computers

MDPI

*Article*

# Automatic Spell-Checking System for Spanish Based on the Ar2p Neural Network Model

**Eduard Puerto [1], Jose Aguilar [2,3,4,*] and Angel Pinto [5]**

[1] Grupo de Investigación en Inteligencia Artificial (GIA), Facultad de Ingeniería, Universidad Francisco de Paula Santander, Cúcuta 540001, Colombia; eduardpuerto@ufps.edu.co
[2] Centro de Estudio en Microcomputación y Sistemas Distribuidos (CEMISID), Facultad de Ingeniería, Universidad de Los Andes, Mérida 5101, Venezuela
[3] Grupo de Investigación, Desarrollo e Innovación en Tecnologías de la Información y las Comunicaciones (GIDITIC), Universidad EAFIT, Medellín 050001, Colombia
[4] IMDEA Networks Institute, 28910 Leganés, Madrid, Spain
[5] Grupo de Investigación TESEEO, Universidad del Sinú, Montería 230001, Colombia; anpima1@hotmail.com or angelpinto@unisinu.edu.co
* Correspondence: aguilarjos@gmail.com or aguilar@ula.ve or jlaguilarc@eafit.edu.co or jose.aguilar@imdea.org

**Abstract:** Currently, approaches to correcting misspelled words have problems when the words are complex or massive. This is even more serious in the case of Spanish, where there are very few studies in this regard. So, proposing new approaches to word recognition and correction remains a research topic of interest. In particular, an interesting approach is to computationally simulate the brain process for recognizing misspelled words and their automatic correction. Thus, this article presents an automatic recognition and correction system of misspelled words in Spanish texts, for the detection of misspelled words, and their automatic amendments, based on the systematic theory of pattern recognition of the mind (PRTM). The main innovation of the research is the use of the PRTM theory in this context. Particularly, a corrective system of misspelled words in Spanish based on this theory, called Ar2p-Text, was designed and built. Ar2p-Text carries out a recursive process of analysis of words by a disaggregation/integration mechanism, using specialized hierarchical recognition modules that define formal strategies to determine if a word is well or poorly written. A comparative evaluation shows that the precision and coverage of our Ar2p-Text model are competitive with other spell-checkers. In the experiments, the system achieves better performance than the three other systems. In general, Ar2p-Text obtains an F-measure of 83%, above the 73% achieved by the other spell-checkers. Our hierarchical approach reuses a lot of information, allowing for the improvement of the text analysis processes in both quality and efficiency. Preliminary results show that the above will allow for future developments of technologies for the correction of words inspired by this hierarchical approach.

**Keywords:** spell-checker; text recognition; Ar2p-Text



**Citation:** Puerto, E.; Aguilar, J.; Pinto, A. Automatic Spell-Checking System for Spanish Based on the Ar2p Neural Network Model. *Computers* **2024**, *13*, 76. https://doi.org/10.3390/computers13030076

Academic Editors: Katia Lida Kermanidis, Manolis Maragoudakis and Phivos Mylonas

Received: 12 February 2024
Revised: 28 February 2024
Accepted: 4 March 2024
Published: 12 March 2024



## 1. Introduction

Spelling errors and misspelled words are a big problem in idioms. For example, in Spanish, in street signs and social networks, among other contexts, this is very visible in expressions such as "Dios vendice a mi madre", "solo Dios jusga", "segidme", "alturista", instead of "altruista", "objetibo", and "la vida no es fasil" y "sonrrisa". Currently, some systems perform an analysis, an extraction, an annotation, and a linguistic correction (based on dictionaries or in statistical analyses) to perform tasks as diverse as lemmatization [1,2], morphosyntactic labeling [3,4], syntactic analysis [3], sentiment analysis (or opinion mining), and conceptual annotation [2], among others.

Although there are some works on misspelled words in Spanish texts in order to recognize and correct them (see Section 2), they are not yet efficient enough, particularly

when the texts are large or the words have a certain complexity in their structure [5–7]. A good example is the large number of word errors contained in millions of tweets and other massive data media [3,4]. Thus, efficient approaches based on a lexical analysis of the syntax of words in Spanish are interesting approaches that are not found in the literature to address this issue.

In this work, we present an automatic system for the orthographic revision of texts in Spanish, for the recognition of misspelled words, and for their automatic corrections, based on PRTM [8]. This work presents a new method for the detection of misspelled words, in a way very similar to how the human brain solves misspellings (specifically, the neocortex), called Ar2p-Text, which reuses information to propose an efficient approach based on the lexical analysis of the syntax of the words in Spanish. Particularly, Ar2p-Text is based on Ar2p, a neural network model that represents the form just like the brain (neocortex) works using recognition modules of patterns [9–11], according to the PRTM theory [8]. Ar2p-Text uses strategies and modules of recognition and correction that allow for the carrying out of different processes of detection and correction of orthographic errors. In synthesis, the architecture of Ar2p-Text is characterized by having recognition module hierarchies, which increase the levels of complexity; i.e., the pattern recognition modules that constitute the lowest-level levels (or $X_{j-1)}$, will always be of less complexity than the modules of the upper-level levels (or $X_j$, for j = 1,. . ., m). In addition, Ar2p-Text has a supervised definition of the weights assigned to the variables used for recognition based on adaptive mechanisms inspired by previous works [10,12,13]. Therefore, the main contribution of this work is to propose a new system to recognize and correct misspelled words in Spanish texts based on AR2P (following the PRTM theory), which (i) is highly recursive and uniform; (ii) is based on a recognition process that uses a hierarchy of patterns that is self-associating; (iii) is adaptable because it can learn new patterns (words); (iv) can analyze large Spanish texts with words with a certain complexity in their structure; and (v) is a new spell-checking approach that follows the highly scalable and efficient human model that is very different from other spell-checking approaches that do not rely on atomic abstractions and recursive processes to generalize information. As far as we know, there is no previous work based on PRTM, and less applied to Spanish.

This paper is organized as follows: Section 2 describes related works. Section 3 describes the PRTM theory, which is the basis of AR2P text. Section 4 makes a formal description of the general architecture of Ar2p-Text, its data structure (pattern recognition modules), and its computational model. Section 5 shows the experiments for the treatment of digital texts, the database used, the quality metrics, and the performance evaluation. Finally, Section 6 describes conclusions and future works.

## 2. Related Works

Currently, there are different approaches for lemmatization, morphosyntactic analysis, and sentiment analysis, among others. Particularly, we are interested in the spell-checking (auto-correction) problem.

Some works in this domain are STILUS [14], which distinguishes four types of errors: grammatical, orthographic, semantic, and stylistic. The system has modules specifically dedicated to each one of them. In the case of orthographic revision, STILUS performs the correction of words in three stages: the generation of alternatives to the wrong word, the weighting of alternatives, and the arrangement of alternatives. Another system is ArText, which is a prototype of an automatic help system for writing texts in Spanish in specialized domains [5]. The system has three modules: the first module handles aspects of structure, content, and phraseology. The second module is for format and linguistic revision. Finally, the last module allows the users to linguistically revise their text. XUXEN is a spell-checker/corrector [6], which has been defined based on two morphological formalisms. It uses a highly flexed standardized language with a broad relationship between nouns and verbs and a lexicon that contains approximately 50,000 items, divided among verbs and other grammatical categories.

On the other hand, Valdehíta [3] proposes a spell- and grammar-checker algorithm for texts where the possible mistakes are not detected by tagging and parsing, but by statistical analysis, comparing combinations of two words used in the text to a hundred-million-word corpus. Ferreira et al. [1] propose a spell-checker where the text is processed according to two ways: word by word and as a chain in search of complex error patterns. In [15], a corpus is presented called JHU FLuency-Extended GUG corpus (JFLEG), which can evaluate grammatical errors. It uses different levels of a language, with holistic fluency edits, both to correct grammatical errors and to make the original text more native-sounding. Also, Singh and Mahmood [7] present a general approach to various uses of natural language processing (NLP) (translation and recognition) using modern techniques such as deep learning techniques. Finally, there are other books and papers in the literature like [16], but there are few systems that deal with lexical or syntactical errors in Spanish, like [3,5,14].

Li et al. [17] developed a multi-round error correction method with ensemble enhancement for Chinese Spelling Check. Specifically, multi-round error correction follows an iterative correction pipeline, where a single error is corrected at each round, and the subsequent correction is conducted based on the previous results. Cheng et al. [18] defined an English writing error correction model to carry out an automatic checking and correction of writing errors in English composition. This paper used a deep learning Seq2Seq_Attention algorithm and a transformer algorithm to eliminate errors. Then, the output of each algorithm is sent to an n-gram language algorithm for scoring, and the highest score is selected as the output. Ma et al. [19] proposed a confusion set-guided decision network based on a long short-term memory model for spoken Chinese spell checking. The model can reasonably locate the wrong characters with a decision network, which ensures the bidirectional long short-term memory pays more attention to the characteristics of the wrong characters. This model has been used to detect and correct Chinese spelling errors. Finally, Hládek et al. [20] presented a survey of selected papers about spelling correction indexed in Scopus and Web of Science from 1991 to 2019. The survey describes selected papers in a common theoretical framework based on Shannon's noisy channel. They finish with summary tables showing the application area, language, string metrics, and context model for each system.

As we can see in the previous works, there are not many works related to Spanish. We confirmed that in the existing literature, the language where the most work has been performed on spelling correction is Chinese. Additionally, there are also no works that are based on a hierarchical approach to pattern recognition (in our case, words) as a basic mechanism that allows for the reuse of text (patterns) as an efficient way that allows for the recognition of many words, some of them complex. In our case, Ar2p-Text allows for it, because the Ar2p neural model on which it is inspired is based on the PRTM theory that emulates the behavior of the neocortex area of the brain that follows these principles (the next sections detail these theories/models).

## 3. PRTM Theory

This model has been described in several previous works; here, we present a short summary. The pattern recognition theory of mind (PRTM) describes the procedure followed by the neocortex, according to some of the aspects of the functioning of the human brain such as [8,21] wherein (i) our memory is handled as a hierarchy of patterns and (ii) if we only perceive a part of a pattern (through sight, hearing, or smell), we can recognize it. Also, PRTM presupposes several hypotheses on the structure of the biological neocortex such as (i) a uniform structure of the neocortex, called the cortical column, which is the module of recognition for PRTM, and (ii) the recognition modules are connected all the time to each other. Figure 1 describes a pattern recognition module of PRTM.

In Figure 1, (a) each of the dendrites sends information (parameters of size, importance, and variability) toward the interior of the module, indicating the presence of a pattern in the lower level or outside. (b) When there is recognition, an output is generated. On the other hand, (c) if a pattern recognizer of a higher level receives a signal coming from almost

all the recognizers that make up its input, this recognizer is likely to send an exciting signal toward the lowest level of the missing pattern recognizers (via a dendrite) to indicate that it is expecting them. In addition, there are inhibitory signals from both (d) a lower-level recognition space and (e) a higher-level recognition space, which can inhibit the process of recognition of a pattern. They are the basis of Ar2p-Text, our system of recognition of texts.

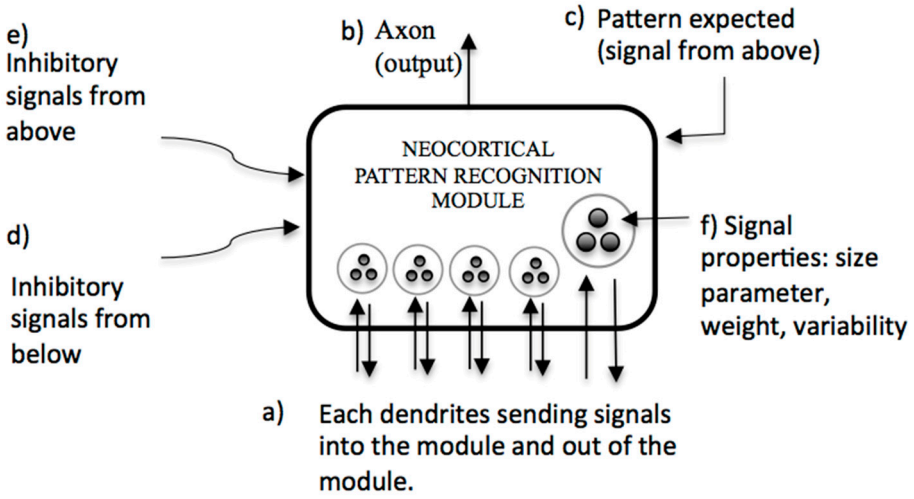


**Figure 1.** Neocortical pattern recognition module [8,9].

## 4. Formalization of the Ar2p-Text Neural Model

In this section, we describe the design of the proposed Ar2p-Text. Ar2p-Text is an extension of the Ar2p neural model [10,16] for the context of Spanish text analysis. The Ar2p neural model has previously been successfully used in different contexts [9,11]. Next, we will describe the aspects of the Ar2p neural model, clarifying its extensions for the case of Ar2p-Text. Ar2p is a neural network model based on the PRTM theory [8].

### *4.1. Formal Definition of Ar2p-Text*

A pattern recognition module is defined in Ar2p by a 3-tuple, which is similarly used by Ar2p-Text [9]. Γ$\rho$ notation is used to represent the module that recognizes the $\rho$ pattern ($\rho$: shapes, letters, and words, etc.).

$$\Gamma\rho = \langle E, U, So \rangle$$

where $E$ is an array defined by the 2-tuple $E = \langle S, C \rangle$ (see Table 1), $S = \langle$Signal, State$\rangle$ is another array of the set of signals of the pattern recognized by Γ with its states, $C$ is another array with information of the pattern described by the 3-tuple $C = \langle D, V, W \rangle$, and D are the descriptors of Γ, V is the vector with the possible values of each descriptor in D, and W is the relevance weight of the descriptors in the pattern $\rho$. Additionally, there is a threshold vector U used by the module (Γ) to recognize the pattern.

Table 1 constitutes one artificial neuron, which is a neocortical pattern recognition module according to the PRTM theory. In the Ar2p neural model, each neuron/module can acknowledge and observe every aspect of the input pattern *s()* and how the different parts of the data of the input pattern may or may not relate to each other.

Two types of thresholds were used: ΔU1 for the recognition using key signals and ΔU2 for the recognition using total or partial mapping. The ΔU1 threshold is stricter than ΔU2 because the process based on key signals uses few signals. Finally, each module generates an acknowledgment signal or a request signal to the lower levels (So). *So* as a request signal is the input signal *s()* of the modules of the lower levels. *So*, as an acknowledgment signal, is sent to its higher levels to modify their states of the signal to "true".

**Table 1.** Pattern recognition module: Matrix E = <S, C>.

| E | | | | |
|---|---|---|---|---|
| **S** | | **C** | | |
| **Signal** | **State** | **Descriptor (D)** | **Domain (V)** | **Weight (W)** |
| 1 | False | Descriptor1 | <possible values of the descriptor> | [0,1] |
| 2 | False | Descriptor2 | <possible values of the descriptor> | [0,1] |
| 3 | False | Descriptor3 | <possible values of the descriptor> | [0,1] |
| ... | ... | ... | ... | ... |
| N | False | DescriptorN | <possible values of the descriptor> | [0,1] |
| U: <ΔU1, ΔU2> | | | | |

Thus, a pattern is represented as a set of lower-level sub-patterns that conform to it (N descriptors), and in turn, it also serves as a sub-pattern of a higher-level pattern. N depends on the descriptors of the pattern to recognize. W is normalized [0,1], and ΔU1 or ΔU2 are thresholds that must be overcome in order to recognize the pattern. These values are defined according to the domain of application.

The previous definitions have been defined for the neural model Ar2p [10], but they are maintained for the case of Ar2p-Text. In the context of Ar2p-Text, the main patterns to recognize ($\rho$) are letters, words, special signs, and numbers.

### *4.2. Text Analysis in Ar2p-Text*

In this section, we describe the general model of Ar2p-Text. Again, Ar2p-Text follows the same formal description of the Ar2p model [9], but it is instanced for the specific case of text analysis. Particularly, the hierarchical system describes the iterative and recursive processes for the recognition and correction of words with Ar2p-Text. Each layer is an interpretative space $\chi_i$, from i = 1 to m, such that $\chi_1$ is the first level to recognize atomic patterns (e.g., letters or letterforms) and $\chi_m$ is the last level to recognize complex patterns (e.g., words and compound words). Each level has $\Gamma_{ji}$ recognition modules (for j = 1, 2, and 3... # of modules at level i). Finally, $\chi_{ji}$ is the pattern that is recognized at level i by module j.

Thus, Ar2p-Txt is a pattern recognition system based on the hierarchical architecture of a neural network. The multiple hidden layers are the recognition spaces of i-level or the levels of recognition of the complex patterns ($\chi_i$). This is how Ar2p is capable of finding extremely complex patterns using bottom-up or top-down approaches.

### *4.3. Strategies of Checking/Correction/Recognition in Ar2p-Text*

An important modification with respect to the Ar2p model is how signal thresholds are used in Ar2p-Text. Ar2p-Text uses two strategies for the correction process: the first one uses key signals; the other uses partial signals, and both use a threshold of satisfaction and the importance of the weights of signals. In this way, the recursive model allows for the decomposition of the problem of recognition of patterns into simpler patterns, which makes it possible to analyze very complex words.

Particularly, the first strategy, named *pattern matching by key signals* is based on the relevance weights of the input signals identified as keys [11,22]. The *partial pattern matching* strategy utilizes the total or partial presence of the signals. A signal is key when it represents information that allows a pattern to be recognized quickly. For example, the final letter "r" in infinitive verbs could be taken as a key.

**Definition 1.** *key signal. $s_i$ is a key signal in the $\Gamma$ module when its relevance weight has a greater or equal value to the mean weight of all the signals in $\Gamma$ (see Equation (1)).*

$$\forall S_i \in S(\Gamma)\ if\ \left[w(S_i) \geq w_{average} S(\Gamma)\right] \rightarrow S_i \in Key_\Gamma \quad (1)$$

**Theorem 1.** *Strategy by key signals. A ρ pattern is recognized by key signals if the mean of the key signals recognized is superior to ΔU1. It utilizes the descriptors (signals or sub-patterns) with the greater relevance weight. The equation is:*

$$\frac{\sum_{i=1 \cap State(S_i=True) \cap S_i \in Key_\Gamma}^{n} w(s_i)}{|Key_\Gamma|} \geq \Delta U1 \rightarrow S_o \quad (2)$$

**Theorem 2.** *Strategy by partial mapping. This strategy validates if the signal number in Γ is superior to ΔU2. The equation is:*

$$\frac{\sum_{i=1 \cap State(s_i=true)}^{n} w(s_i)}{n} \geq \Delta U2 \rightarrow S_0 \quad (3)$$

This process is performed for each module of each level of recognition $X_i$ during the recognition process.

### 4.4. Computational Model of Ar2p-Text

Again, Ar2p-Text follows the same general computational process of the Ar2p model [9], but it introduces certain modifications for the specific case of text analysis. Next, the algorithm of Ar2p-Text is presented with these modifications. This algorithm has two processes: A first process, the bottom-up process, for the atomic patterns, such that the output signals of the recognized patterns go to the modules of which they are part of the top levels to activate them if they pass a recognition threshold [9]. The other process is a top-down one for the input pattern by decomposition. The top-level module uses the modules of recognition of the lower level, and then they recursively do the same.

The algorithm works as follows: The input text is received (y = s(): sentences and word (s)). Then, this input is broken down into sub-patterns that are stored in L (e.g., if it is a sentence, then it is decomposed into words, and so on for the rest). The level of depth of decomposition depends on the level of detail and analysis with which the pattern is recognized. Once the pattern has been simplified, then the Xi level of the hierarchy is determined (this is performed through a metric that relates the input pattern to some level, see [11]), where the recognition of the input pattern (*y*) should start and end. Subsequently, L requests that patterns for recognition of (*y*) are created. At this point, the process of recognition and correction can continue through two possible paths: a bottom-up or a top-down process. If the input pattern is made up of lower-level signals (i.e., sub-patterns of the lower level) or it has atomic signals recognized (i.e., all the lower levels are Xi = 1), then a bottom-up process occurs, otherwise, it is a top-down process. In the case of the bottom-up process, the recognition of (y) is calculated using the strategies defined above (see Equations (1)–(3)). If the calculation was successful by any strategy and the recursive recognition process is already at the level of the initial input pattern s(), then it sends an *So* of recognition (y) as the final output of the system. If it is not yet in the Xm level of the initial input pattern, then an *So* is created and sent to the immediately higher level. If the sub-patterns of the pattern have not yet been recognized (letter, word, etc.), then the top-down process starts and L requests that acknowledgment from (y) to the lower levels is sent. Then, it generates recursive calls. Next, it receives the L responses of the lower levels, and the recognition of the (y) is calculated. If the calculation is successful by any strategy and it is the last level of the hierarchy Xm, then it sends an *So* recognition (y) like the system output. Otherwise, it continues going down with the recognition.

To understand the process better, next, we explain how the algorithm works in a more elaborate way. But first it is important to clarify that Ar2p-Text for text analysis can receive as input shapes, words, letters, numbers, special characters, sentences, etc. The input will depend on the hierarchy level where the neuron is. For example, in the lower levels, it will receive shapes, letters, numbers, or special characters and in the upper levels, it will receive words, sentences, or more elaborate texts.

Suppose we need to recognize the word "Casa/House" (see Figure 2). In this case, Ar2p-Text receives "Casa" as the input and should return as recognition output (y = Casa). Figure 2 schematically shows the levels of Ar2p-Text (and its recognition modules) that intervene during the recognition process. Specifically, there are three levels in the hierarchy; The first level ($X_1$) has the modules ($\Gamma_{j1)}$ that recognize the atomic patterns (curves or lines) that will be used to construct the letters (that is, they are indivisible patterns). The second level ($X_2$) has modules ($\Gamma_{k2}$) that recognize letter patterns (for example, "a", "e", "C", "s", "M", and "z"). Finally, the top level ($X_3$) contains the word pattern recognition modules $\Gamma_{l3}$ (e.g., "Casa/House" and "Manzana/Apple").

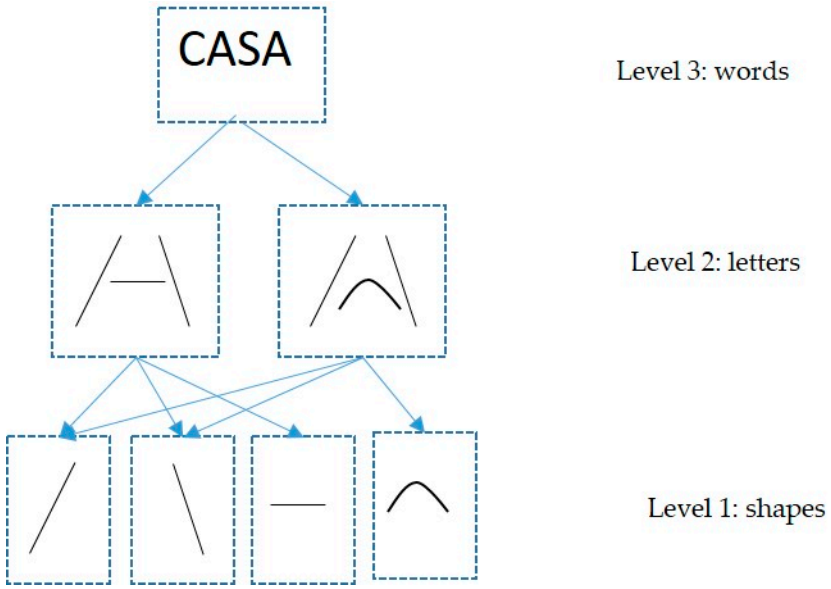


**Figure 2.** Three levels to recognize "*Casa*/House".

As we can see in the example above, our approach could be used on handwritten or printed texts. In the case of handwritten texts, level 1 would have the figures used by individuals to write their letters, and in the case of printed text, level 1 would have the figures that describe the different fonts of the letters used (for example, in the case of words, for fonts such as Calibri and Arial). Thus, it can recognize different handwriting (people who write very differently) or print written with different fonts. The instantiation of the "Casa" pattern recognition module in the top level is shown in Table 2.

**Table 2.** Recognition module of the "Casa" pattern: Matrix E = "Casa".

| E | | | | |
|---|---|---|---|---|
| **S** | | **C** | | |
| **Signal** | **State** | **Descriptor (D)** | **Domain (V)** | **Weight (W)** |
| 1 | True | *C* | <possible forms of *C*> | 0.9 |
| 2 | True | *a* | <possible forms of *a*> | 0.8 |
| 3 | True | *s* | <possible forms of *s*> | 0.8 |
| 4 | False | *a* | <possible forms of *a*> | 0.5 |
| <ΔU1 = 0.8, ΔU2 = 0.6> | | | | |

The algorithm receives $y = s()$ = "Casa". It decomposes the word "Casa": <"C", "a", "s", "a">. Then, it defines the $X_i$ level of the hierarchy to start the recognition process for (y). For this example, $X_{i=3}$. When the initial patterns are atomic (i.e., of the level $X_{i=1}$), then the *bottom-up* process starts. But as $y$ is not an atomic pattern (level $X_{i=3}$), L requests acknowledgment of $y$, and <$y_1$ = C, $y_2$ = a, $y_3$ = s, $y_4$ = a> are sent at the lower level $X_{i=2}$. At this point, the recursive process of recognition starts, such that each $y_i$ of the list L becomes the input signal to the level $X_{i=2}$.

For $y_1 = C$, the algorithm begins the process of recognition of *s()* = ("C") for the $X_{m=2}$ level. Again, the letter "C" is decomposed in its sub-patterns. In this case, "C" will be divided into two parts to facilitate its interpretation (<$y_1$ = volute and $y_2$ = base) (see Figure 3). In the case of "A", it will be divided into two parts (<$y_1$ = stick and $y_2$ = base).

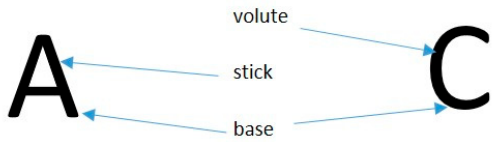


**Figure 3.** Parts of "A" and "C".

Then, the level of the hierarchy to start the process of recognition for this pattern is defined, which is ($X_2$). The algorithm sends the L requests it determined for $y_1$ and $y_2$, and waits.

When shapes such as $y_1$ = "|" (i.e., the stick) are recognized, the algorithm identifies an atomic pattern (of the level $X_1$), and the bottom-up process is carried out. When a successful calculation is determined (i.e., a Γ recognizes "|") and it is not the last level of the hierarchy, the algorithm sends an output ($S_o$) to indicate the recognition to the top level ($X_2$). It is the same for the rest of the atomic patterns (in this case, $y_2$). If all atomic patterns are recognized ($y_1$ = s() = "*stick*" and $y_2$ = s()$_-$ = "*base*"), the responses ascend into the hierarchy and modify the states of the corresponding signals. These signals are received in $X_2$ to calculate the recognition of their patterns.

With a successful recognition of the letter "A" at level $X_2$, the algorithm creates and sends a signal $S_o$ of recognition to its upper level $X_3$. The same procedure is followed for the rest of the patterns of this level. If all the patterns of level 2 are recognized, the last level receives the recognized signals and calculates the recognition.

To understand this sub-process better, next, we explain how the algorithm works in a more elaborate way. Suppose ΔU1 = 0.8, and three of the four signals are recognized in $X_1$. Then, according to Theorem 1, we must define the active key signals (see Equation (1)). The key signals are 1, 2, and 3. Now, Equation (2) defines if the pattern is recognized with these signals. They overcome the threshold (0.83 < 0.8), and if it is the last level, it carries out reinforcement learning (see [9] for more details) and generates the output signal $S_o$ that is the output signal of the system "recognized pattern <Casa>".

## 5. Experiments

This section describes the results and evaluation of Ar2p-Text and compares them with other works. The state of the art (SOTA) was sought in models, techniques, or datasets in the automatic spelling check in Spanish against which to compare, but as can be seen in recent works, nothing exists for the case of Spanish (see Section 2). Even for other languages, it is very difficult to obtain, as indicated [17–19]. Therefore, it was decided to compare with three tools that automatically correct texts in Spanish, carefully studying the four categories of misspelled words defined by Cook [23]. To generalize the test, a group of news texts from a Spanish newspaper (32 texts) were then used as input and compared with the same tools. Finally, an analytical comparison was performed with an n-gram language model to evaluate its robustness.

Thus, to compare, three systems were chosen: SpanishChecker [24], STILUS [14], and Microsoft Word. These three systems were chosen because they are tools designed to analyze spelling, as well as find basic grammar and stylistic mistakes, in Spanish texts. All show the errors automatically. For these test scenarios, several paragraphs were made artificially with misspelled words in Spanish. In these tests, the inputs, i.e., the paragraphs, are introduced as an array of words to Ar2p-text, while in the other systems, they are introduced as a plain text file. Finally, the standard metrics of Precision (P), Coverage (C), and F-measure (F) [22] were used during these experiments. These tests allowed for a comparison of the performance of these systems.

Finally, Ar2P has two hyperparameters that must be optimized, which are the thresholds to determine when a word is recognized. These parameters were optimized using a hyperparameter optimization scheme. On the other hand, Ar2P was trained with a vocabulary of more than 100 texts extracted from Spanish websites that cover almost 100 thousand Spanish words. From there, the experiments presented below were carried out.

### *5.1. Treatment of Types of Spelling Errors Using Ar2p-Text*

This section shows how Ar2p-text works in the treatment of misspelled words according to the four categories defined by Cook [23] by the omission of letters (e.g., instead of writing "limpia", "lipia" is written); aggregates of letters (e.g., instead of writing "salir", "salire" is written); confusion of letters (e.g., instead of writing "campesinos", c is changed to p and written as "pampesinos" or instead of writing "chocolate", t is changed to l and/or a is changed to e and written "chocotate" or "chocolete"); confusion of letters with similar symmetric orientation (e.g., it is difficult to clearly distinguish letters with the same symmetric orientations, such as "d-b" and "p-q-g" and instead of writing "debe", "dede" is written, or instead of "zapato", "zagato" is written); and investment of letters (e.g., instead of writing "juega", "gueja" is written). These errors are solved word by word; i.e., the text string is divided into independent words for processing. The words are recognized by Ar2p-Text through the modules that were activated by the recognized signals (see Sections 4.1 and 4.2) and the strategies of checking/correction/recognition (see Section 4.2) as described above.

Several studies justify that the spelling errors considered in our study reflect the typical spelling errors made by human beings. As we said before, Cook suggests four classic categories of misspelled words [23]. In [25], they analyzed around 76 K misspellings in real-life texts of humans (see Table 3). The majority of errors tend to be insertions, deletions, transpositions, and letter replacements. They also found that the most frequent misspellings in Spanish are 1. replacement of a lower case for an upper case at the beginning of a proper noun; 2. omissions (mainly of an accent or one character); 3. addition of a letter; 4. replacement of one character; and 5. transposition of a letter.

**Table 3.** Spelling error average produced by humans [25].

| Type of Error | Percentages |
|---|---|
| Insertion or addition of one character (e.g., aereopuerto → aeropuerto) | 4.7% |
| Omission of diacritics (e.g., dia → día) | 51.5% |
| Omission of one character (e.g., mostar → mostrar) | 6.8% |
| Substitution of one character | 4.1% |
| Transposition or repetition of the same letter (e.g., Interpetración → interpretación, movimineto → movimiento, and dirrección → dirección) | 2.8% |
| Cognitive errors (biene → viene) | 5.9% |

Table 3 shows that 51% of the spelling errors found are omissions of a diacritic sign in a vowel. By the way, the same types of errors are also made in other languages. For example, work [26] shows that Iraqi students studying English also make these errors. The study is based on 30 students. The types of errors, frequency, and percentages are shown in Table 4.

**Table 4.** Frequency and percentages of spelling errors in a total number of 1109 words [26].

| Type of Error | Frequency | Percentages |
|---|---|---|
| Insertion | 84 | 24% |
| Omission | 182 | 53% |
| Substitution | 62 | 18% |
| Transposition | 16 | 5% |
| Total | 344 | 100% |

The students have 84 insertion spelling errors (24%) and 182 omission spelling errors (53%). Also, there are 62 substitution spelling errors (18%) and 16 transposition spelling errors (5%). Thus, the majority of the spelling errors are in omission and insertion (77% of all the errors). Finally, in [27], another real scenario is presented. In this case, the participants are 40 learners who were studying English language and literature (see Table 5).

**Table 5.** Frequency and percentages of spelling errors for the scenarios proposed in [27].

| Type of Error | Frequency | Percentages |
|---|---|---|
| Insertion | 20 | 8.6% |
| Omission | 59 | 25.3% |
| Substitution | 41 | 17.6% |
| Transposition | 10 | 4.3% |
| Others | 103 | 43.7% |
| Total | 233 | 100% |

Table 5 shows that the commonest type of spelling error is omission (25.3%), followed by substitution (18%).

#### 5.1.1. Correction of Words Due to the Addition of Letters

Misspelled words by the addition of letters anywhere in the word are corrected by Ar2p-Text based on the upper-level modules that are activated by the recognized signals. For example, if the misspelled word is "Ambrousio", where the sixth letter "u" was included in the name, then it will be corrected via the recognized module "Ambrosio". Next is a paragraph that has 205 words, and more than 5% of the words have errors (the real errors will appear marked in bold type).

List (L1) = {"*La* ***entrdada*** *de Isabel Pantoja en "Sálvame" ocurrida la semana pasada, fue la guinda a meses de silencio de la* ***tonadilleera****. Una representación que merecería un Goya a una cantante que ha hecho de su vida el mejor melodrama de la historia de la* ***tellevisión*** *española. No es la primera vez que Isabel coge el* ***teléfonoo*** *y llama a un programa para soltar toda la bilis que lleva dentro, toda la angustia que* ***ssiente*** *cada vez que ve a su hija Chabelita pasearse por los platós, contando* ***laas*** *miserias de una familia que hace tiempo decidió que es mejor vivir. Al oír a Isabel repartir estopa a diestro y siniestro, sentí* ***vergüeenza*** *y pena. Vergüenza, porque todo lo que sea* ***exhiibición*** *impúdica de la vida privada de las personas, me la produce. Pena, porque puedo* ***eentender*** *su desesperación, aislada como vive, sin apenas amigos, casi olvidada de un público que* ***poese*** *a que la sigue admirando por la gran artista que ha sido, se va apartando de su lado al ver en lo que se ha convertido la "viuda de España": una mujer* ***amargaada****, sin otro clavo al que aferrarse que a sus recuerdos más íntimos, tanto en su papel de madre como de* ***abueila***"}

Table 6 shows the results for the different systems and the percentage of words with errors.

According to the results in Table 6, Ar2p-Text detects very well all the real errors by the addition of letters "*entrdada*", "*tellevisión*", "*teléfonoo*", etc. SpanishChecker instead generates three false positives that correspond to the names "Pantoja", "Goya", and "Chabelita". STILUS® obtains similar results with the two false positives "Pantoja" and "Goya", like Microsoft Word with "Chabelita" and "Goya". Ar2p-Text recognizes the words "*Chabelita*", "*Goya*", and "*Pantoja*" *as well-written words.* On the other hand, the percentage of words with errors does not affect the order in the quality of the results obtained by the techniques. The best results are obtained with Ar2p-Text, with a quality higher than 90%.

**Table 6.** Results of the different corrections for L1.

| Methods | % Words with Errors | Detected Errors | False Negatives | False Positives | P | C | F |
|---|---|---|---|---|---|---|---|
| Ar2p-Text | 5% | 12 | 0 | 0 | 100% | 100% | 100% |
| | 10% | 24 | 0 | 0 | 100% | 100% | 100% |
| SpanishChecker® | 5% | 15 | 0 | 3 | 83% | 100% | 90% |
| | 10% | 30 | 1 | 5 | 80% | 95% | 89% |
| STILUS® | 5% | 14 | 0 | 2 | 88% | 100% | 93% |
| | 10% | 28 | 1 | 3 | 85% | 95% | 91% |
| Microsoft Word | 5% | 13 | 0 | 2 | 92% | 100% | 95% |
| | 10% | 28 | 0 | 4 | 89% | 100% | 94% |

5.1.2. Correction of Words Due to the Omission of Letters

Misspelled words by the omission of letters anywhere in the word are corrected by Ar2p-Text based on the upper-level modules that are activated by the recognized signals. For example, if the misspelled word is "mbrosio" where the first letter of the name "A" was omitted, it will be corrected via the recognized module "Ambrosio". Next is a paragraph that has 203 words, and 5% of the words have this error (the real errors will appear marked in bold type and are underlined).

List (L2) = {"*El expresidente Aznar, hoy en el Congreso. La derecha sin complejos,* ***efectivamete.*** *Pero no lo que la izquierda quiere que se entienda con ello, es decir, la* ***reación*** *y la caspa sin maquillajes; simplemente una derecha que no se acompleja cuando la izquierda pretende* ***sometela*** *a su habitual tratamiento de superioridad moral. El momento con el líder de la* ***podemia*** *fue, en este sentido, una* ***interesate*** *lección. Se presentó Pablo Iglesias con su aire habitual de fiscal soviético,* ***disiulando*** *la falsedad fáctica de la mayoría de sus* ***pregutas*** *con una dicción seria y reposada, más reposada aún desde la toma de hipoteca. Aznar replicó con facilidad, no en vano las* ***afimaciones*** *implícitas de Iglesias parten de una fake news -aunque la más exitosa de la democracia española-, esto es, que el Partido Popular ha sido condenado por corrupción. Pero el momento crucial no tuvo que ver con los hechos, sino con las* ***perceciones****. Fue cuando le dijo: «Señor Iglesias, su populismo no me impresiona» y empezó a* ***abochornale*** *con la exhibición de sus vínculos con partidos hermanos, venezolanos o iraníes. A Iglesias no le quedó más remedio que correr a* ***refugiase*** *en la patria, diciendo que le avergüenza patrióticamente tener un expresidente así."*}

Table 7 shows the results for the different systems and the percentage of words with errors.

**Table 7.** Results of the different corrections for L2.

| Methods | % Words with Errors | Detected Errors | False Negatives | False Positives | P | C | F |
|---|---|---|---|---|---|---|---|
| Ar2p-Text | 5% | 10 | 0 | 0 | 100% | 100% | 100% |
| | 10% | 19 | 0 | 0 | 100% | 100% | 100% |
| SpanishChecker® | 5% | 14 | 0 | 4 | 77% | 100% | 87% |
| | 10% | 23 | 2 | 6 | 78% | 96% | 85% |
| STILUS® | 5% | 10 | 0 | 0 | 100% | 100% | 100% |
| | 10% | 21 | 1 | 1 | 96% | 96% | 96% |
| Microsoft Word | 5% | 10 | 0 | 0 | 100% | 100% | 100% |
| | 10% | 21 | 0 | 2 | 94% | 100% | 96% |

According to these results, Ar2p-Text detects very well all the errors for omitted letters, just like Microsoft Word and Stilus. SpanishCheck instead generates four false positives that correspond to the words "expresidente", "Aznar", "populismo", and "patrióticamente" (two false positives) and omits "Congres" (one false negative). In this case, the quality of Ar2p-Text is still very good with 5% or 10% errors in the words, but that does not happen

with the other techniques, which begin to show performance degradation with 10% errors in the words.

5.1.3. Correction of Words Due to the Changing of a Letter (Cognitive Errors)

Misspelled words by changing of letters anywhere in the word are corrected by Ar2p-Text based on the upper-level modules that are activated by the recognized signals. For example, if the pattern to be recognized is "ambrocio" and there is a recognized module related to the word "Ambrosio", then Ar2p-Text recognizes it and corrects the two orthographic errors by substitution: the proper names start with capital letters, and Ambosio is written with "s" and not "c". It will be corrected, changing the letter "c" to "s" and changing the first letter to upper case. Those changes are made via the recognized module "Ambrosio". As this error is very frequent, the following paragraph will be analyzed with 20% of errors (the real errors will appear marked in bold type and are underlined).

List (L3) = {"*Lo que empezó como un simple comentario en* ***Facebuok*** *sobre por qué no unir fuerzas para ayudar a los médicos de la región a afrontar el coronavirus COVID-19, se convirtió en una* ***crusada*** *cucuteña para* ***confexionar*** *inicialmente 630 trajes de* ***vioseguridad****, que se entregarán sin costo alguno a los centros clínicos de la ciudad. Con asombro y satisfacción por la* ***acojida*** *que ha tenido esta propuesta, que empezó a materializarse el* ***biernes*** *por la noche, Beatriz Oquendo, una de las cabezas de esta iniciativa, aseguró que por lo menos unos 500 cucuteños se han querido sumar. Un* ***trage*** *de muestra de los que se utilizaron en la región cuando se desató el brote del H1N1, fue el punto de partida para la creación de los nuevos diseños, que cubren de* ***piez*** *a cabeza al que los utilice. La vestimenta está* ***elavorada*** *con telas quirúrgicas y velcro a cambio de cierres, pues por costos resultó más* ***varato*** *conseguir este otro material. "Esperamos poder mejorar estos modelos, sacar más tallas y poder llevar más unidades a otros municipios, pues me han escrito de otros lugares del departamento que requieren estos trajes", aseguró. "Somos una región de confeccionistas y si todos ponemos de nuestra parte podemos ayudar enormemente en esta* ***emergensia****". "Estuvimos en una carrera contrarreloj para comprar el resto de* ***inzumos*** *antes de que cerraran la mayoría de los* ***almasenes****. Ya cortamos las muestras y las distribuimos por toda la ciudad y esperamos que este lunes (hoy) nos las entreguen para poderlas donar", indicó Oquendo"}

Table 8 shows the results for the different systems and the percentage of words with errors.

**Table 8.** Results of the different corrections for L3.

| Methods | % Words with Errors | Detected Errors | False Negatives | False Positives | P | C | F |
|---|---|---|---|---|---|---|---|
| Ar2p-Text | 5% | 14 | 0 | 0 | 100% | 100% | 100% |
| | 10% | 23 | 0 | 1 | 100% | 98% | 99% |
| SpanishChecker® | 5% | 16 | 2 | 5 | 76% | 88% | 81% |
| | 10% | 33 | 3 | 7 | 75% | 87% | 80% |
| STILUS® | 5% | 15 | 0 | 2 | 88% | 100% | 93% |
| | 10% | 26 | 0 | 3 | 87% | 100% | 92% |
| Microsoft Word | 5% | 14 | 1 | 0 | 100% | 92% | 95% |
| | 10% | 26 | 2 | 1 | 95% | 89% | 93% |

Ar2p-Text detects the 14 errors. Microsoft Word detects a false negative, "Confexionar". Although SpanishChecker® identifies the errors, many of its correction recommendations are far from the correct word syntactically and semantically. For example, for the words "Confexionar", SpanishChecker suggests {conexionar, confinar, confinara, confinare, confinará, confinaré, confinaría, confina}, for vioseguridad SpanishChecker suggests {vio seguridad, vio-seguridad, seguridad, esguardad, resguardad, seguridades, etc.}, and for varato SpanishChecker suggests {va rato, va-rato, verato, grato, arto, parto, urato, verte, varío, barato, etc.}. Among its false positives are Cofeccionistas {confusionistas, confesionistas, confusionista, cancionista, etc} and Cucuteños {cicutinas, cacereños, cicateros, etc}.

The two false negatives are "Crusada" and "Biernes". Finally, STILUS® generates the false positives "Oquendo" and "Cucuteña". In this case, the quality of all methods degrades, but Ar2p-Text is much lower. It is in the first case that Ar2p-Text does not correctly recognize all errors.

#### 5.1.4. Correction of Words Due to the Exchanging of Two Letters

Misspelled words by the exchanging of two letters are corrected by Ar2p-Text in the same way as the previous corrections. Table 9 shows the results for the different systems for a paragraph with 5% and 10% of words with this error. In this case, all systems incorrectly detect more errors, and Ar2p-Text only incorrectly detects three errors. On the other hand, again, all the methods degrade quite a bit with a greater percentage of errors (10%), and Ar2p-Text still yields better results.

**Table 9.** Results of the different corrections for a paragraph with 5% of misspelled words by the exchanging of two letters.

| Methods | % Words with Errors | Detected Errors | False Negatives | False Positives | P | C | F |
|---|---|---|---|---|---|---|---|
| Ar2p-Text | 5% | 8 | 0 | 3 | 72% | 100% | 83% |
| | 10% | 15 | 0 | 5 | 70% | 100% | 80% |
| SpanishChecker® | 5% | 11 | 0 | 6 | 64% | 100% | 78% |
| | 10% | 19 | 0 | 9 | 62% | 100% | 74% |
| STILUS® | 5% | 13 | 0 | 8 | 61% | 100% | 75% |
| | 10% | 22 | 0 | 12 | 57% | 100% | 71% |
| Microsoft Word | 5% | 14 | 0 | 9 | 60% | 100% | 75% |
| | 10% | 23 | 1 | 12 | 57% | 95% | 70% |

#### 5.1.5. Correction of Words Due to Digits or Special Characters

Misspelled words with digits or special characters anywhere in the word are corrected by Ar2p-Text in the same way as the previous corrections. Table 10 shows the results for the different systems for a paragraph with 5% and 10% of words with this error. In this case, only Ar2p-Text can detect the correct errors. The rest do not detect the errors (SpanishChecker®, STILUS®, and Microsoft Word) or they incorrectly detect more errors (STILUS®). In this case, again the quality of Ar2p-Text is not degraded by increasing the percentage of words with errors, which does negatively impact the other methods.

**Table 10.** Results of the different corrections for a paragraph with 5% of misspelled words due to digits or special characters.

| Methods | % Words with Errors | Detected Errors | False Negatives | False Positives | P | C | F |
|---|---|---|---|---|---|---|---|
| Ar2p-Text | 5% | 5 | 0 | 0 | 100% | 100% | 100% |
| | 10% | 9 | 0 | 0 | 100% | 100% | 100% |
| SpanishChecker® | 5% | 1 | 4 | 0 | 100% | 20% | 45% |
| | 10% | 3 | 5 | 0 | 100% | 18% | 42% |
| STILUS® | 5% | 1 | 3 | 1 | 50% | 25% | 33% |
| | 10% | 3 | 5 | 3 | 45% | 22% | 32% |
| Microsoft Word | 5% | 2 | 3 | 0 | 100% | 40% | 57% |
| | 10% | 4 | 5 | 0 | 100% | 35% | 55% |

The ability of Ar2p-Text for special character word recognition is that it finds similarities between special characters and letters like the brain, such as @ ≅ a, E ≅ 3, S ≅ 5, and 9 ≅ q. These symbols are introduced and treated like the other characters of the alphabet.

### 5.2. A Hard Comparison

In the previous tests, we predefined the errors to detect. In order to make a more complete comparison and to show the capacity of Ar2p-text versus other systems, many more texts were taken, analyzing all the misspelled words detected by each solution. The dataset used in the experiment consisted of 32 texts of the digital version of *El País* of 17 May 2001, copied in a Word document, with about 9000 words analyzed, and where 14 spelling errors appear in the texts. There data were used in STILUS® [14]. The results are summarized in Table 11.

**Table 11.** Results obtained for the dataset used in [14] (https://spanishchecker.com/es/, https://www.mystilus.com/Pagina_de_inicio accessed on 15 July 2023).

| Methods | Detected Errors | False Negatives | False Positives | P | C | F |
|---|---|---|---|---|---|---|
| Ar2p-Text | 23 | 0 | 9 | 71.5% | 100% | 83% |
| SpanishChecker® | 56 | 3 | 45 | 57.9% | 95% | 72% |
| STILUS® | 54 | 1 | 41 | 58.1% | 98% | 73% |
| Microsoft Word | 51 | 2 | 39 | 58.6% | 96% | 73% |

The SpanishChecker® is the one that identified the least errors (with 11). Microsoft Word and Stilus® are more effective because they detected 13 and 12 errors. Our approach detected all errors but incorrectly detected nine more errors. The rest of the approaches incorrectly detected many more errors. The main problem with the other systems is that they detected errors that occurred due to the inclusion of digits or special characters, which is not the case (a lot of false positives). In general, their results are far from the results achieved by Ar2p-text.

In addition, we created a dataset with 120 texts from the digital version of *El País* on 21–22 November 2023, with about 30,000 words and with 42 spelling errors manually introduced at random into the text. The results obtained with our model are summarized in Table 12. Our model is still capable of obtaining very good results, by significantly increasing the errors in a much larger sample of texts that contain almost 30% of the words in Spanish (the *Dictionary of the Royal Spanish Academy* has around 100,000 words), making the correct corrections in each case.

**Table 12.** Results obtained by Ar2p-Text for a large dataset.

| Precision | Coverage | F |
|---|---|---|
| 82% | 96% | 86% |

### 5.3. Comparison with an n-Gram Language Model

As a point of reference, the algorithm n-gram language model is considered the standard [28]. This system of spellchecking and auto-correction uses the Web to infer misspellings, a term list, an error model, a language model (LM), and a confidence classifier algorithm. It suggests a candidate for each token in the input text from the term list. The candidates are evaluated using an LM and re-ranked. The classifier is used in each token to determine the confidence of whether a word has been misspelled, and in this case, if it can be autocorrected using the best scoring available. The main contribution this work claims is that it does not need any manual procedure and uses the Web to infer its linguistic knowledge. In this sense, we defined a deep learning architecture for Ar2p for the discovery and selection of features for classification problems [11,29]. Our approach has three phases: The feature analysis phase has two feature-engineering approaches to choose or discover atomic features. The phase of aggregation defines a feature hierarchy from the atomic features. Finally, the classification phase calculates the classification. This phase uses a supervised learning approach, and the rest of the phases combine supervised and unsupervised learning approaches. If the word is misspelled, it uses the clusters of well-written words for its recommendation, and if the word does

not exist there, it searches the web (for example, in repositories such as word reference (https://www.wordreference.com/)) and recommends it and includes it in the group of well-written words. Future work can be the quantitative evaluation of the performances of these approaches.

### *5.4. Comparison with Deep Learning Techniques*

Finally, in this section, we compare our proposal with two deep learning models on two datasets. The first dataset was created by us based on the digital version of *El País* that we used in Section 5.2, with about 30,000 words and with 42 spelling errors manually introduced at random into the text. The second dataset is from short story texts by several Latin American authors stored at https://huggingface.co/datasets/Fernandoefg/cuentos_es (accessed on 1 February 2024). A single dataset was built, and two versions of it were defined. One version had 10% random spelling errors, and another had 25% spelling errors. On the other hand, the deep learning models used were the BART model, which uses a standard machine translation/seq2seq architecture with a bidirectional encoder (such as BERT) and a left-to-right decoder (such as GPT) [30], and the LLaMA (Large Language Model Meta AI) model [31]. Specifically, the BART model links the corrupted text in Spanish with the original text from which it is derived. For that, the BART model uses six layers in the encoder and decoder where each decoder layer performs cross-attention to the final hidden layer of the encoder. The BART model was prepared using the corrupt Spanish texts to optimize the loss of reconstruction between the output of the decoder and the original Spanish text. For this preparation, source texts in Spanish were corrupted with the average spelling errors produced by humans according to Table 3. On the other hand, LLaMA is an autoregressive language model based on the transformer architecture. LLaMA covers a wide range of publicly available texts from various languages such as German, English, Spanish, and French. Of the LLaMA models available in various sizes (7b, 13b, 33b, and 65B parameters), we used the 13b, which is one of the smallest but ideal for our test because it uses minimal computing resources. According to the work [31], LLaMA was trained with the following hyperparameters: $\beta 1 = 0.9$, $\beta 2 = 0.95$, a cosine learning rate schedule such that the final learning rate is equal to 10% of the maximal learning rate, a weight decay of 0.1, and a gradient clipping of 1.0. LLaMA was also subjected to corrupted Spanish texts according to Table 3. Table 13 shows the results in the different cases analyzed.

**Table 13.** Results of the comparison of Ar2p-Text with deep learning techniques.

| Methods | Dataset with 120 Texts from the Digital Version of *El País* | | | Texts of Stories by Latin American Authors (10% Errors) | | | Texts of Stories by Latin American Authors (25% Errors) | | |
|---|---|---|---|---|---|---|---|---|---|
| | **P** | **C** | **F** | **P** | **C** | **F** | **P** | **C** | **F** |
| Ar2p-Text | 82% | 96% | 86% | 91% | 100% | 92% | 85% | 91% | 92% |
| LLaMA | 86% | 98% | 96% | 94% | 100% | 95% | 82% | 89% | 90% |
| BART | 85% | 97% | 91% | 90% | 100% | 94% | 80% | 86% | 88% |

In this comparison, we can see that our algorithm, without a training phase or prior pre-training like deep learning techniques, achieves quality metrics very close to the two techniques. Furthermore, when we have a higher error rate, our technique surpasses the quality of the results obtained with the other metrics. Now, to validate the quality of our results, we carried out a statistical analysis of those results using the non-parametric Friedman test. In particular, the Friedman test was applied to the results, showing statistical differences in performance between the proposed models, according to the order of quality established in Table 13, with $p < 0.001$.

Finally, with respect to the previous techniques, it is good to note that Ar2p-Text is an approach with a lot of explainability because the tree that is built allows for the recognition

process to be described clearly and it is highly scalable due to the recursion and reuse of the patterns at the different levels of the hierarchy.

### 5.5. A Final Discussion about the Characteristics of Ar2p-Text

Our Ar2p-Text pattern recognition model is highly uniform and recursive. It recognizes input patterns through a hierarchy process of self-associated patterns. Ar2p-Text allows for the decomposition of the pattern recognition problem into simpler patterns, allowing for the analysis of the patterns regardless of their level of complexity or nature (a line, a word, a sentence, a paragraph, etc.). Finally, our recognition model is adaptable due to the fact that it learns both new modules (patterns) or the possible changes in the pattern descriptors (such as their relevance weights), which is very useful in the context of a language for the self-learning of words and idiomatic sentences. Unlike other approaches, Ar2p-Text can recognize words with special characters just like the brain, such as @ = a, E = 3, S = 5, 9 = q, m@ma, and p3ra. Also, its novelty is the way it solves the problem. Although several NLP models have achieved very good performances, they have high computational costs.

In this work, different comparisons have been presented with other works that use other techniques, or are based on other assumptions. Because automatic spell-checking systems are currently of great interest in different languages [32–35], a challenge will be to test this approach on them.

## 6. Conclusions and Future Works

In this paper, we have presented an automatic recognition and correction system for misspelled words in Spanish texts. We compared the approach with previous works, demonstrating its superiority. Ar2p-Text detects more different types of orthographic errors and has fewer false positives. Regarding the limitations of the Ar2p-Text recognizer, it requires a supervised definition of the weights assigned to the variables used for the recognition.

The proposed algorithm improves upon the state of the art because our Ar2p-Text pattern recognition model is highly recursive and uniform. It recognizes input patterns through a process of self-association in a hierarchy of patterns. Ar2p-Text allows for the decomposition of the pattern recognition problem into simpler patterns, allowing us to analyze input patterns regardless of their level of complexity or nature (a line, a word, a sentence, a paragraph, etc.). In addition, the Ar2p-text model compared to other pattern recognition methods is easily parallelizable, because its calculations defined in the theorems are simpler and distributed on a hierarchy. Also, the computational cost can be improved with respect to other approaches, with more efficient use of memory, due to a single abstract data structure that can be instanced by various text patterns. Finally, our recognition model is adaptable because it learns both new modules and the possible changes in the pattern descriptors, which is very useful in the context of a language for the self-learning of words and idiomatic sentences.

In future work, the architecture of Ar2p-Text must be extended with unsupervised learning mechanisms, which will allow it to improve its functioning (learning new words). Also, it must be extended for use in other languages. Additionally, Ar2p-Text could simultaneously correct texts written in English and Spanish, which can be interesting in translation tasks. For that, Ar2p-Text must be extended with more recognition modules in different languages (its lexical basis).

Finally, a comparison with other approaches in the domain of NLP is not presented because in this case, we are only interested in the spell-checking (auto-correction) problem. Several NLP approaches exist for machine translation, cognitive dialogue systems, sentiment analysis, text classification, and text summarization, among others, using techniques of natural language understanding and natural language generation present in the state-of-the-art NLP. Thus, future work will be the analysis of the utilization of our approach in these contexts to compare with these techniques.

**Author Contributions:** Conceptualization, E.P. and J.A.; methodology, E.P. and J.A.; formal analysis, E.P. and J.A. resources, A.P.; data curation, E.P.; writing—original draft preparation E.P. and J.A.; writing—review and editing, A.P.; supervision, J.A.; funding acquisition, A.P. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research received no external funding.

**Data Availability Statement:** Data will be available upon request.

**Conflicts of Interest:** The authors declare no conflicts of interest.

## References

1. Ferreira, A.; Hernández, S. Diseño e implementación de un corrector ortográfico dinámico para el sistema tutorial inteligente. *Rev. Signos* **2017**, *50*, 385–407. [CrossRef]
2. Zelasco, J.; Hohendahl, A.; Donayo, J. Estado del arte en. . . Corrección ortográfica automática. *Coordenadas* **2015**, *101*, 10–16.
3. Valdehíta, A. Un corpus de bigramas utilizado como corrector ortográfico y gramatical destinado a hablantes nativos de español. *Rev. Signos* **2016**, *49*, 94–118.
4. Gamallo, P.; Garcia, M. LinguaKit: A multilingual tool for linguistic analysis and information extraction. *Linguamatica* **2017**, *9*, 19–28.
5. da Cunha, I.; Montané, M.; Hysa, L. The arText prototype: An automatic system for writing specialized texts. In *Proceedings European Chapter of the Association for Computational Linguistics*; Association for Computational Linguistics: Stroudsburg, PA, USA, 2017; pp. 57–60.
6. Agirre, E.; Alegria, I.; Arregi, X.; Artola, X.; de Ilarraza, A.D.; Maritxalar, M.; Sarasola, K.; Urkia, M. XUXEN: A spelling checker/corrector for Basque based on Two-Level morphology. In Proceedings of the Third Conference on Applied Natural Language Processing, Trento, Italy, 31 March–3 April 1992; pp. 119–125.
7. Singh, S.; Mahmood, A. The NLP Cookbook: Modern Recipes for Transformer Based Deep Learning Architectures. *IEEE Access* **2021**, *9*, 68675–68702. [CrossRef]
8. Kurzweil, R. How to make mind. *Futurist* **2013**, *47*, 14–17.
9. Puerto, E.; Aguilar, J. Learning algorithm for the recursive pattern recognition model. *Appl. Artif. Intell.* **2016**, *30*, 662–678. [CrossRef]
10. Jiang, K.; Wang, Z.; Yi, P.; Jiang, J. Hierarchical dense recursive network for image super-resolution. *Pattern Recognit.* **2020**, *107*, 107475. [CrossRef]
11. Puerto, E.; Aguilar, J.; Vargas, R.; Reyes, J. An Ar2p Deep Learning Architecture for the Discovery and the Selection of Features. *Neural Process. Lett.* **2019**, *50*, 623–643. [CrossRef]
12. Morales, L.; Aguilar, J.; Garces-Jimenez, A.; Gutiérrez de Mesa, J.; Gómez, J. Advanced Fuzzy-Logic-Based Context-Driven Control for HVAC Management Systems in Buildings. *IEEE Access.* **2020**, *8*, 16111–16126. [CrossRef]
13. Waissman, J.; Sarrate, R.; Escobet, T.; Aguilar, J.; Dahhou, B. Wastewater treatment process supervision by means of a fuzzy automaton model. In Proceedings of the IEEE International Symposium on Intelligent Control, Patras, Greece, 19 July 2000; pp. 163–168.
14. González, V.; González, B.; Muriel, M. STILUS: Sistema de revisión lingüística de textos en castellano. *Proces. Leng. Nat.* **2002**, *29*, 305–306.
15. Napoles, C.; Sakaguchi, K.; Tetreault, J. A Fluency Corpus and Benchmark for Grammatical Error Correction. In Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, Valencia, Spain, 3–7 April 2017; pp. 229–234.
16. Leacock, C.; Chodorow, M.; Gamon, M.; Tetreault, J. Automated grammatical error detection for language learners. In *Synthesis Lectures on Human Language Technologies*, 2nd ed.; Morgan & Claypool Publishers: San Diego, CA, USA, 2014.
17. Li, X.; Du, H.; Zhao, Y.; Lan, Y. Towards Robust Chinese Spelling Check Systems: Multi-round Error Correction with Ensemble Enhancement. In *Lecture Notes in Computer Science*; Springer: Cham, Switzerland, 2023; Volume 14304, pp. 325–336.
18. Cheng, L.; Ben, P.; Qiao, Y. Research on Automatic Error Correction Method in English Writing Based on Deep Neural Network. *Comput. Intell. Neurosci.* **2022**, *2022*, 2709255. [CrossRef]
19. Ma, C.; Hu, M.; Peng, J.; Zheng, C.; Xu, Q. Improving Chinese spell checking with bidirectional LSTMs and confusionset- based decision network. *Neural Comput. Appl.* **2023**, *35*, 15679–15692. [CrossRef]
20. Hládek, D.; Staš, J.; Pleva, M. Survey of Automatic Spelling Correction. *Electronics* **2020**, *9*, 1670. [CrossRef]
21. Andrés, B.; Luján, J.; Robles, R.; Aguilar, J.; Flores, B.; Parrilla, P. Treatment of primary and secondary spontaneous pneumothorax using videothoracoscopy. *Surg. Laparosc. Endosc.* **1998**, *8*, 108–112. [CrossRef] [PubMed]
22. Powers, D. Evaluation: From Precision, Recall and F-Measure to ROC, Informedness, Markedness & Correlation. *J. Mach. Learn. Technol.* **2011**, *2*, 37–63.
23. Cook, V. Teaching Spelling. from. Available online: http://privatewww.essex.ac.uk/~vcook/OBS2O.htm (accessed on 17 May 2023).
24. Spanishchecker. Available online: https://spanishchecker.com/ (accessed on 17 May 2023).

25. Bustamante, F.; Díaz, E. Spelling Error Patterns in Spanish for Word Processing Applications. In Proceedings of the Fifth International Conference on Language Resources and Evaluation, Genoa, Italy, 22–28 May 2006; pp. 93–98.
26. Subhi, S.; Yasin, M. Investigating study of an English spelling errors: A sample of Iraqi students in Malaysia. *Int. J. Educ. Res.* **2015**, *3*, 235–246.
27. Ahmed, I. Different types of spelling errors made by Kurdish EFL learners and their potential causes. *Int. J. Kurd. Stud.* **2017**, *3*, 93–110. [CrossRef]
28. Whitelaw, C.; Hutchinson, B.; Chung, G.; Ellis, G. Using the web for language independent spellchecking and autocorrection. In Proceedings of the Conference on Empirical Methods in Natural Language Processing, Singapore 6–7 August 2009; Volume 2, pp. 890–899.
29. Morales, L.; Ouedraogo, C.; Aguilar, J.; Chassot, C.; Medjiah, S.; Drira, K. Experimental Comparison of the Diagnostic Capabilities of Classification and Clustering Algorithms for the QoS Management in an Autonomic IoT Platform. *Serv. Oriented Comput. Appl.* **2019**, *13*, 199–219. [CrossRef]
30. Lewis, M.; Liu, Y.; Goyal, N.; Ghazvininejad, M.; Mohamed, A.; Levy, O.; Stoyanov, V.; Zettlemoyer, L. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. *CoRR J.* **2019**. Available online: http://arxiv.org/abs/1910.13461 (accessed on 1 February 2024).
31. Touvron, H.; Lavril, T.; Izacard, G.; Martinet, X.; Lachaux, M.A.; Lacroix, T.; Rozière, B.; Lample, G. LLaMA: Open and Efficient Foundation Language Models. *arXiv* **2023**, arXiv:2302.13971.
32. Ridho, L.; Yusnida, L.; Abdul, R.; Deden, W. Improving Spell Checker Performance for Bahasa Indonesias Using Text Preprocessing Techniques with Deep Learning Models. *Ingénierie Syst. D'inf.* **2023**, *28*, 1335–1342.
33. Gueddah, H.; Lachibi, Y. Arabic spellchecking: A depth-filtered composition metric to achieve fully automatic correction. *Int. J. Electr. Comput. Eng.* **2023**, *13*, 5366–5373.
34. Toleu, A.; Tolegen, G.; Mussabayev, R.; Krassovitskiy, A.; Ualiyeva, I. Data-Driven Approach for Spellchecking and Autocorrection. *Symmetry* **2022**, *14*, 2261. [CrossRef]
35. Singh, S.; Singh, S. HINDIA: A deep-learning-based model for spell-checking of Hindi language. *Neural Comput. Appl.* **2021**, *33*, 3825–3840. [CrossRef]