beam\_width = 10$) [20].

In second scenario (text as input), we use statistical and neural network approaches. Thus, for the statistical approach, we used three correction techniques—(i) Similarity with N-gram [94]; (ii) Edit distance [95]; and (iii) Edit distance with Symmetric Delete algorithm (SymSpell) [96].

The Similarity method applies the N-gram at the character level to each word and stores the substrings as a representation. To find similar strings, the same N-gram process is performed on the input text and selects the words that share the most substrings.

The Edit distance method, using the [95] algorithm, generates all possible terms using editing operations with a distance N from the query term and searches for it in a dictionary. Thus, for a word of length L, a charset size C and an edit distance $N = 1$, there will be L deletions, $L - 1$ transpositions, CL substitutions, and $C(L + 1)$ insertions, totaling $2L + 2LC + C - 1$ terms at search time. Finally, candidate terms are ordered by the shortest distance.

The Symspell method is an improvement to the Norvig algorithm, in which it generates all possible terms using only delete editing operation with a distance N from the query term and searches for it in a word frequency dictionary. Thus, for a word of length L, a charset size C and an edit distance $N = 1$, there will be only L deletions, totaling L terms at search time. Finally, candidate terms are ordered by the shortest distance with most frequency.

On the other hand, for a neural network approach, we use two models' variations: (i) Sequence to sequence (Seq2seq) with Attention mechanism; and (ii) Transformer. It is worth mentioning that the Seq2seq model was developed and proposed for this work (presented in Section 4), while the Transformer model was used as proposed by [22].

For a better analysis of the proposed architecture, we extended the Seq2seq model in three new variations: (i) using Bahdanau's Attention [30] with 512 hidden units; (ii) using Luong's Attention [31] with 512 hidden units; and (iii) using Luong's Attention [31] with 1024 hidden units. Luong's Attention was chosen in the last variation because it offers fast model training and a fast decoding process, when compared to Bahdanau's Attention and Transformer model, respectively.

Transformer model [22] is also used, which in particular uses only the Attention mechanism instead of the recurrent ones. Thus, unlike the traditional Seq2seq, it uses Embeddings layers with Positional Encoding in the inputs, to create and extract more information from the texts. Then, multiple Multi-Head Attention is performed, followed by normalization layers and Feed Forward processes, which are composed of two dense layers, where the first one is activated by the ReLU function. In addition, the model architecture is the same as the original model base presented by [22], with the exception of the inner-layer dimensionality $d_{ff} = 512$. In this way, it minimizes the difference between the number of trainable parameters between the models. Figure 8 shows the overview of Transformer architecture.

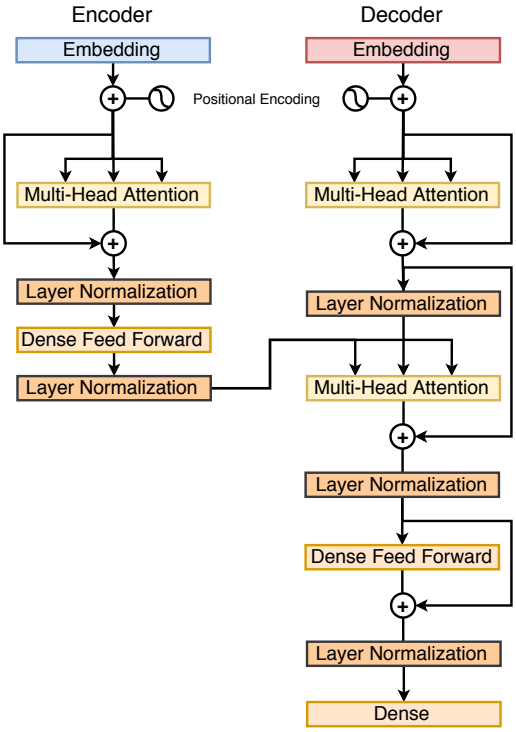

Figure 8. Transformer architecture overview.

The training and inference phase are applied in the same way for all models. Thus, we use the Adam optimizer [97] with learning rate schedule throughout the training [22]. This schedule function corresponds to linearly increasing the learning rate for the first warm-up step of training and then decreasing it proportionally to the inverse square root of the step number. In addition, we define $warmup\_steps = 4000$.

Furthermore, we used batches of 64 text samples per step with a maximum length of 128 per sentence. The Early Stopping mechanism is also used after 20 epochs with no improvement in the validation loss value. Finally, different from optical models, the Spelling Corrector considers printable characters and accented ones as well, totaling a charset size of 150.

5.6. Experimental Design

The experiment is executed over two environments: (i) the Google Colab platform, equipped with Graphics Processing Units (GPU) and responsible for executing the training phase of all models of neural networks (optical and encoder–decoder models); and (ii) a workstation equipped only with Central Processing Unit (CPU) and responsible for performing the classification and inference phase of the models and statistical approaches. The configuration of the two environments is as follows:

(i) Intel Xeon (1) @ 2.20 GHz, 12 GB RAM, NVIDIA Tesla P100 16 GB, Ubuntu 18.04 OS;
(ii) Intel i7-7500U (4) @ 3.500 GHz, 16 GB RAM, Arch Linux OS.

Therefore, to summarize all the possibilities of combinations between the techniques, we organized each component of the experiment in groups: (i) Dataset, responsible for providing the data in image and text format; (ii) Optical Model, responsible for transcribing the content of the image into text; and (iii) Spelling Corrector, responsible for correcting potential errors in the transcribed text. Figure 9 shows the group organization, as well the components of each one.

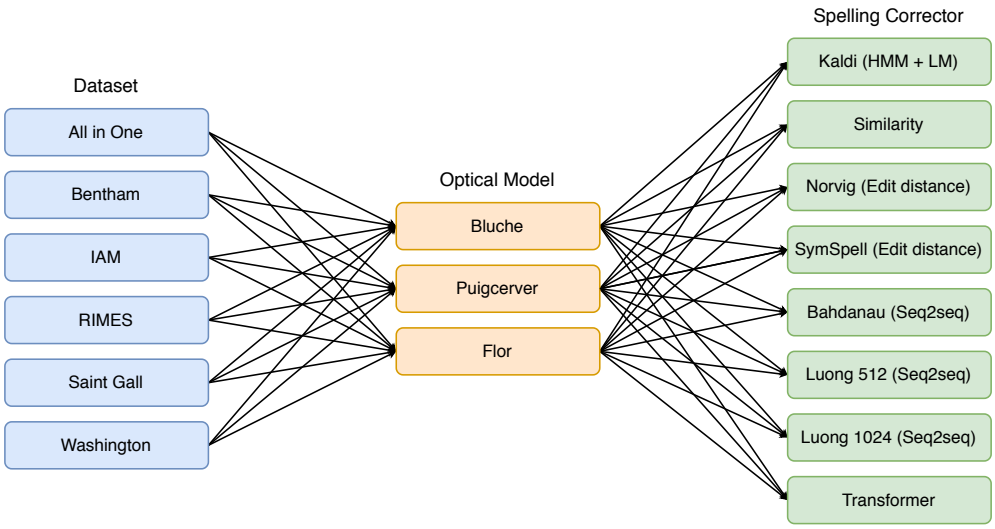

Figure 9. Experiment design overview.

6. Results and Discussion

In this section, we show and analyze all the results following the experimental methodology previously described. At first, the consolidated results with a statistical analysis is presented. Then, in the following subsections, we present the results obtained in each tested dataset, detailing the rates of each optical model and spelling corrector.

6.1. Statistical Analysis

As we used several approaches for spelling correction, including some proposed neural network models, we performed the statistical analysis comparing the method with the best result against the traditional one, performed by Kaldi. In order to define our candidate for analysis, we considered the lowest average error rate in the metrics CER, WER, and SER. Thus, to compose this average, we take into account all the results among all datasets, optical models, and executions.

Furthermore, we take as baseline the output of the three optical models (Bluche, Puigcerver, and Flor) over the three metrics (CER, WER, and SER). This baseline refers to the standalone application of the optical model, that is, no post-processing is applied.

In this context, our proposed model with Luong 1024 had the best results with the average character, word, and sentence error rates of 3.2% (±0.0632), 7.7% (±0.1095) and 35.3% (±0.3898), respectively. On the other hand, Kaldi obtained 6.5% (±0.0632), 19.0% (± 0.1612), and 76.7% (±0.4000). This means that the model with Luong 1024 reached around 65% corrected sentences on average, while Kaldi, 23%. Figure 10 shows in detail the average performance of each technique on the error rate metrics (baseline included).

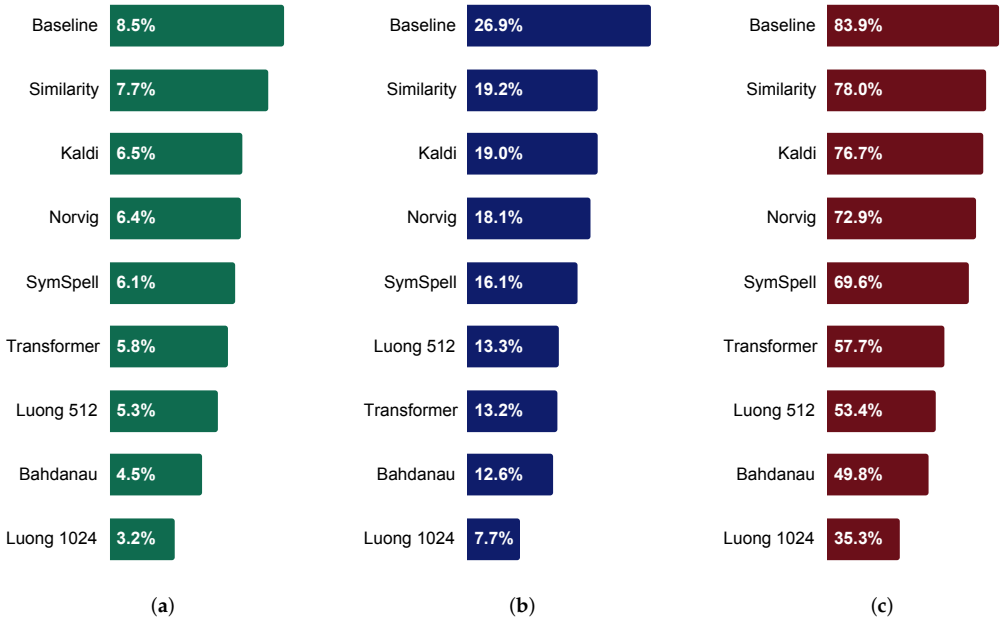

Figure 10. Error rates summary (lower is better), (**a**) Character Error Rate (CER); (**b**) Word Error Rate (WER); (**c**) Sequence Error Rate (SER).

Thus, for confidence interval of 95%, we need to validate if the p-value is below the $\alpha = 0.05$ threshold ($p < 0.05$) to reject the null hypothesis (H_0) or if the p-value is above ($p > 0.05$) to reject the alternative hypothesis (H_1). In addition, an alternative hypothesis to be tested is formulated for each metric. Table 7 details the alternative hypotheses and results obtained, providing statistical evidence.

Table 7. Statistical analysis ($\alpha = 0.05$).

Hypothesis Test	p-Value
Proposed model presents lower CER results than traditional decoding	6×10^{-7} (H_0 rejected)
Proposed model presents lower WER results than traditional decoding	5×10^{-7} (H_0 rejected)
Proposed model presents lower SER results than traditional decoding	5×10^{-7} (H_0 rejected)

Therefore, in all three alternative hypotheses, we compute the p-value lower of $\alpha = 0.05$ threshold and, thus, we reject the null hypothesis. In other words, the rejection indicates a statistically significant

superiority of our proposal when using spelling correction as post-processing of text over the traditional decoding method.

6.2. All in One

The All in One dataset is formed by all the other datasets presented in this work, so that the linguistic models and the neural networks were built and trained from this combination. For a first analysis, the application of these models within the All in One test partition is considered. In a second analysis, the application of All in One's trained models in each specific dataset is considered to analyze whether there is improvement or not compared to standard training. This last analysis is presented in the next sections, within each dataset under analysis.

For the All in One dataset, we obtained up to 58% corrected sentences through the proposed model with Luong 1024, followed by Transformer and Luong 512 with 44% and 38%, respectively. SymSpell ($N = 3$) was better than models with Bahdanau and Luong 512 in correcting characters and words, while, in sentence correction, it matched only with Bahdanau, both with 36%. Then, Norvig ($N = 3$) reached up to 32% corrected sentences, also performed better in some cases at the level of characters and words. Finally, traditional Kaldi ($N = 9$) and Similarity ($N = 2$) presented a sentence correction rate of up to 24%. Figure 11 shows the performance of each technique on the error rate metrics, also considering the standalone application of the three optical models as worst case scenario (baseline).

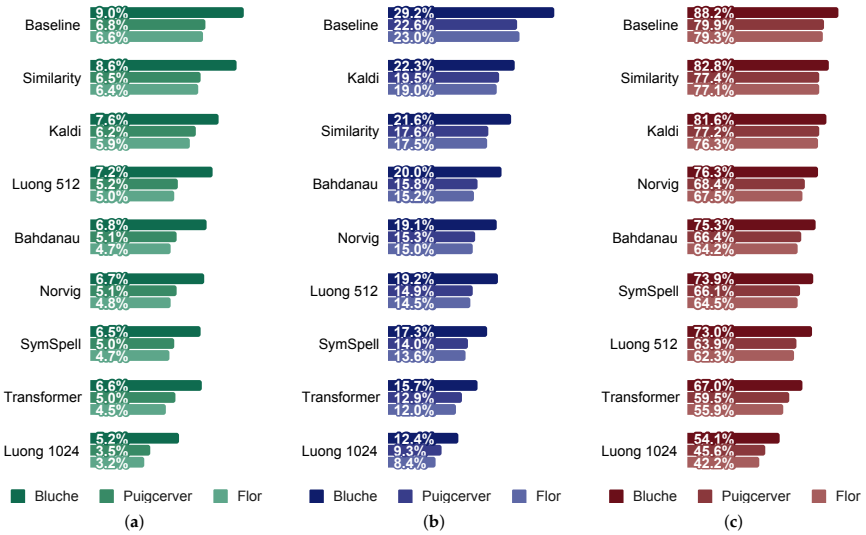

Figure 11. Error rates in the All in One dataset (lower is better), (**a**) Character Error Rate (CER); (**b**) Word Error Rate (WER); (**c**) Sequence Error Rate (SER).

In a way, the accentuation marks, ignored by the optical model, are corrected by spelling correction. However, the All in One dataset presents three languages (English, French, and Latin), which increases the complexity of learning and correction. With an outstanding benefit, the ability to learn sequences of terms (punctuation included) is a the differential of neural networks for statistical dictionaries.

6.3. Bentham

In the Bentham dataset, several punctuation marks in sequence are present in the sentences, so that it makes it difficult for techniques that use sequence-based knowledge. In addition, punctuation marks represent about 14% of the error rates obtained; this means that, if the punctuation marks are disregarded, the results obtained are improved. However, we maintain punctuation marks for analysis.

Thus, our proposed model achieved about 75% correct sentences with Luong 1024, followed by the Transformer and Bahdanau model, with 68% and 61%, respectively. It is important to mention that, regarding the metrics at character and word level, SymSpell ($N = 6$) performed better, compared to the

proposed model with Bahdanau and Luong 512, and Norvig ($N = 3$). Finally, Kaldi ($N = 9$) reached about 48% correction of complete sentences, followed again by Similarity ($N = 2$), with 45%. Figure 12 shows each technique on the error rate metrics in detail.

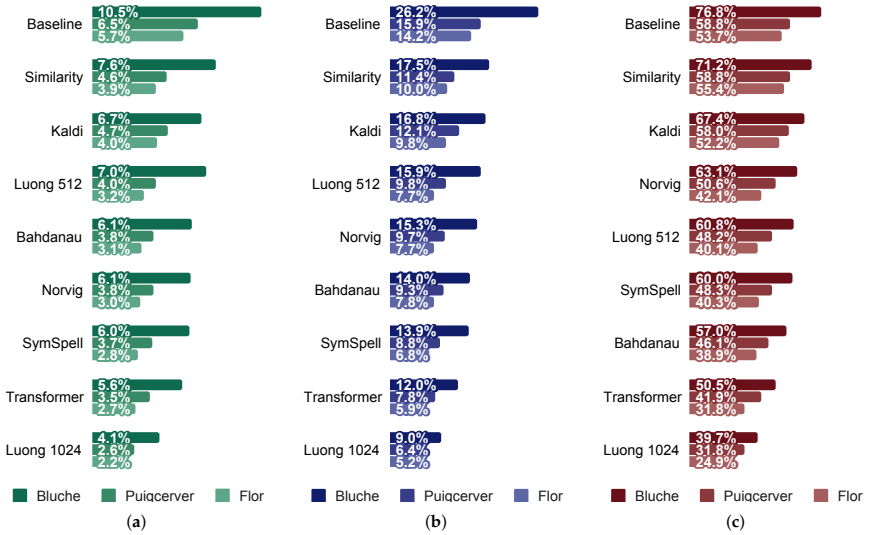

Figure 12. Error rates in the Bentham dataset (lower is better), (**a**) Character Error Rate (CER); (**b**) Word Error Rate (WER); (**c**) Sequence Error Rate (SER).

The main challenge in this dataset was dealing with punctuation marks and small tokens. On the other hand, errors caused by the optical model are maintained or just exchanged for another one. In addition, the Kaldi approach, in some cases, ignores some punctuation marks in more complex corrections. Figure 13 exemplifies a correction between the techniques through optical model output.

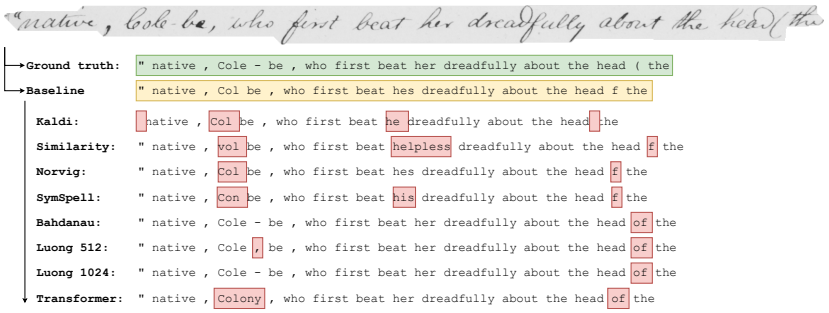

Figure 13. Correction sample in the Bentham dataset.

Using the trained model from the All in One dataset, in addition to our own dataset corpus for statistical methods, the scenario changes slightly. In general, the optical models have a decrease in recognition, also causing a decrease in corrections. Thus, our model proposed with Luong 1024 reached 62% corrected sentences, followed by the Transformer with 60%. SymSpell ($N = 6$) and Norvig ($N = 3$) now achieved a correction of 54% and 52%, respectively, while the models proposed with Luong 512 and Bahdanau reached 53% and 51%. Finally, Kaldi ($N = 9$) reached 43% and Similarity ($N = 2$) 40%.

6.4. IAM

The IAM dataset presents a scenario similar to Bentham, in which there is the presence of sequential punctuation marks within sentences and has about 9000 tokens distributed by the set. However, IAM has a smaller number of tokens in each sentence, causing smaller sentences. In addition, it has about 7% in the error rates obtained referring to punctuation marks.

In this way, our proposed model reached about 79% of correct sentences with Luong 1024, followed by the Transformer model with 62%. Unlike the Bentham dataset, the proposed model using Luong 512 had performance equivalent to Bahdanau's one, reaching 58% in both. In the statistical methods, SymSpell ($N = 3$) corrected about 47% of the sentences, while Norvig ($N = 3$) and Kaldi ($N = 8$) reached 41% and 38%, respectively. Finally, Similarity ($N = 2$) with only 29% corrected sentences. Figure 14 shows each technique performed on the error rate metrics.

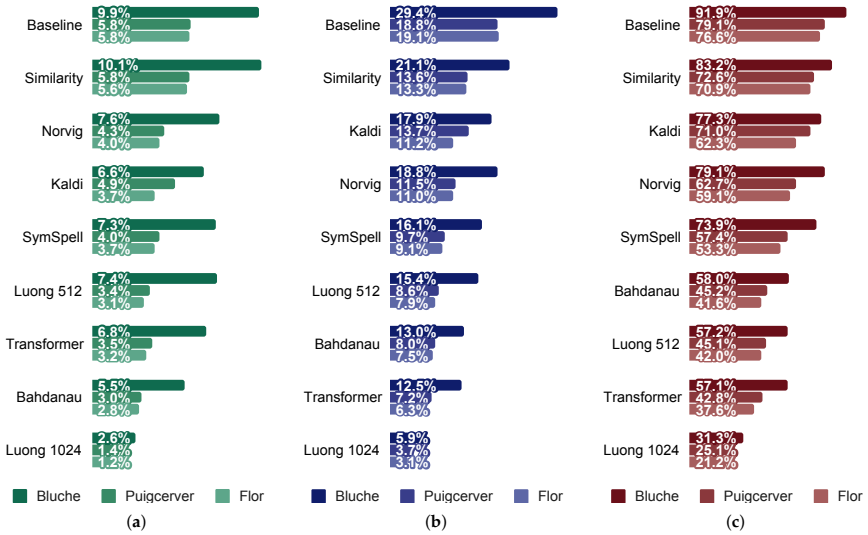

Figure 14. Error rates in the IAM dataset (lower is better), (**a**) Character Error Rate (CER); (**b**) Word Error Rate (WER); (**c**) Sequence Error Rate (SER).

As a challenge, there is the low occurrence of some specific tokens, in a way that makes it difficult to use frequency dictionaries or knowledge construction from sequences, for example. Thus, it is more likely to exchange a wrong token for another, but with greater probability of sequence. This problem was overcome, in some cases, by the neural networks when considering a larger context of the sentences, which in turn have a lower average in length. Figure 15 exemplifies a sentence correction between the techniques.

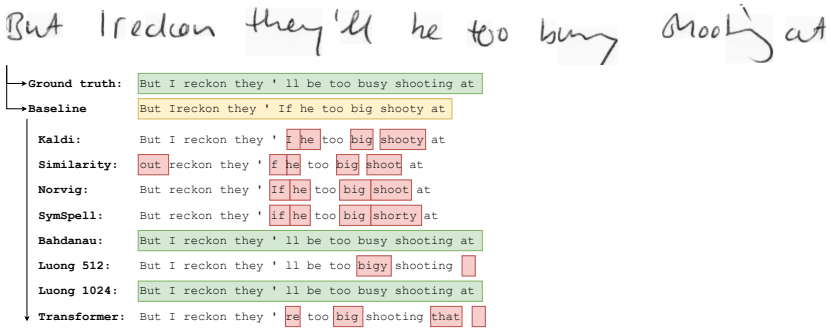

Figure 15. Correction sample in IAM dataset.

Trained on the All in One dataset, the optical models Bluche and Flor achieved a small improvement in recognition, while the Puigcerver model worsened by about 12%. Even so, our model with Luong 1024 achieved 53% of corrected sentences, while Transformer 46%. Ranked below, these are SymSpell ($N = 3$) with 43% and Norvig ($N = 3$) with 39%. Kaldi ($N = 8$), now reaching 31% of corrected sentences, surpassed on average the models proposed with Luong 512 and Bahdanau, reaching 33% and 31%, respectively. Finally, Similarity ($N = 2$) is very close to the baseline with only 23%, in addition, character correction was worsened.

6.5. RIMES

The RIMES dataset has a particular scenario with the French language, characterized by accentuation marks. Thus, punctuation marks can only reach up to 3%, while accentuation marks can reach up to 48% of the obtained error rates, mainly in statistical methods. It is important to mention that accentuation markings are normalized in the optical model, which means that the output matrix (used by Kaldi) does not have the accented character mapping.

Thus, our proposed model reached about 90% of the correct sentences with Luong 1024, followed by the model with Bahdanau with 80%. The Transformer model had the worst result among neural networks, about 77%. The scenario changes dramatically in statistical methods, with SymSpell ($N = 2$) correcting about 30% of sentences and Norvig ($N = 3$) with 25%. Finally, again, Kaldi ($N = 12$) and Similarity ($N = 2$), with 22% and 20% of correct sentences, respectively. Figure 16 shows each technique performed on the error rate metrics.

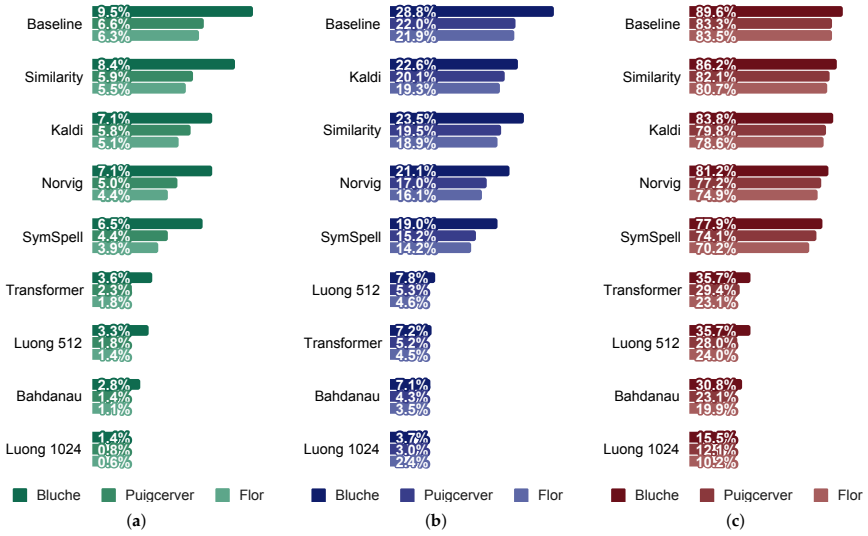

Figure 16. Error rates in the RIMES dataset (lower is better), (**a**) Character Error Rate (CER); (**b**) Word Error Rate (WER); (**c**) Sequence Error Rate (SER).

As a challenge, the accentuation marks make correction difficult, since it is another type of error in the sentence. For example, Kaldi has a constraint due to the optical model integration, which does not have accented characters in its charset. The other techniques can deal with the correction since the corpus for knowledge build (dictionaries and neural networks) contains the accented terms. Figure 17 exemplifies a sentence correction between the techniques.

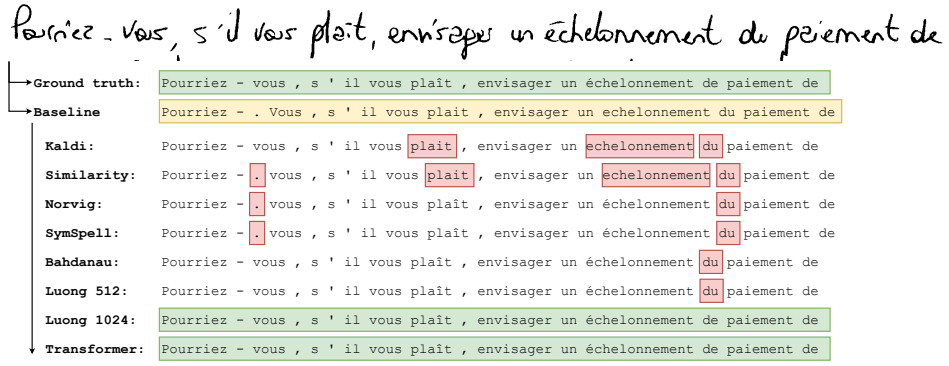

Figure 17. Correction sample in the RIMES dataset.

Trained on the All in One dataset, the Bluche optical model had a small improvement in recognition, while Puigcerver and Flor had a small worsening. This also occurred in the spelling correction process, for which, in general, the results worsened. In this context, the proposed model reached about 72% corrected sentences with Luong 1024, followed by 65% with Transformer, 60% with Bahdanau, and 58% with Luong 512. SymSpell (N = 2) now reaches 26% correction, while Norvig (N = 3), Similarity (N = 2) and Kaldi (N = 12) reach 24%, 26% and 21%, respectively.

6.6. Saint Gall

The Saint Gall dataset has no punctuation or accentuation marks. In addition, the set is composed only of the Latin language, which has few characters, and the volume of data is much smaller than the previous one. In this context, our proposed model with Luong 1024 performed up to 94% of corrected sentences, followed by the models with Bahdanau, Luong 512, with 86% and 76%, respectively. The Transformer model had the worst result among neural networks, reaching only 18%. Similarity (N = 2) and SymSpell (N = 2) both reached 20% correction, while Norvig (N = 3) with 18%. It is important to note that Kaldi (N = 11), even the last with about 8% sentence correction, ranks first among statistical methods when evaluating the character error rate. Figure 18 details the results of each technique.

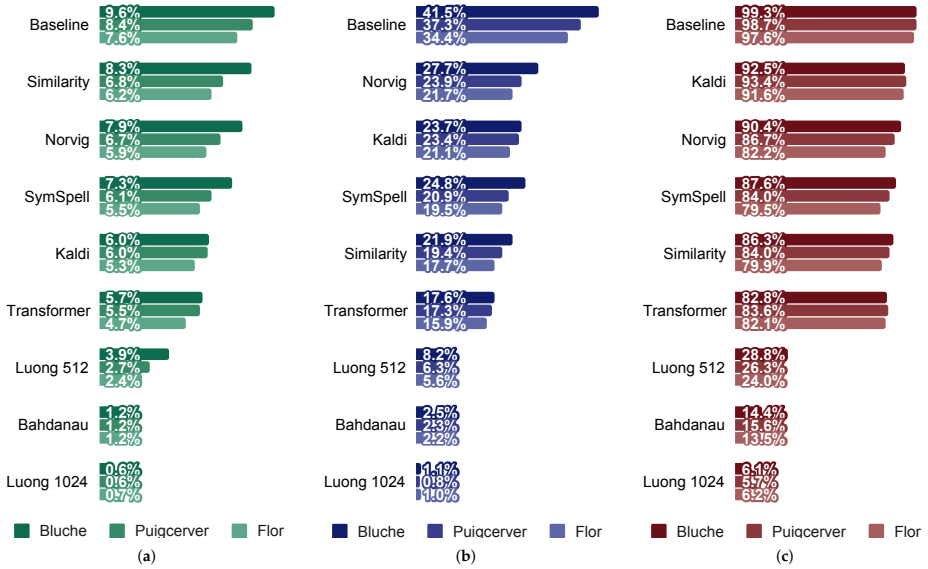

Figure 18. Error rates in the Saint Gall dataset (lower is better), (**a**) Character Error Rate (CER); (**b**) Word Error Rate (WER); (**c**) Sequence Error Rate (SER).

Among the evaluated datasets, Saint Gall has the longest tokens. In a way, this scenario of long tokens and no punctuation or accentuation marks facilitate the model's learning. On the other hand, statistical methods presented difficulties in discerning wrong tokens (formed by small ones), causing a false correction. Figure 19 exemplifies a sentence correction between the techniques.

When trained on the All in One, the Bluche optical model had a small improvement in recognition, while Puigcerver and Flor worsened a little, that is, spelling correction results worsened, as the RIMES dataset. Thus, the model proposed with Luong 1024 reached 50% corrected sentences, followed by SymSpell (N = 2) and Similarity (N = 2) with 15% and 16%, respectively. The model with Luong 512 and Norvig (N = 3) had equivalent results, both with 12% correction. Finally, the model with Bahdanau reaching 7%, Kaldi with 3% and the Transformer with only 2% of corrected sentences.

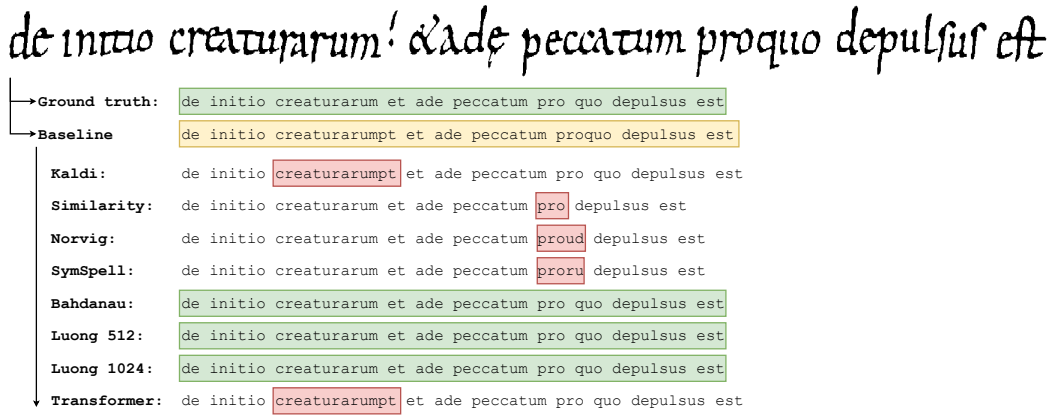

Figure 19. Correction sample in the Saint Gall dataset.

6.7. Washington

The Washington dataset has the lowest data volume among previous datasets. In this scenario, the training of the Puigcerver optical model suffers from overfitting, causing the Early Stopping in the first epochs. Thus, we have a scenario in which recognition has a high error rate.

The proposed model with Bahdanau had the best rate of sentence correction, reaching up to 90%. Then, the models proposed with Luong 1024 and 512, with up to 88% and 82%. The Transformer model ranked last among neural networks, with a correction of 67%. In addition, Kaldi (N = 10) obtained a correction rate of up to 56%, followed by SymSpell (N = 6) with 44%. Finally, Similarity (N = 2) and Norvig (N = 3), with 39% and 37%, respectively. It is important to highlight that Kaldi also maintained good results between the character and word levels, performing better than the Transformer model in these last two scenarios. Figure 20 shows each technique performed on the metrics.

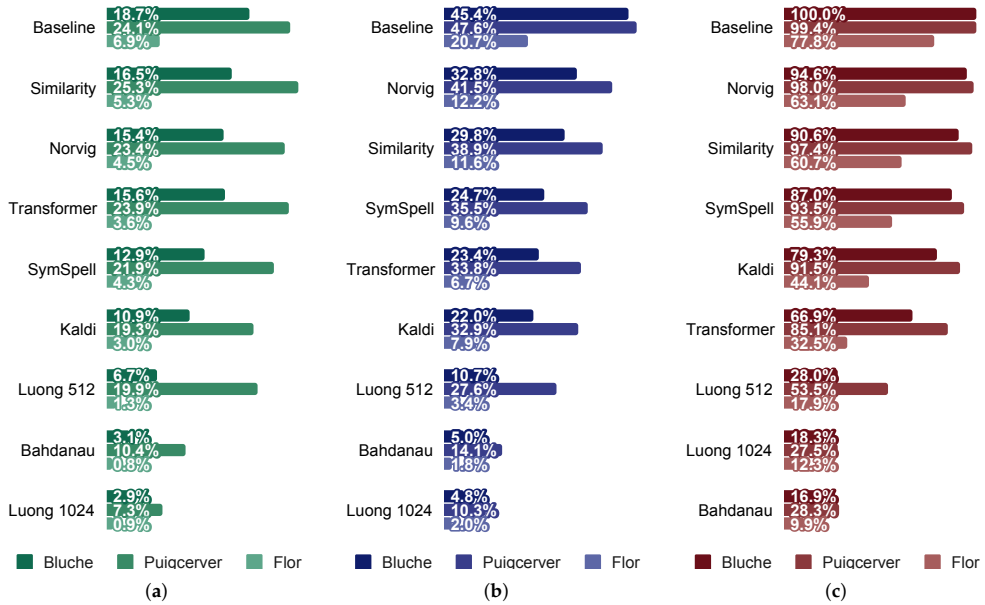

Figure 20. Error rates in the Washington dataset (lower is better), (**a**) Character Error Rate (CER); (**b**) Word Error Rate (WER); (**c**) Sequence Error Rate (SER).

As mentioned, the challenge in this dataset was the low volume of data. Thus, statistical methods have a little information about the distribution of tokens, especially frequency dictionaries for unusual terms. On the other hand, neural networks are able to deal better, since they consider larger contexts. Figure 21 exemplifies a sentence correction between the techniques.

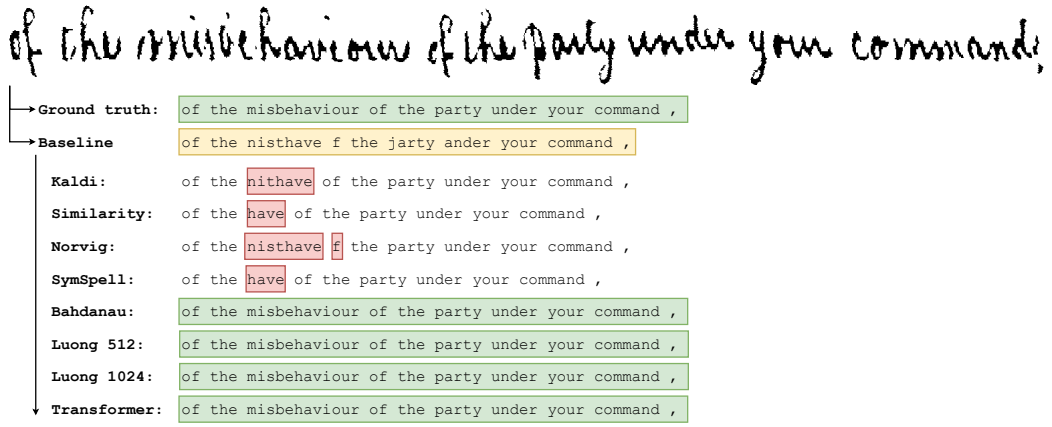

Figure 21. Correction sample in the Washington dataset.

When trained on the All in One, our model with Luong 1024 obtained up to 38% corrected sentences, followed by Transformer with 31%. SymSpell (N = 6) and Kaldi (N = 10) had equivalent correction averages, reaching up to 30% and 29%, respectively. The proposed model with Luong 512 corrected up to 28% of the sentences, while Norvig (N = 3) and Bahdanau, both with 25%. Finally, Similarity (N = 2) reached 21% corrected sentences.

7. Discussion

The significant correction rate of our proposed models compared to the statistical approach, about a 54% improvement, can be explained not only by the use of encoder–decoder architecture in conjunction with the Attention mechanism, but mainly for the standardization of texts between Handwritten Text Recognition and Spelling Correction, and also training methodology, through the random insertion of errors. Furthermore, the more hidden units are defined in the encoder–decoder layers, the more the network learns about the tokens and sequences of a dataset.

It is important to mention that the proposed models, and also the Transformer, have a number of parameters that are considerably lower than the Natural Language Processing state-of-the-art (about 10 and 100 times less), such as Machine Translation and Grammatical Error Correction. This occurred because the main goal of this work is spelling correction small datasets, aimed at HTR systems, while state-of-the-art models have as a main goal, more complex works. In fact, the datasets used in both scenarios exceed millions or even billions of sentences for training, which makes it possible for the model to fully learn through the balanced distribution of tokens between partitions.

In addition for HTR systems, other aspects must also be considered in addition to the correction rates, such as the trade-off between data volume, hidden units, computational cost, training, and inference time. The volume of data directly influences the increase in training time, while the hidden units increase the computational cost. These two factors can considerably increase the training and inference time, and depending on the Attention mechanism used, they can impact the construction and execution of a spelling correction system, for example. Table 8 details the average values of the mentioned aspects, obtained from all correction techniques ("-" corresponds not computed).

In a way, the Luong Attention mechanism (multiplicative) is more optimized compared to Bahdanau (concatenative), even with a slightly lower result when using the same value of hidden units. In addition, both have a much more optimized decoding process compared to the Transformer model. As expected, neural networks have a very long inference time compared to statistical models, mainly the traditional method performed by Kaldi toolkit.

Table 8. Spelling corrector average execution summary.

Technique	# of Params (Million)	GPU Memory (GB)	Convergence Epoch	Training Time [1] (Hours)	Inference Time [2] (Seconds/Sentence)
Kaldi	0.5	-	-	-	0.003
Similarity	-	-	-	-	0.032
Norvig	-	-	-	-	0.321
SymSpell	-	-	-	-	0.004
Bahdanau	5.9	9.5	336	83	0.091
Luong 512	5.9	2.5	348	53	0.090
Luong 1024	21.4	4.5	250	45	0.111
Transformer	8.0	4.4	338	83	7.077

[1] GPU mode; [2] CPU mode.

Furthermore, the All in One dataset worsens the results when applied individually to each set. For the HTR system, the optical model deals with five types of image patterns (Bentham, IAM, RIMES, Saint Gall, Washington), which makes it difficult to learn. The only exception scenario was the Puigcever optical model applied to the Washington dataset, in which the recognition results were significantly improved. The same difficulty in learning the model also happened with the Spelling Corrector, which now has five text patterns and three languages. The exceptions were dictionary-based techniques, especially through frequencies (SymSpell), in which the volume of data helped in the distribution of tokens. Finally, it is also important to consider the pattern imbalance in the All in One dataset as a way of tending the knowledge learned to a certain set, such as the RIMES over Washington.

Finally, there is a notable improvement in the application of encoder–decoder models for applications with sentences within the field of Natural Language Processing. Our proposed models are very competitive in this aspect and bring a new path into HTR systems. In addition, SymSpell also proved to be a viable alternative as it offers competitive results, speed of execution, and flexibility in frequency dictionaries, in which larger dictionaries can be easily applied. The combinations between the techniques is also interesting, mainly among those that use sequence correction at the character level (*N*–gram or neural networks), followed by the correction of tokens at the level of words in order to refine the final text.

8. Conclusions

In this paper, we presented the offline Handwritten Text Recognition (HTR) system and its linguistic restriction when using the decoding method through the Hidden Markov Model (HMM) with a language model. As a proposal, we built a Spelling Correction system that acts as post-processing to the HTR system. It enabled the experimentation of eight independent spelling correction techniques within the field of Natural Language Processing (NLP), from statistical to neural network, including the traditional approach of HMM with language model.

A statistical analysis was performed to validate the significance of the results. We considered all the executions of the combinations and group them by spelling correction technique; thus, our proposed encoder–decoder model, using 1024 hidden units and Luong Attention, had the best averages of error rates, about 3.2% CER, 7.7% WER, and 35.3% SER, while traditional HMM with language model reached about 6.5% CER, 19.0% WER, and 76.7% SER. This means that there were significant results in the total of completely correct sentences, around 54% improvement, and a decrease of 3.3, 11.3, and 41.4 percentage points in the metrics, respectively.

In the future, we want to expand the linguistic model independently of the optical model, optimizing its precision to apply the correction in very noisy sentences or even in sentences that are missing words, due to damaged images.

Author Contributions: Conceptualization, A.F.d.S.N., B.L.D.B., and A.H.T.; methodology, A.F.d.S.N. and B.L.D.B.; software, A.F.d.S.N. and A.H.T.; validation, A.F.d.S.N., B.L.D.B., and A.H.T.; formal analysis, A.H.T.; investigation, B.L.D.B.; resources, B.L.D.B.; writing—original draft preparation, A.F.d.S.N.; writing—review and editing, B.L.D.B. and A.H.T.; visualization, A.F.d.S.N.; supervision, B.L.D.B. and A.H.T.; project administration, B.L.D.B.; funding acquisition, B.L.D.B. All authors have read and agreed to the published version of the manuscript.

Funding: This research was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior—Brasil (CAPES)—Finance Code 001, and CNPq Grant No. 315251/2018-2.

Acknowledgments: This study was financed in part by the founding public agencies: Coordenação de Aperfeiçoamento de Pessoal de Nível Superior—Brasil (CAPES)—Finance Code 001, and CNPq.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Singh, S.; Kariveda, T.; Das Gupta, J.; Bhattacharya, K. Handwritten words recognition for legal amounts of bank cheques in English script. In Proceedings of the ICAPR 2015—2015 8th International Conference on Advances in Pattern Recognition, Kolkata, India, 4–7 January 2015; pp. 1–5. [CrossRef]
2. Sánchez, J.A.; Romero, V.; Toselli, A.H.; Villegas, M.; Vidal, E. A Set of Benchmarks for Handwritten Text Recognition on Historical Documents. *Pattern Recognit.* **2019**, *94*, 122–134. [CrossRef]
3. Kamalanaban, E.; Gopinath, M.; Premkumar, S. Medicine Box: Doctor's Prescription Recognition Using Deep Machine Learning. *Int. J. Eng. Technol.* **2018**, *7*, 114–117. [CrossRef]
4. Burie, J.C.; Chazalon, J.; Coustaty, M.; Eskenazi, S.; Luqman, M.M.; Mehri, M.; Nayef, N.; Ogier, J.M.; Prum, S.; Rusiñol, M. ICDAR2015 competition on smartphone document capture and OCR (SmartDoc). In Proceedings of the 2015 13th International Conference on Document Analysis and Recognition (ICDAR), Nancy, France, 23–26 August 2015; pp. 1161–1165.
5. Palehai, D.; Fanany, M.I. Handwriting Recognition on Form Document Using Convolutional Neural Network and Support Vector Machines (CNN-SVM). In Proceedings of the 5th International Conference on Information and Communication Technology (ICoIC7), Melaka, Malaysia, 17–19 May 2017. [CrossRef]
6. Bezerra, B.; Zanchettin, C.; Toselli, A.; Pirlo, G. *Handwriting: Recognition, Development and Analysis*; Computer Science, Technology and Applications; Nova Science Publishers: New York, NY, USA, 2017; pp. 1–394.
7. George, C.; Podhumani, S.; Monish, R.K.; Pavithra, T. Survey on Handwritten Character Recognition using Artificial Neural Network. *IJSTE Int. J. Sci. Technol. Eng.* **2016**, *2*, 287–290.
8. Sonkusare, M.; Sahu, N. A Survey on Handwritten Character Recognition (HCR) Techniques for English Alphabets. *Adv. Vis. Comput. Int. J.* **2016**, *3*, 1–12. [CrossRef]
9. Bunke, H.; Roth, M.; Schukat-Talamazzini, E.G. Off-line Cursive Handwriting Recognition using Hidden Markov Models. *Pattern Recognit.* **1995**, *28*, 1399–1413. [CrossRef]
10. Saritha, B.S.; Hemanth, S. An Efficient Hidden Markov Model for Offline Handwritten Numeral Recognition. *arXiv* **2010**, arXiv:1001.5334.
11. Toselli, A.H.; Vidal, E. Handwritten Text Recognition Results on the Bentham Collection with Improved Classical N-Gram-HMM methods. In Proceedings of the 3rd International Workshop on Historical Document Imaging and Processing (HIP@ICDAR), Nancy, France, 22 August 2015; pp. 15–22. [CrossRef]
12. Ahmed, S.B.; Naz, S.; Swati, S.; Razzak, M.I. Handwritten Urdu character recognition using one-dimensional BLSTM classifier. *Neural Comput. Appl.* **2019**, *31*, 1143–1151. [CrossRef]
13. Scheidl, H.; Fiel, S.; Sablatnig, R. Word Beam Search: A Connectionist Temporal Classification Decoding Algorithm. In Proceedings of the 16th International Conference on Frontiers in Handwriting Recognition (ICFHR), Niagara Falls, NY, USA, 5–8 August 2018; pp. 253–258. [CrossRef]
14. Ingle, R.R.; Fujii, Y.; Deselaers, T.; Baccash, J.; Popat, A.C. A Scalable Handwritten Text Recognition System. In Proceedings of the 2019 International Conference on Document Analysis and Recognition (ICDAR), Sydney, Australia, 20–25 September 2019; pp. 17–24.
15. Puigcerver, J. Are Multidimensional Recurrent Layers Really Necessary for Handwritten Text Recognition? In Proceedings of the 14th IAPR International Conference on Document Analysis and Recognition (ICDAR), Kyoto, Japan, 9–15 November 2017; pp. 67–72. [CrossRef]
16. Bluche, T. Joint Line Segmentation and Transcription for End-to-End Handwritten Paragraph Recognition. *Adv. Neural Inf. Process. Syst.* **2016**, 838–846. [CrossRef]

17. Pirinen, T.A.; Lindén, K. State-of-the-Art in Weighted Finite-State Spell-Checking. In Proceedings of the 15th International Conference on Computational Linguistics and Intelligent Text Processing, Kathmandu, Nepal, 6–12 April 2014; Springer: Berlin/Heidelberg, Germany, 2014; pp. 519–532. [CrossRef]
18. Swaileh, W. Language Modelling for Handwriting Recognition. Ph.D. Thesis, Normandie Université, Normandy, France, 2017.
19. Stolcke, A. SRILM—An extensible language modeling toolkit. In Proceedings of the 7th International Conference on Spoken Language Processing (ICSLP 2002), Denver, CO, USA, 16–20 September 2002; pp. 901–904.
20. Povey, D.; Ghoshal, A.; Boulianne, G.; Burget, L.; Glembek, O.; Goel, N.; Hannemann, M.; Motlíček, P.; Qian, Y.; Schwarz, P.; et al. The Kaldi speech recognition toolkit. In Proceedings of the IEEE 2011 Workshop on Automatic Speech Recognition and Understanding, Waikoloa, HI, USA, 11–15 December 2011.
21. Bluche, T. Deep Neural Networks for Large Vocabulary Handwritten Text Recognition. Ph.D. Thesis, Université Paris-Sud, Orsay, France, 2015.
22. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, L.; Polosukhin, I. Attention is All you Need. In Proceedings of the Conference on Neural Information Processing Systems (NIPS), Long Beach, CA, USA, 4–9 December 2017.
23. Liu, X.; Wang, W.; Liang, W.; Li, Y. Speed Up the Training of Neural Machine Translation. *Neural Process. Lett.* **2019**. [CrossRef]
24. Susanto, R.H.; Phandi, P.; Ng, H.T. System Combination for Grammatical Error Correction. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Doha, Qatar, 25–29 October 2014; pp. 951–962. [CrossRef]
25. Chollampatt, S.; Ng, H.T. A Multilayer Convolutional Encoder–Decoder Neural Network for Grammatical Error Correction. In Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, Frontiers, New Orleans, LA, USA, 2–7 February 2018.
26. Sutskever, I.; Vinyals, O.; Le, Q. Sequence to Sequence Learning with Neural Networks. *Adv. Neural Inf. Process. Syst.* **2014**, *4*. [CrossRef]
27. Schmaltz, A.; Kim, Y.; Rush, A.; Shieber, S. Sentence-Level Grammatical Error Identification as Sequence-to-Sequence Correction. In Proceedings of the 11th Workshop on Innovative Use of NLP for Building Educational Applications (BEA@NAACL-HLT 2016), The Association for Computer Linguistics, San Diego, CA, USA, 12–17 June 2016; pp. 242–251. [CrossRef]
28. Junczys-Dowmunt, M.; Grundkiewicz, R.; Guha, S.; Heafield, K. Approaching Neural Grammatical Error Correction as a Low-Resource Machine Translation Task. In Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL), New Orleans, LA, USA, 1–6 June 2018; pp. 595–606. [CrossRef]
29. Cho, K.; van Merriënboer, B.; Bahdanau, D.; Bougares Fethi Schwenk, H.; Bengio, Y. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Doha, Qatar, 25–29 October 2014; Association for Computational Linguistics: Doha, Qatar, 2014; pp. 1724–1734. [CrossRef]
30. Bahdanau, D.; Cho, K.H.; Bengio, Y. Neural machine translation by jointly learning to align and translate. In Proceedings of the 3rd International Conference on Learning Representations (ICLR), San Diego, CA, USA, 7–9 May 2015.
31. Luong, T.; Pham, H.; Manning, C.D. Effective Approaches to Attention-based Neural Machine Translation. In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, Lisbon, Portugal, 17–21 September 2015; Association for Computational Linguistics: Lisbon, Portugal, 2015; pp. 1412–1421. [CrossRef]
32. Dai, Z.; Yang, Z.; Yang, Y.; Carbonell, J.; Le, Q.; Salakhutdinov, R. Transformer-XL: Attentive Language Models beyond a Fixed-Length Context. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, Florence, Italy, 28 July–2 August 2019; Association for Computational Linguistics: Florence, Italy, 2019; pp. 2978–2988. [CrossRef]
33. Thus, D.R.; Liang, C.; Le, Q.V. The Evolved Transformer. In Proceedings of the 36th International Conference on Machine Learning, Long Beach, CA, USA, 10–15 June 2019.
34. Wu, C.H.; Liu, C.H.; Harris, M.; Yu, L.C. Sentence Correction Incorporating Relative Position and Parse Template Language Models. *IEEE Trans. Audio, Speech Lang. Process.* **2010**, *18*, 1170–1181. [CrossRef]

35. Gatos, B.; Louloudis, G.; Causer, T.; Grint, K.; Romero, V.; Sánchez, J.A.; Toselli, A.H.; Vidal, E. Ground-truth production in the tranScriptorium project. In Proceedings of the 11th IAPR International Workshop on Document Analysis Systems (DAS), Tours, France, 7–10 April 2014; pp. 237–241.
36. Marti, U.V.; Bunke, H. The IAM-database: An English sentence database for offline handwriting recognition. *Int. J. Doc. Anal. Recognit.* **2002**, *5*, 39–46. [CrossRef]
37. Grosicki, E.; Carre, M.; Brodin, J.M.; Geoffrois, E. RIMES evaluation campaign for handwritten mail processing. In Proceedings of the ICFHR 2008: 11th International Conference on Frontiers in Handwriting Recognition, Montréal, QC, Canada, 19–21 August 2008; Concordia University: Montreal, QC, Canada, 2008; pp. 1–6.
38. Fischer, A.; Indermühle, E.; Bunke, H.; Viehhauser, G.; Stolz, M. Ground truth creation for handwriting recognition in historical documents. *ACM Int. Conf. Proc. Ser.* **2010**, 3–10. [CrossRef]
39. Fischer, A.; Frinken, V.; Fornés, A.; Bunke, H. Transcription Alignment of Latin Manuscripts Using Hidden Markov Models. In Proceedings of the 2011 Workshop on Historical Document Imaging and Processing, Beijing, China, 16–17 September 2011; Association for Computing Machinery: New York, NY, USA, 2011; HIP'11; pp. 29–36. [CrossRef]
40. Bluche, T.; Messina, R. Gated Convolutional Recurrent Neural Networks for Multilingual Handwriting Recognition. In Proceedings of the 14th IAPR International Conference on Document Analysis and Recognition (ICDAR), Kyoto, Japan, 9–15 November 2017; pp. 646–651. [CrossRef]
41. Granell, E.; Chammas, E.; Likforman-Sulem, L.; Martínez-Hinarejos, C.D.; Mokbel, C.; Cirstea, B.I. Transcription of Spanish Historical Handwritten Documents with Deep Neural Networks. *J. Imaging* **2018**, *4*, 15. [CrossRef]
42. Toselli, A.; Juan, A.; González, J.; Salvador, I.; Vidal, E.; Casacuberta, F.; Keysers, D.; Ney, H. Integrated Handwriting Recognition In addition, Interpretation Using Finite-State Models. *Int. J. Pattern Recognit. Artif. Intell. (IJPRAI)* **2004**, *18*, 519–539. [CrossRef]
43. Graves, A.; Fernández, S.; Gomez, F.; Schmidhuber, J. Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks. In Proceedings of the ICML 2006—23rd International Conference on Machine Learning, Pittsburgh, PA, USA, 25-29 June 2006; pp. 369–376. [CrossRef]
44. Graves, A.; Liwicki, M.; Fernández, S.; Bertolami, R.; Bunke, H.; Schmidhuber, J. A Novel Connectionist System for Unconstrained Handwriting Recognition. *IEEE Trans. Pattern Anal. Mach. Intell.* **2009**, *31*, 855–868. [CrossRef]
45. Tensmeyer, C.; Wigington, C.; Davis, B.; Stewart, S.; Martinez, T.; Barrett, W. Language Model Supervision for Handwriting Recognition Model Adaptation. In Proceedings of the 2018 16th International Conference on Frontiers in Handwriting Recognition (ICFHR), Niagara Falls, NY, USA, 5–8 August 2018; pp. 133–138.
46. Toselli, A.H. Reconocimiento de Texto Manuscrito Continuo. Ph.D. Thesis, Universidad Politécnica de Valencia, Valencia, Spain, 2004.
47. Chen, K.N.; Chen, C.H.; Chang, C.C. Efficient illumination compensation techniques for text images. *Digit. Signal Process.* **2012**, *22*, 726–733. [CrossRef]
48. Vinciarelli, A.; Luettin, J. A New Normalization Technique for Cursive Handwritten Words. *Pattern Recognit. Lett.* **2001**, *22*, 1043–1050. [CrossRef]
49. Scheidl, H. Handwritten Text Recognition in Historical Documents. Ph.D. Thesis, Technischen Universität Wien, Vienna, Austria, 2018.
50. Dauphin, Y.N.; Fan, A.; Auli, M.; Grangier, D. Language Modeling with Gated Convolutional Networks. In Proceedings of the 34th International Conference on Machine Learning, JMLR.org, ICML'17, Sydney, Australia, 6–11 August 2017; Volume 70, pp. 933–941.
51. Koerich, A.; Sabourin, R.; Suen, C. Large vocabulary offline handwriting recognition: A survey. *Pattern Anal. Appl.* **2003**, *6*, 97–121. [CrossRef]
52. Pham, V.; Bluche, T.; Kermorvant, C.; Louradour, J. Dropout Improves Recurrent Neural Networks for Handwriting Recognition. In Proceedings of the 2014 14th International Conference on Frontiers in Handwriting Recognition, Hersonissos, Greece, 1–4 September 2014; pp. 285–290. [CrossRef]

53. Voigtlaender, P.; Doetsch, P.; Ney, H. Handwriting Recognition with Large Multidimensional Long Short-Term Memory Recurrent Neural Networks. In Proceedings of the 15th International Conference on Frontiers in Handwriting Recognition (ICFHR), Shenzhen, China, 23–26 October 2016; pp. 228–233. [CrossRef]
54. Moysset, B.; Messina, R.O. Are 2D-LSTM really dead for offline text recognition? *Int. J. Doc. Anal. Recognit.* **2019**, 22, 193–208. [CrossRef]
55. Chen, S.F.; Goodman, J. An Empirical Study of Smoothing Techniques for Language Modeling. *Comput. Speech Lang.* **1999**, *13*, 359–394. [CrossRef]
56. Knerr, S.; Augustin, E. A neural network-hidden Markov model hybrid for cursive word recognition. In Proceedings of the Fourteenth International Conference on Pattern Recognition (Cat. No.98EX170), Brisbane, Australia, 20 August 1998; Volume 2, pp. 1518–1520.
57. De Castro Oliveira, D.W. Deep Multidimensional Recurrent Neural Networks for Offline Handwriting Text Line Recognition. Master's Thesis, Universidade de Pernambuco, Pernambuco, Brazil, 2018.
58. Schuster-Böckler, B.; Bateman, A. An Introduction to Hidden Markov Models. *Curr. Protoc. Bioinform.* **2007**. [CrossRef]
59. Jurafsky, D.; Martin, J.H. *Speech and Language Processing—An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*, 3rd ed.; Prentice Hall Series in Artificial Intelligence; Prentice Hall: Englewood Cliffs, NJ, USA, 2019.
60. Whitelaw, C.; Hutchinson, B.; Chung, G.Y.; Ellis, G. Using the Web for Language Independent Spellchecking and Autocorrection. In Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, Singapore, 6–7 August 2009; Association for Computational Linguistics: Singapore, 2009; pp. 890–899.
61. Li, Y.; Duan, H.; Zhai, C. CloudSpeller: Query spelling correction by using a unified Hidden Markov Model with Web-scale resources. In Proceedings of the WWW'12 21st Annual Conference on World Wide Web Companion, Lyon, France, 16–20 April 2012; pp. 567–568. [CrossRef]
62. Guo, J.; Sainath, T.N.; Weiss, R.J. A Spelling Correction Model for End-to-end Speech Recognition. In Proceedings of the 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Brighton, UK, 12–17 May 2019; pp. 5651–5655.
63. Bassil, Y.; Alwani, M. Context-sensitive Spelling Correction Using Google Web 1T 5-Gram Information. *Comput. Inf. Sci.* **2012**, *5*, 37–48. [CrossRef]
64. Lichtarge, J.; Alberti, C.; Kumar, S.; Shazeer, N.; Parmar, N.; Tong, S. Corpora Generation for Grammatical Error Correction. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Minneapolis, Minnesota, USA, 2–7 June 2019; pp. 3291–3301. [CrossRef]
65. Kurdi, M. *Natural Language Processing and Computational Linguistics 2: Semantics, Discourse, and Applications*; ISTE Wiley: London, UK, 2017. [CrossRef]
66. Taylor, A.; Marcus, M.; Santorini, B. *The Penn Treebank: An Overview*; Springer: Dordrecht, The Netherlands, 2003; pp. 5–22._1. [CrossRef]
67. Flor, M. Four types of context for automatic spelling correction. *TAL* **2012**, *53*, 61–99.
68. Gupta, N.; Mathur, P. Spell Checking Techniques in NLP: A Survey. *Int. J. Adv. Res. Comput. Sci. Softw. Eng.* **2012**, *2*, 217–221.
69. Mishra, R.; Kaur, N. A Survey of Spelling Error Detection and Correction Techniques. *Int. J. Comput. Trends Technol.* **2013**, *4*, 372–374.
70. Hakkinen, J.; Tian, J. N-gram and decision tree based language identification for written words. In Proceedings of the IEEE Workshop on Automatic Speech Recognition and Understanding, 2001 ASRU '01, Madonna di Campiglio, Italy, 9–13 December 2001; pp. 335–338.
71. Godev, C. Word-frequency and vocabulary acquisition: An analysis of elementary spanish college textbooks in the USA. *RLA Revista de LingüíStica TeóRica y Aplicada* **2009**, *47*, 51–68. [CrossRef]
72. Masrai, A. Vocabulary and Reading Comprehension Revisited: Evidence for High-, Mid-, and Low-Frequency Vocabulary Knowledge. *SAGE Open* **2019**, *9*, 1–13. [CrossRef]
73. Yannakoudakis, E.J.; Fawthrop, D. An intelligent spelling error corrector. *Inf. Process. Manag.* **1983**, *19*, 101–108. [CrossRef]

74. Pollock, J.J.; Zamora, A. Collection and characterization of spelling errors in scientific and scholarly text. *J. Assoc. Inf. Sci. Technol.* **1983**, *34*, 51–58. [CrossRef]
75. Wagner, R.A.; Fischer, M.J. The String-to-String Correction Problem. *J. ACM* **1974**, *21*, 168–173. [CrossRef]
76. Church, K.; Gale, W.A. Probability scoring for spelling correction. *Stat. Comput.* **1991**, *1*, 93–103. [CrossRef]
77. Zimmermann, M.; Bunke, H. N-Gram Language Models for Offline Handwritten Text Recognition. In Proceedings of the Ninth International Workshop on Frontiers in Handwriting Recognition, IWFHR '04, Tokyo, Japan, 26–29 October 2004; IEEE Computer Society: Washington, DC, USA, 2004; pp. 203–208. [CrossRef]
78. Mokhtar, K.; Bukhari, S.S.; Dengel, A. OCR Error Correction: State-of-the-Art vs an NMT-based Approach. In Proceedings of the 2018 13th IAPR International Workshop on Document Analysis Systems (DAS), Vienna, Austria, 24–27 April 2018; pp. 429–434.
79. Ioffe, S. Batch Renormalization: Towards Reducing Minibatch Dependence in Batch-Normalized Models. In Proceedings of the 31st International Conference on Neural Information Processing Systems, NIPS'17, Long Beach, CA, USA, 4–9 December 2017; Curran Associates Inc.: Red Hook, NY, USA, 2017; pp. 1942–1950.
80. He, K.; Zhang, X.; Ren, S.; Sun, J. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. In Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV), Santiago, Chile, 7–13 December 2015; pp. 1026–1034. [CrossRef]
81. Jaf, S.; Bradley, S.; Gajbhiye, A.; Al Moubayed, N.; Mcgough, A. An Exploration of Dropout with RNNs for Natural Language Inference. In Proceedings of the 27th International Conference on Artificial Neural Networks (ICANN), Rhodes, Greece, 4–7 October 2018.
82. Ba, J.; Kiros, J.R.; Hinton, G.E. Layer Normalization. *arXiv* **2016**, arXiv:1607.06450.
83. Agarwal, S.; Godbole, S.; Punjani, D.; Roy, S. How Much Noise Is Too Much: A Study in Automatic Text Classification. In Proceedings of the Seventh IEEE International Conference on Data Mining (ICDM 2007), Washington, DC, USA, 28–31 October 2007; pp. 3–12.
84. Graves, A. *Supervised Sequence Labelling with Recurrent Neural Networks*; Studies in Computational Intelligence; Springer: Berlin/Heidelberg, Germany, 2012; doi:10.1007/978-3-642-24797-2. [CrossRef]
85. Efron, B. Better Bootstrap Confidence Intervals. *J. Am. Stat. Assoc.* **1987**, *82*, 171–185. [CrossRef]
86. Wilcoxon, F. *Individual Comparisons by Ranking Methods*; Springer: New York, NY, USA, 1992; pp. 196–202. [CrossRef]
87. Conover, W.J. *Practical Nonparametric Statistics*; John Wiley & Sons: New York, NY, USA, 1971.
88. Hwang, K.; Sung, W. Character-level incremental speech recognition with recurrent neural networks. In Proceedings of the 2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Shanghai, China, 20–25 March 2016; pp. 5335–5339.
89. Glorot, X.; Bengio, Y. Understanding the difficulty of training deep feedforward neural networks. In Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS'10), Sardinia, Italy, 13–15 May 2010; Society for Artificial Intelligence and Statistics: Princeton, NJ, USA, 2010.
90. Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; Salakhutdinov, R. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *J. Mach. Learn. Res.* **2014**, *15*, 1929–1958.
91. Ioffe, S.; Szegedy, C. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In Proceedings of the 32nd International Conference on International Conference on Machine Learning—JMLR.org, ICML'15, Lille, France, 6–11 July 2015; Volume 37, pp. 448–456.
92. Maas, A.L.; Hannun, A.Y.; Ng, A.Y. Rectifier nonlinearities improve neural network acoustic models. In Proceedings of the ICML Workshop on Deep Learning for Audio, Speech and Language Processing, Atlanta, Georgia, 16 June 2013.
93. Tieleman, T.; Hinton, G. Lecture 6.5–RmsProp: Divide the gradient by a running average of its recent magnitude. *COURSERA Neural Netw. Mach. Learn.* **2012**, *4*, 26–31.
94. Stefanovič, P.; Kurasova, O.; Štrimaitis, R. The N-Grams Based Text Similarity Detection Approach Using Self-Organizing Maps and Similarity Measures. *Appl. Sci.* **2019**, *9*, 1870. [CrossRef]
95. Norvig, P. How to Write a Spelling Corrector. 2007. Available online: https://norvig.com/spell-correct (accessed on 17 October 2019).

96. Garbe, W. 1000x Faster Spelling Correction Algorithm. 2012. Available online: https://towardsdatascience.com/symspellcompound-10ec8f467c9b (accessed on 17 October 2019).
97. Kingma, D.P.; Ba, J. Adam: A Method for Stochastic Optimization. In Proceedings of the 3rd International Conference on Learning Representations (ICLR), San Diego, CA, USA, 7–9 May 2015; Conference Track Proceedings: San Diego, CA, USA, 2015.