computers

MDPI

Article

Automatic Correction of Arabic Dyslexic Text

Maha M. Alamri [1,2,*] and William J. Teahan [2]

1 School of Computer Science and Information Technology, Albaha University, Albaha 65431, Saudi Arabia
2 School of Computer Science, Bangor University, Bangor LL57 1UT, UK; w.j.teahan@bangor.ac.uk
* Correspondence: maha.alamri@bangor.ac.uk

Received: 31 December 2018; Accepted: 1 February 2019; Published: 21 February 2019

Abstract: This paper proposes an automatic correction system that detects and corrects dyslexic errors in Arabic text. The system uses a language model based on the Prediction by Partial Matching (PPM) text compression scheme that generates possible alternatives for each misspelled word. Furthermore, the generated candidate list is based on edit operations (insertion, deletion, substitution and transposition), and the correct alternative for each misspelled word is chosen on the basis of the compression codelength of the trigram. The system is compared with widely-used Arabic word processing software and the Farasa tool. The system provided good results compared with the other tools, with a recall of 43%, precision 89%, F_1 58% and accuracy 81%.

Keywords: Arabic; corpus; dyslexia; errors; spelling

1. Introduction

Dyslexia is defined as a neurobiological condition characterised by an individual's inability to read, spell, decode text and recognise words accurately or fluently [1]. Dyslexia is considered to arise from a deficiency in the phonological dimension of language. Dyslexia affects about 10% of the population, with around 4% of the world's population being severely dyslexic [2]. The Kuwait Dyslexic Association [3] identified a rate of 6% among Kuwaiti. A later study by Elbeheri et al. [4] reported a percentage of 20% among young Kuwaiti. Around 18% of United Arab Emirates University students were found to have evidence of deficits consistent with dyslexia [5].

Goodwin and Thomson and Washburn et al. [6,7], among others, have found that a key characteristic of dyslexic writing is inelegance, imprecision, poor spacing, missing words, suffix errors, mistaken letter ordering and inconsistent spelling. In the case of Arabic writing, it is worth recognising that short vowels are not considered to be independent graphemes; rather, they are represented as additional diacritical marks. Diacritics of this kind are small symbols that are placed above or below letters, the purpose being to indicate to the reader how the short vowels in certain words should be pronounced [8]; for example, "i + n" indicates the kasratain short vowel; for example, in the word "كرةٍ" [E: "a ball" B: "krpK" R: "krtin"] (the syntax used here is as follows: "Arabic text" [E: "English translation" B: "Buckwalter transliteration" R: "ALA-LCRomanization"]). The literature indicates that dyslexic individuals often write diacritical marks as letters [9–11] as in "كرتن" [B: "krptn" R: "krttn"] instead of "كرةٍ". Additionally, they also tend to write text from left to right instead of right to left used when writing in Arabic [12]. Moreover, most Arabic letters have more than one written form, depending on the letter's place in a word: the beginning, middle or end. These errors are made as a result of confusion of letter-shape [9].

Rello et al. [13] remarked that spelling errors have a significant influence on the way an individual is perceived within a community. Spelling errors and the frequency of such errors are often viewed as being linked to an individual's intelligence. In this context, Graham et al.'s finding that technologies (e.g., spellcheckers) can be used to aid dyslexic individuals in minimising the incidence of spelling

errors in their writing is especially significant [14]. Furthermore, as noted by Hiscox, automatic spelling correction can increase the motivation of dyslexic individuals to write, thus elevating both their quality of life and the quality of their writing [15]. The motivation behind this study, therefore, is to develop an Arabic automatic spelling correction system to help dyslexic writers.

An automatic correction function and spellchecking function may seem similar, but they work differently. A spellchecker flags uncorrected words in the document and provides potential alternatives, called a suggestion list. In contrast, the purpose of the automatic correction function is to correct spelling errors automatically in the text without the need to manually choose the word from a suggestion list. This is also called "autocorrect", "replace as you type" and "text replacement" [16].

Sean Douglas [17], an Internet broadcaster who is dyslexic, highlighted some of the issues relating to spelling correction for a person who is dyslexic: "I generally have two options to deal with spelling mistakes; stop my writing and address every red line as I make a mistake, or wait till I get to the end to go through each spelling mistake one by one. While the built-in spellcheck in programmes like MS Word are pretty comprehensive, the extra time and fatigue caused by using them is far from desirable".

Spellcheckers can help users to self-monitor typos; they can also help users that have the cognitive ability to choose the correct spelling from a suggestion list [18,19]. MacArthur et al. and Montgomery found that the spellchecker is most effective if the correct spelling is provided in the top three of the suggestion list [20,21]. Dyslexic users may face difficulty in choosing the correct word out of the ones suggested, so it is advisable to keep the suggestion list as short as possible [22]. As stated in the Douglas quote above, with spellcheckers, the writer needs to make an extra effort to correct errors. Therefore, the authors of this paper believe that an effective spelling correction tool for dyslexic writers would be one that corrects the text automatically without requiring that the writer choose the right word out of the suggested list.

This paper is organised as follows. Section 2 discusses studies related to Arabic spelling correction and dyslexic spelling correction. Section 3 describes the new Sahah "صحح" [E: "Correct" B: "SHH"] system used here to automatically correct Arabic dyslexic text. Section 3.4.2 presents the experimental results, while Section 4 concludes the study.

2. Related Work

Issues relating to spelling error correction have been investigated by many researchers for several decades. It remains a topic of interest to NLP researchers. The first study was carried out by Damerau in 1964. He developed a rule-based string-matching technique based on substitution, insertion, deletion and transposition [23]. Kernighan et al. [24] used the noisy channel model for spelling correction. Church and Gale [25] used probability scores (word bigram probabilities) and a probabilistic correction process based on the noisy channel model for the purpose of spellchecking. Kukich [26] divided spelling errors into three types: error detection, isolated word correction and context-sensitive correction. In 2000, Brill and Moore published a paper that described a new error model for noisy channel spelling correction based on generic string-to-string edits [27].

There are different approaches that can be used to solve the problem of spelling error detection and correction such as the dictionary-based approach used to detect error words [28], the noisy channel model using n-grams [24,25] and the edit-distance approaches [23] used for error correction. Specific approaches to Arabic spelling correction will be reviewed in the next section.

2.1. Arabic Spelling Correction

Several studies have characterised and classified spelling errors in Arabic [29,30], and some studies have also looked at dyslexic spelling errors in Arabic [9–11].

Several attempts have been made to detect and correct Arabic text with spelling errors using various combinations of approaches such as rule-based and statistical approaches. Mars [31] developed a system for automatic Arabic text correction based on a sequential combination of approaches including lexicon based, rule based and statistical based. The F_1 score obtained through the study

was 67%. Likewise, AlShenaifi et al. [32] used the rule-based, statistical-based and lexicon-based approaches in a cascade fashion, with an F_1 score of 57%. A study conducted by Zerrouki et al. [33] created a list that contained misspelled words with their corrections and also used regular expressions to detect errors and give a single replacement, which resulted in an F1 score of 20%.

Another study by Mubarak and Darwish [34] employed an approach based on two correction models and two punctuation recovery models: a character-level model and a case-specific model, a simple statistical model and a conditional random fields model, respectively. The best result was by using a cascaded approach that involves a character-level model, then case-based correction, resulting in an F_1 score of 63%. Alkanhal et al. [35] used the Damerau–Levenshtein edit distance to generate alternatives for each misspelled word. The selection-based method was then used on the maximum marginal probability via an A* lattice search and n-gram probability estimation to select the most applicable word. The study focused on inserting and removing spaces. For misspelled word detection, the experimental result showed an F_1 score of 98%. In terms of misspelled words' correction, the system achieved an F_1 score of 92%.

Conversely, Zaghouani et al. [36] used regular expression patterns to detect errors by using the Arabic verb forms and affixes and built a rule-based correction method that added linguistic rules using existing lexicons and regular expressions to correct native and non-native text. The system achieved an F_1 score of 67% for native speakers; and an F_1 score of 32% for non-native speakers. Similarly, Nawar and Ragheb performed two studies in 2014 and 2015 [37,38]. The first study developed a rule-based probabilistic system, which achieved a 65% F_1 score on the data [37]. In 2015, Nawar and Ragheb [38] made improvements to a previous statistical rule-based system in which word patterns were used to improve error correction. They also used a statistical system using the syntactic error correction rules; the system achieved an F_1 score of 72% on the Aljazeera articles by native Arabic speakers' dataset and an F_1 score of 35% on the non-native speakers' data. Mubarak et al. [39] employed a case-specific correction approach that addressed particular errors such as substitution and word splits and some errors that are specific to non-native speakers such as gender-number agreement. The best result on non-native speakers' data gave an F_1 score of 27%.

Some studies have adopted the noisy channel model approach. Shaalan et al. [40] detected errors by building a character-based trigram language model in order to classify words as valid and invalid. For correction, they used finite-state automata to propose candidate corrections within a specified edit distance measured by Levenshtein distance from the misspelled word. After choosing candidate corrections, they used the noisy channel model and knowledge-based rules to assign scores to the candidate corrections and choose the best correction independent of the context. Additionally, Noaman et al. [41] used pairs of spelling errors and a corrected form extracted from the Qatar Arabic Language Bank (QALP) to build an error confusion matrix, then used this confusion matrix with the noisy channel model to generate a candidates' list and select a suitable candidate for the erroneous word. The overall system accuracy that was obtained was 85%. On the other hand, a study by Attia et al. [29] attempted to improve three main components: the dictionary, the error model and language model. The way they improved the error model was by analysing error types and creating an edit distance-based re-ranker that analysed the level of noise in different sources to improve the language model. By improving the three main components, they achieved an accuracy rate of 83%.

2.2. Dyslexia Spelling Correction

There are various studies that deal with dyslexic spelling correction in different languages. Pedler [42] developed a program to detect and correct dyslexic real-word spelling errors in English. The program considered words that differed in their parts-of-speech. Decisions for words that have the same parts-of-speech were left for the second stage, which used semantic associations derived from WordNet. Rello et al. [13] used a probabilistic language model, a statistical dependency parser and Google n-grams to detect and correct real-word errors in Spanish in a system called Real Check. For the Arabic language, Alamri and Teahan [43] investigated the possibility of utilising automatic noiseless

channel model encoding-based correction techniques [44] as a form of assistance for dyslexic writers of Arabic, correcting the spelling errors of dyslexic writers in Arabic texts using a PPM model, but were only able to correct single-character errors.

Apart from Alamri and Teahan [43], there is a general lack of research into automatic spelling correction for Arabic dyslexic text. Despite this, there have been a number of studies that investigated assistive technology to help dyslexics to overcome reading and writing difficulties or strengthening skills for dyslexic learners [45–47].

2.3. Commercial Tools

With respect to dyslexia spelling correction software, there is software such as Global AutoCorrect (https://www.lexable.com/education/) and Ghotit (https://www.ghotit.com), but both of these are only available for English. In contrast, we were not able to find any Arabic dyslexia software online. However, there is a toolbar called ATbar (https://www.atbar.org), which is a free cross-browser toolbar that can help in reading, spellchecking and word prediction when writing. The ATbar toolbar can help people with dyslexia and poor vision.

2.4. Differences from Previous Studies

The study presented in this paper is different from previous studies in several aspects. Firstly, this study examines Arabic dyslexia, while previous studies were based on non-native speakers learning Arabic data or news data that were not written by dyslexics. The types of errors handled in the data of previous studies were punctuation errors, grammar errors, real spelling errors and non-word spelling errors; this study specifically examines the spelling errors made by dyslexic writers of Arabic text. Dyslexic writers have a higher level of frequency and severity of misspellings compared to non-dyslexic writers [48]. Many studies have observed that individuals with dyslexia struggle more with words that have difficult structures and spellings than with words that can be easily retained through the process of repetition or which have more simple spellings [49,50]. Examples if dyslexic errors that differ from regular errors that are unique for the Arabic language are as follows: the Arabic language has unique characteristics, such as diacritics, while some letters are written cursively, and the form of Arabic letters changes in accordance with their position within words; that is, whether they are placed at the beginning, in the middle or at the end of the word. For example, the letter 'ت' has three shapes: at the beginning of a word, it is 'تـ'; in the middle it is 'ـتـ'; and at the end. it is 'ـت'. Dyslexic individuals have difficulties in writing the correct form of letters [43]. Furthermore, Abu-Rabia and Taha [9] found that the individuals with dyslexia have difficulties in differentiating between letters with similar forms and different sounds such as the letter 'غ' [R = "gh"] and letter 'ع' [R = "ayn"]. Furthermore, difficulties differentiating between letters with similar sounds and different forms such as in the word "صوت" [E: "sound"], the letter 'ت' sounds like the letter 'ط'. Moreover, they have difficulty with short and long vowels; for example, if the erroneous word was "ثيمر" and the intended word was "ثمار" [E: "fruits"], the author wrote 'ي', instead of the diacritical [R: "ksrh"], and omitted the letter 'ا' [43]. Secondly, the system used in this study is based on various approaches, combining a statistical approach by using the PPM language model and edit operations to generate possible alternatives for each error (a candidates' list). The correct alternative for each misspelled word is then selected automatically using the compression codelength of the sentence. The codelength is the number of bits required to encode the text using the compression algorithm.

The contribution of this paper is that it proposes a new system called Sahah for the automatic spelling correction for dyslexic Arabic text and that it empirically shows how dyslexic errors in Arabic text can be corrected. Experiments were carried out to evaluate the usefulness of the system used. Firstly, the accuracy of the system was evaluated using an Arabic corpus containing errors made by dyslexic writers. Secondly, the results of the system were compared with the results obtained using word processing software and the Farasa (http://qatsdemo.cloudapp.net/farasa/) tool.

3. The Sahah System for the Automatic Spelling Correction of Dyslexic Arabic Text

In order to propose an efficient spelling correction for dyslexic Arabic text, it is necessary to study and categorize the error patterns of dyslexia. Such a study was the focus of the authors' previous work [11]. Following this study, creating a prototype of an automatic spelling correction system called Sahah described in this paper was undertaken. However, the Sahah system is intended to correct dyslexic text automatically by using both a statistical and an edit operation approach, via a hybrid mechanism combining the different, but complementary approaches to correcting the errors.

The workflow of the proposed Sahah system starts with transliteration using the Buckwalter transformation and consists of three stages. The first stage (Stage 1) is a pre-processing stage. The second stage (Stage 2) is a detection and correction stage that contains three further sub-stages: a sub-stage for error detection using a dictionary and two sub-stages for correction. The first sub-stage (2a) uses a statistical model to correct dyslexic errors according to their context based on the dyslexic error classification for Arabic texts (DECA) [11]. The second sub-stage (2b) is for error detection and uses an AraComLexdictionary [29]. The third sub-stage (2c) is based on edit operations to generate a candidate list, then the codelength of the trigram is calculated in order to score the candidate and choose the correct word for the error.

The final stage (Stage 3) is the post-processing stage. The Sahah system ends with the reverse of the Buckwalter transliteration back to Arabic text. More details about each stage are described below, as presented in Figure 1.

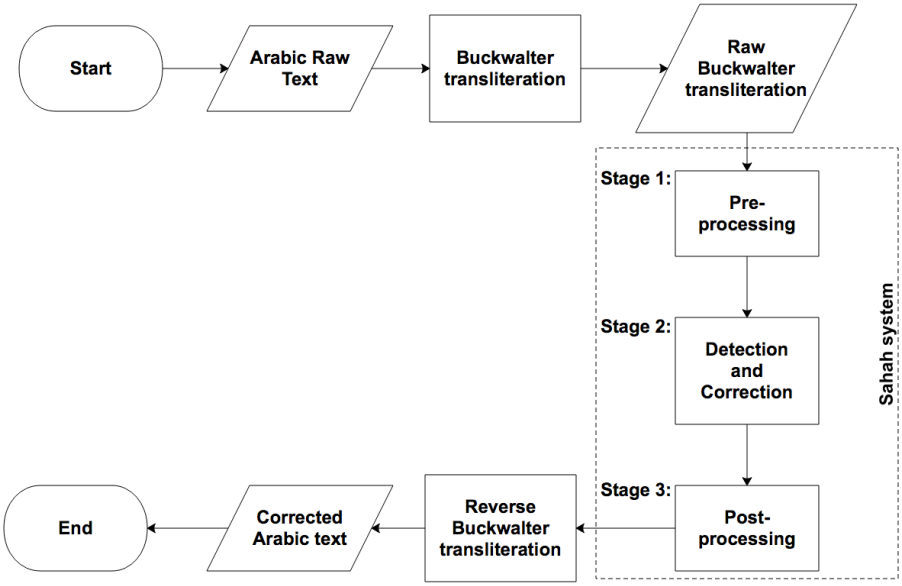

Figure 1. Workflow of the Sahah system.

3.1. Pre-Processing Stage

While preparing the data for the process of error detection and correction, some errors such as tackling of split words and repeated characters were identified as causing 'noise' within the data, complicating the process. Following analysis, the dyslexic errors identified were that sometimes, the dyslexic author divides words into two. This division could be due to the word having a short vowel or due to the pronunciation of the word.

The analysis of dyslexic errors indicated that in some cases, a dyslexic author inserts a space after prefixes or before suffixes; for instance, "أن ها" [B: "<n hA"] to represent the word "أنها" [E: "it" B: "<nhA" R: "annaha"], which is not acceptable, especially in Arabic texts for children. The characters "ها" [B: "hA"] are an Arabic suffix; thus, the way to cover the space insertion was inspired by a light stemming process, which refers to a process of removing prefixes and/or suffixes, without recognising

patterns, dealing with infixes or finding roots [51]. However, instead of splitting the prefix or suffix, we concatenated the two together as part of the pre-processing process; for example, as in the word "أن" [B: "<n"] and the suffix "ها" [B: "hA"] to be "أنها" [E: "it" B: "<nhA" R: "annaha"].

The most common prefixes and suffixes based on the dyslexic corpus analysis were selected as represented in Table 1.

Table 1. The most common prefixes and suffixes based on the dyslexic corpus analysis.

	Arabic	Buckwalter		Arabic	Buckwalter
Prefixes	لل	ll	**Suffixes**	تما	tmA
	ف	f		ها	hA
	ك	k		وا	wA
	ل	l		نا	nA
	ا	A		تا	tA
	كا	kA		تي	ty
	ال	Al		ن	n
	با	bA		ه	h
	إل	<l		ت	t
	فا	fA		ي	y
	بي	by		ة	p

Additionally, the pre-processing stage covered the case where 'ۀ' [B: 'p'] is used in the middle of the word as the grapheme 'ة' [B: 'p'] only appears in the last position in words. Consequently, it is replaced with the grapheme 'ت' [B: 't'], as it is the most likely intended character. For example, the erroneous word "مكۀبة" [B: 'mkpbp'] is replaced by "مكتبة" [B: 'mktbp'].

Hassan et al. [52] removed the incorrect redundant characters from the word. Likewise, our pre-processing stage corrected the redundant characters, but with some modification. The modification is that in the Arabic language, there are some words in which a character can be repeated twice; for instance, "ممتاز" [E: "Excellent" B: "mmtaz"] repeats the character 'م' [B: 'm'] twice. Therefore, characters that were repeated more than twice were reduced to just two repeated characters because no Arabic word contains three consecutive characters.

However, there are some characters that cannot be repeated consecutively twice, which are:

Arabic	اا	آآ	إإ	ءء	ئئ	ىى	ةة
Buckwalter	AA	||	«	"	}}	YY	pp

This case is solved by reducing the repeated characters to one. Table 2 illustrates the way repeated characters were removed.

Table 2. Cases of removing the redundant characters.

Error	Intended Word	After Pre-Processing
المملك [B: "Almmmlk"]	الملك [E: "The king" B: "Almlk"]	المملك [B: "Almmlk"]
سمايه [B: "smAAyh"]	سمائه [E: "His sky" B: "smA}h"]	سمايه [B: "smAyh"]
اللصوره [B: "AllSwrh"]	الصورة [E: "The picture" B: "AlSwrp"]	اللصوره [B: "AllSwrh"]

One example from the experiment is the word "سماايه" [B: "smAAyh"], which contains redundant characters "اا" [B: "AA"]. If the Sahah system does not include the pre-processing stage, Sahah will

change the word "سمايه" [B: "smAAyh"] to "سماوية" [B: "smAwyp"], which is not the intended word. However, by using the pre-processing stage, the Sahah system can correct the error to the intended word, which is "سمائه" [E: "His sky" B: "smA}h" R: "smi"ah"].

Furthermore, it was found that if a step was introduced prior to the error detection and correction stage, it would resolve the issue of the split words and repeated characters, which meant that the accuracy of the detection and correction stage would be enhanced. Therefore, the pre-processing stage included the tackling of split words and repeated characters.

3.2. Error Detection and Correction Stage

This stage was as stated divided into three sub-stages: a sub-stage (2a) that employed the Prediction by Partial Matching (PPM) compression-based language model; a sub-stage (2b) that employed error detection based on a dictionary; and a sub-stage (2c) that employed the edit-operations to generate the candidate list, then score the candidate list based on the codelength. The workflow of Stage 2 is shown in Figure 2; also, more detail about each sub-stage is explained below.

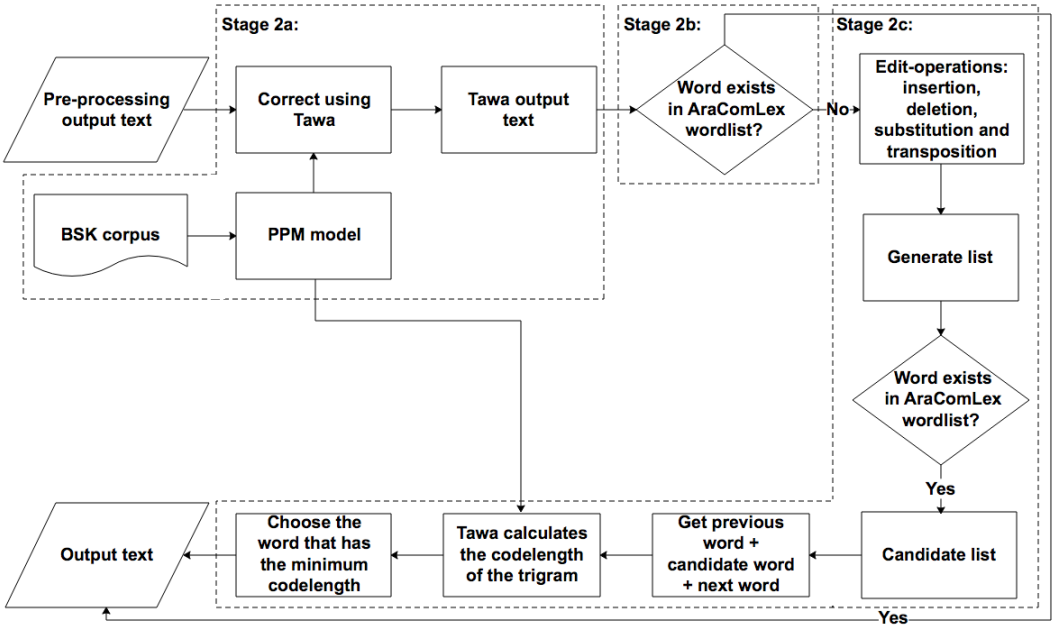

Figure 2. Workflow of the error detection and correction stage of the Sahah system. PPM, Prediction by Partial Matching.

This order of sub-stages arose after discovering the importance of correcting some errors first such as the Hamzaerror type before checking the word with the dictionary and generating the candidate list. For example, the sentence "وتبين انظمه التشيغل للحاسوب" [B: "wtbyn AnZmh Alt$ygl llHAswb"] contains three errors. The second word "انظمه" [B: "AnZmh"] contains two errors; Alif 'A' needs to be replaced with Alif hamza above [B: '<'], and Taa marbuta [B: 'p'] needs to be used instead of Haa [B: 'h']. The third word is "التشيغل" [B: "Alt$ygl"] with the transposition of "غيـ" [B: "gy"] to "يغـ" [B: "yg"].

Table 3 illustrates the result of using the statistical sub-stage (2a) first, then the edit operation correction sub-stage (2c) next, and vice versa.

Table 3. Example of an error that corrects with a different ordering of sub-stages.

	Statistical	**Detection → Edit Operation**
Order 1	وتبين أنظمة التشيغل للحاسوب [B: "wtbyn <nZmp Alt$ygl llHAswb"]	وتبين أنظمة التشغيل للحاسوب [B: "wtbyn <nZmp Alt$gyl llHAswb"]
	Detection → Edit Operation	**Statistical**
Order 2	وتبين نظمه التشغيل للحاسوب [B: "wtbyn nZmh Alt$gyl llHAswb"]	وتبين نظمه التشغيل للحاسوب [B: "wtbyn nZmh Alt$gyl llHAswb"]

As shown in Table 3, if the statistical sub-stage (2a) is utilised first then followed by the detection (2b) and edit operation sub-stage (2c), the system can correct all the errors in the sentence. Contrariwise, if we utilised the detection (2b) and edit operation sub-stage (2c) first then the statistical sub-stage (2a) next, the system can correct the transposition error only.

3.2.1. Sub-Stage 2a: Statistical Stage

This sub-stage is based on using the Prediction by Partial Matching (PPM) language model, which applies an encoding-based correction process. PPM is a lossless text compression method that was designed by Cleary and Witten [53] in 1984. PPM is an adaptive context-sensitive statistical method of compression. A statistical model sequentially processes the symbols (typically characters) that are currently available in the input data [54]. The algorithm essentially uses the previous input data to predict a probability distribution for upcoming symbols. For the correction process, we use an encoding-based noiseless channel model approach as opposed to the decoding-based noisy channel model [44].

As reported by Teahan et al. [55], a fixed order of five is usually the most effective on English text for compression purposes. The variant usually found to be most effective for both compression and correction purposes is PPMD. The experiments conducted for this paper used the version of PPMD implemented by the Tawa Toolkit combined with the noiseless channel model [44].

The formula below can be applied to calculate the probability p of the subsequent symbol φ for PPMD:

$$p(\varphi) = \frac{2c_d(\varphi) - 1}{2T_d} \tag{1}$$

where the current coding order is represented by d, the total number of times that the current context has occurred is shown by T_d and $c_d(\varphi)$ is the total number of times the symbol φ has appeared in the current context [56].

A problem occurs (called the "zero frequency problem") when the current context cannot predict the upcoming symbol. In this case, PPM "escapes" or backs off to a lower order model where the symbol has occurred in the past. If the symbol has never occurred before, then PPM will ultimately escape to what is called an order -1 context, where all symbols are equiprobable. The escape probability e for PPMD is estimated as follows:

$$e = \frac{t_d}{2T_d} \tag{2}$$

where t_d represents the total number of times that a unique character has appeared after the current context.

For example, if the next character in the string "dyslexicornotdyslexic" to be encoded is o, we must make the prediction ic→o using the maximum order, let us say using an order two context. Since the character o has been seen once before in the context ic, then a probability of $\frac{1}{2}$ will be assigned by using Equation (1) since $c = 1$. Correspondingly, 1 bit will be required by the encoder to encode the character because $-\log_2 \frac{1}{2} = 1$.

However, if the subsequent character has not previously been seen in the order two context (i.e., presuming the next letter would be n instead of o, say), it will be necessary to conduct an escape procedure or back off to a lower order. In this case, the escape probability will be $\frac{1}{2}$ (calculated

by Equation (2)), and a lower order of one will then be applied by the model. When this happens, the character n will also not be present after c. As a result, the model will need to encode a further escape (whose probability will also be estimated as $\frac{1}{2}$), and there will be a reduction in the current context to order zero. In this order, the probability that will be applied to encode letter n will be $\frac{1}{42}$. The total cost of predicting this letter is $\frac{1}{2} \times \frac{1}{2} \times \frac{1}{42} = \frac{1}{168}$, which costs around 7.39 bits to encode it ($-\log_2 \frac{1}{168} \approx 7.39$).

The probability $p(S)$ (S is the sequence of length m characters c_i being encoded) is estimated by training a PPM model on Arabic text:

$$p(S) = \prod_{i=1}^{m} p'(c_i | c_{i-5} c_{i-4} c_{i-3} c_{i-2} c_{i-1}) \tag{3}$$

where p' are the probabilities estimated by the order five PPM model.

The codelength can be used to estimate the cross-entropy of the text, which can be calculated according to the following formula:

$$H(S) = -\log_2 p(S) = -\log_2 \prod_{i=1}^{m} p'(c_i | c_{i-5} c_{i-4} c_{i-3} c_{i-2} c_{i-1}) \tag{4}$$

where $H(S)$ is the number of bits required to encode the text.

Improvements in prediction are possible by two mechanisms: full exclusions and Update Exclusions (UE). Full exclusions result in higher order symbols being excluded when an escape has occurred, while Update Exclusions (UE) only update the counts for the higher orders until an order is reached where the symbol has already been encountered [56]. On the other hand, when PPM is applied Without Update Exclusions (WUE), all the counts for all orders of the model are updated. The counts are incremented even if they are already predicted by a higher order context.

To perform the experiment, 10% of the Bangor Dyslexic Arabic Corpus (BDAC), created by Alamri and Teahan [11,43], was used. The size of the BDAC is 28,203 words collected from Saudi Arabian schools, forms and text provided by parents. The texts were written by dyslexics aged between eight and 13 years old. The BDAC contains text written by both male and female students. Ten percent of the BDAC corpus containing different types of errors was used. Firstly, two models were created to enable input of text representative of the training corpus; one model with update exclusion and one without update exclusion. The training corpus consisted of the BACC corpus, a 31,000,000-word corpus called the Bangor Arabic Compression Corpus (BACC) created by Alhawiti [57] for standardising compression experiments on Arabic text. Alkahtani [58] developed a parallel corpus that includes 27,775,663 words in Arabic, based on corpora from Al Hayat articles and the open-source online corpora database and from the King Saud University Corpus of Classical Arabic (KSUCCA), which is part of research attempting to study the meanings of words used in the holy Quran through analysis of their distributional semantics in contemporaneous texts [59]. The above three corpora combined are jointly referred to here as the BSK corpus. Hence, a large text corpus was needed in order to develop a well-estimated language model. This need was met by the BSK corpus.

Then, these models were used in the initial statistical sub-stage (2a). The findings indicated that using update exclusion produced precision 92%, recall 62%, F_1 score 74% and accuracy 82% for detection. Furthermore, precision 80%, recall 22%, F_1 score 35% and accuracy 67% were produced for the correction. With respect to the without update exclusion option, precision was 93%, recall 53%, F_1 score 67% and accuracy 79% for detection. In this case, precision was 86%, recall 26%, F_1 score 40% and accuracy 69% for the correction. As a result of the above experiments, the model without update exclusion was selected for sub-stage (2a). Subsequently, two models with and without update exclusion were created using the BSK to see which one worked better in calculating the codelength (Sub-stage 2c). The results are presented below in Table 4:

Table 4. Improving the model of Sub-stages 2a and 2c. WUE, Without Update Exclusions.

Sub-Stage	Sub-Stage	Detection				Correction			
2a	**2c**	**Rec.**	**Prec.**	**F_1**	**Acc.**	**Rec.**	**Prec.**	**F_1**	**Acc.**
WUE	**WUE**	76	94	84	88	42	90	57	75
WUE	**UE**	76	94	84	88	40	89	56	74

The results of these different experiments revealed that the language model without update exclusion performed better than the model with update exclusion, which is compatible with the findings of Al-kazaz for cryptanalysis [60].

The Tawa toolkit facilitates the definition of transformations in the form of an 'observed→corrected' rule, which denotes the transformation from the observed state to the corrected state when the noiseless channel correction process is applied. The PPM model was applied in order to correct these errors by searching through possible alternative spellings for each character and then using a Viterbi-based algorithm to find the most compressible sequence from these possible alternatives at the character level [61].

The Viterbi algorithm guarantees that the alternative with the best compression will be found by using a trellis-based search: all possible alternative search paths are extended at the same time, and the poorer performing alternatives that lead to the same conditioning context are discarded [44].

In order to correct the erroneous word "احمد" [B: "AHmd"], which contains one error, 'ا' [B: 'A'] is replaced with 'أ' [B: '>']. The correct version is "أحمد" [B: ">Hmd"]. The toolkit generated a possible alternative for each character by using the confusion Table 5. From the confusion, the character 'ا' [B: 'A'] can be ('إ' [B: '<'], or 'أ' [B: '>'], or 'ى' [B: 'Y']). Probabilities for each likely error can be estimated from a large training corpus. Table 6 below is the output of utilising the PPM language model to calculate the codelengths.

Table 5. Confusions list from the DEAC used by sub-stage 2a in the Sahah system.

ظ → ض	أ → ا
ض → ظ	إ → ا
ؤ → و	ى → ا
ٱ → ن	ا → ى
إ → ن	ء → أ
أ → ن	ئ → ي
ً → ن	ت → ة
ٌ → ن	ة → ت
ٍ → ن	ه → ة
ُ → و	ة → ه
ِ → ي	وا → و

Table 6. The codelength of possible alternatives for each character by using the confusions in Table 5 for the erroneous word "احمد".

Confusion	Codelength
'ا' [B: 'A'] → 'إ' [B: '<']	إحمد [B: "<Hmd"] = 70.697 bits
'ا' [B: 'A'] → 'ى' [B: 'Y']	ىحمد [B: "YHmd"] = 75.513 bits
'ا' [B: 'A'] → 'أ' [B: '>']	أحمد [B: ">Hmd"] = 61.424 bits

Thus, the smallest codelength was given to the word "أحمد" [B: ">Hmd"], which is the correct version of the word.

The pre-processing stage and the statistical stage covered many categories from the DEAC, which include the Hamza, almadd, diacritics, differences and form, but it did not include the common errors, which are substitution, deletion, transposition and insertion. Norvig's approach [62] was deemed appropriate for this type of error. However, first, it is necessary to know whether or not the word is an erroneous word; hence, Sub-stage 2b is required.

3.2.2. Sub-Stage 2b: Error Detection

The most direct means of detecting misspelled words is to search for each word in a dictionary and report the words that are not located therein. Based on this principle, an open-source dictionary was used to detect errors in a list containing nine million Arabic words. The words in this dictionary were generated automatically from the AraComLex open-source finite state transducer [29], since it is a free resource that has proven to be effective in previous studies to either correct or detect general spelling errors [40,52] and for the detection of non-native spelling errors [36].

Prior to checking whether a word is in the AraComLex or not, any diacritical marks have to be removed for two reasons. The first reason is that the dyslexic corpus itself does not contain diacritical marks. However, dyslexic individuals have diacritical issues for example when the teacher dictates the word and it contains a diacritical mark (Tanwin). Dyslexic individuals wrote the diacritical Tanwin as character 'ن' [B: 'n'], but did not usually put diacritical marks in their writing. The second reason is that the AraComLex does not contain diacritical marks. Thus, if the input word was not located in the AraComLex dictionary as illustrated in Figure 2, it was considered to contain a spelling error and was passed to the edit operation Sub-stage 2c.

3.2.3. Sub-Stage 2c: Edit Operation

This sub-stage is based on using edit operations, which consist of applying insertion (add a letter), deletion (remove one letter), substitution (change one letter to another) or transposition (swap two adjacent letters) of the misspelled word, and returns a set of all of the edited strings that can be achieved using one or two edit operations. A set of candidate corrections is then generated, including real and non-real words. The candidate list was filtered with reference to an open-source dictionary (AraComLex), commencing with the list of known words for the first edit operation, if any existed, and proceeding to the list of known words for the second edit operation. Once the Sahah system has generated the candidate list, the PPM language model is run to calculate the codelength of the candidate trigram (previous word, candidate word and next word), then returns the candidate word with the lowest candidate trigram codelength. Using the previous example in Section 3.2 above, "وتبين انظمه التشيغل للحاسوب" [B: "wtbyn AnZmh Alt$ygl llHAswb"], following Sub-stage 2a, which corrected the second word "وتبين أنظمة التشيغل للحاسوب" [B: "wtbyn >nZmp Alt$ygl llHAswb"], there was still an error in the third word "التشيغل" [B: "Alt$ygl"], which was transposed under the common category. Table 7 shows the candidate list for the error word "التشيغل" [B: "Alt$ygl"]:

Table 7. Codelengths for different candidate trigrams for a sample correction.

Candidate Word	Candidate Trigram	Codelength (Bits)
"التشغيل" [B: "Alt$gyl"]	"أنظمة التشغيل للحاسوب" [B: ">nZmp Alt$gyl llHAswb"]	100.821
"التشاغل" [B: "Alt$Agl"]	"أنظمة التشاغل للحاسوب" [B: ">nZmp Alt$Agl llHAswb"]	106.453

The lowest codelength is for the candidate word "التشغيل" [B: "Alt$gyl"], which required 100.821 bits to encode the surrounding trigram. Therefore, the Sahah system corrected all errors

in the following sentence: "وتبين أنظمة التشغيل للحاسوب" [E: "and show the operating systems of the computer" B: "wtbyn >nZmp Alt$gyl llHAswb"].

3.3. Post-Processing Stage

The space omission problem was tackled using word segmentation during the post-processing stage.

In order to correct the segmentation of dyslexic errors where spaces had been omitted, the order five character-based PPM model was first trained on the three corpora (BSK). Two segmentations are possible for each character: the character itself and the character followed by a space. In order to find the most probable segmentation sequence that exhibits the best encoding performance, as determined by the PPM language model, the Viterbi-based search algorithm via the noiseless channel model approach was used again to find the best segmentation as measured by the sequence of text with spaces inserted that had the lowest compression codelength. For example, a sample incorrect sequence is "الطائرغرد" [B: "AlTA}rgrd"], while the intended sequence is "الطائر غرد" [E: "the bird is chirping" B: "AlTA}r grd"].

The last step in the Sahah system is the reverse transliteration of the output back into Arabic.

3.4. Evaluation

This section discusses the evaluation methodology and the experiments that were conducted to evaluate the performance of the dyslexic Arabic spelling corrector using the Sahah system that is presented in this paper.

3.4.1. Evaluation Methodology

There are five possible outcomes of the Sahah system. These cases are based on those proposed by Pedler [42]. However, the case where "the error was considered by the program but wrongly accepted as correct" was not applicable in the Sahah system, so it was not adopted. This is because once the Sahah system detected the error, it is either changed to a correct word or an incorrect alternative word. Errors can be dealt with in the first three cases below, and correctly spelt words can be dealt with in the last two cases as below:

Corrected case: The error is detected and replaced with the intended word (Case I).
Incorrect alternative case: The error is detected and replaced with an incorrect alternative (Case II).
Missed case: The error is not detected, and therefore, the system does not correct it (Case III).
Skipped case: The word that is spelt correctly is accepted (Case IV).
False alarm case: A word that is spelt correctly is changed (Case V).

The sentence below illustrate the five possible outcomes as represented by the error→correction form:
The raw text: "Thei were not the onle ones leving on that island."
The gold-standard text: "They were not the only ones living on that land."

Case I: thei→they.
Case II: leving→leaving.
Case III: onle→onle.
Case IV: were→were.
Case V: land→island.

The evaluation methodology used in this study is based on recall, precision, F_1 score and accuracy, which are common natural language processing measures. The gold-standard correction for each spelling error was manually prepared as described by Alamri and Teahan [11].

The two main functions of the Sahah system are error detection and error correction. The evaluation of the Sahah system was therefore separated into two parts: error detection evaluation and error correction evaluation.

Error detection evaluation: The error detection function evaluates whether a word is detected when compared with the gold-standard manual annotation. Recall, precision, F_1 score and accuracy are calculated as follows:

$$Recall\ (Rec.) = \frac{TP}{TP + FN} \tag{5}$$

$$Precision\ (Prec.) = \frac{TP}{TP + FP} \tag{6}$$

$$F_1 score = 2 \times \frac{R \times P}{R + P} \tag{7}$$

$$Accuracy\ (Acc.) = \frac{TP + TN}{TP + TN + FP + FN} \tag{8}$$

where:

True Positive (TP): This implies that a spelling error has successfully been detected.
False Negative (FN): This implies that a spelling error has not been detected.
False Positive (FP): This implies that a correctly-spelled word was detected as being a misspelled word.
True Negative (TN): This implies that a correctly-spelled word was detected as being a correct word.

The total of Case I words and Case II words gives the TP, while the FN is the number of Case III words, FP is the number of the Case V words and TN is the number of Case IV words. Table 8 gives an example of how the recall, precision, F_1 score and accuracy are calculated for a text that contains 60 error words.

Table 8. Sample of error detection evaluation.

Cases	Total
TP = Case I + Case II	50 + 4 = 54
FN = Case III	6
TN = Case IV	20
FP = Case V	23
Rec. = 90%, **Prec.** = 70%, $\mathbf{F_1}$ = 79%, **Acc.** = 72%	

Error correction evaluation: The error correction evaluation is calculated by determining whether a word has been successfully or unsuccessfully corrected based on the gold-standard manual annotation. Recall, precision, F_1 score and accuracy can then also be calculated as follows:

True Positive (TP): This implies that a spelling error was successfully corrected.
False Negative (FN): This implies that a spelling error was not corrected.
False Positive (FP): This implies that a correctly-spelled word was changed.
True Negative (TN): This implies that a correctly-spelled word was not changed.

Using the same example given in Table 8, TP is the number of corrected words (Case I) and TN is the number of skipped words (Case IV), while FN is the total number of incorrect alternative words and missed words (Cases II and III) and FP is the number of false alarm words (Case V). Table 9 gives an example of how recall, precision, F_1 and accuracy is calculated to evaluate the error correction.

Table 9. Sample of error correction evaluation.

Cases	Total
TP = Case I	50
FP = Case V	23
FN = Case II + Case III	6 + 4 = 10
TN = Case IV	20
Rec. = 83%, **Prec.** = 68%, $\mathbf{F_1}$ = 75%, **Acc.** = 68%	

3.4.2. Experimental Results

The Sahah system developed for this study was evaluated in two ways: (i) using a BDAC corpus that consisted of text written by people with dyslexia; and (ii) using a comparison with commonly-used spellcheckers/tools.

(i) Experiment using the BDAC corpus: This experiment used the BDAC corpus (28,203 words). The recall rate, precision and F_1 score for the pre-processing stage and Sub-stage 2a using the PPM language model are presented in Table 10.

Table 10. Detection and correction result after the pre-processing stage and Sub-stage 2a of the Sahah system.

	Rec.	**Prec.**	**F_1**	**Acc.**
Detection	54	93	68	84
Correction	29	88	44	77

When all stages and sub-stages are taken into consideration, the Sahah system achieved a better result (see Table 11).

Table 11. Detection and correction result after all stages and sub-stages of the Sahah system were applied.

	Rec.	**Prec.**	**F_1**	**Acc.**
Detection	75	94	83	91
Correction	43	89	58	81

The F_1 score for correction increased by 14% when the edit operations Sub-stage 2c was used. It is clear that the inclusion of Sub-stages 2b and 2c led to a higher rate of recall, precision, F_1 score and accuracy.

(ii) Experiment using a comparison with commonly-used spellcheckers/tools: For the experimental comparison with commonly-used tools, there are two parts: namely detection comparison and correction comparison.

Detection comparison:

For our comparison, we compared the results of the Sahah system against Microsoft Office 2013 and Ayaspell 3.0 used in OpenOffice because it is a widely-used word processing software. There are a number of previous studies that used Microsoft Office 2013 and Ayaspell 3.0 to evaluate their approach [13,29,31,41]. The results in Table 12 list recall, precision, F_1 score and accuracy by using BDAC corpus evaluation.

Table 12. Detection comparison using the Bangor Dyslexic Arabic Corpus (BDAC) corpus.

Spellchecker Tool	**Rec.**	**Prec.**	**F_1**	**Acc.**
MS word	47	98	64	83
Open Office Ayaspell	53	98	69	85
Sahah	75	94	**83**	91

The assessment of our system's ability to detect errors is based on the F_1 score. Sahah's 83% (shown in bold font) was significantly higher than that for both Ayaspell for OpenOffice (69%) and Microsoft Word (64%). It is also noteworthy that the number of false negatives for Sahah was lower than that of the others systems, while the number of true positives was higher for Sahah.

Correction comparison:

The Sahah system does not show a suggestion list, which means there is no need for human interaction to replace erroneous words. Thus, the spellcheckers investigated above in Table 12 are not compatible with our correction system that was investigated for these experiments. For comparison purposes, the results obtained from this study for the Sahah system in Section 3.4.2 above were compared to the results obtained using the Farasa tool, which is a text processing toolkit for Arabic text.

Farasa comprises a segmentation/tokenisation module, a part-of-speech tagger, an Arabic text diacritizer and spellchecker. Farasa is available online and operates in a similar way to the Sahah system in this study, i.e., the Farasa tool corrects the text automatically without showing a suggestion list. The use of Farasa has been described in two papers [34,39] as detailed in Section 2.1. Both studies produced results with respect to correcting Arabic news, native and non-native text.

The results in terms of recall, precision, F_1 score and accuracy using the BDAC corpus are presented in Table 13.

Table 13. Comparing the Sahah system with the Farasa tool.

Tool	Rec.	Prec.	F_1	Acc.
Farasa	23	84	36	75
Sahah	43	89	**58**	81

When compared with the Farasa tool, the Sahah system achieved higher precision and recall. The number of true positives for Sahah is higher than Farasa, while the number of false negatives is smaller for Sahah.

Although the Sahah system produced good recall, precision and F_1 score rates as discussed above, it could not detect some errors (Type I) or could not correct some errors that were detected (Type II). The errors can be categorised as follows:

- Type I: The Sahah system in some cases could not detect an error if the word used matched with a word in the dictionary. Furthermore, it could not detect errors falling under the word boundary error category, for example the use of "لي عقولهم" [B: "ly Eqwlhm"] instead of "لعقولهم" [E: "To their minds" B: "lyEqwlhm"] where both words are valid. One solution is to check by pair instead of tokens. However, it is worth noting that none of the widely-used word processing software, Microsoft Office 2013 and Ayaspell 3.0 used in OpenOffice or the Farasa tool referred to above can detect this type of error.
- Type II: If more than one letter in the word is deleted or added, it makes the word hard to correct. In such cases, the Sahah system inserted an alternative word. For example, instead of the erroneous word "التر" [B: "Altr"], which is missing three letters, the Sahah system substituted "البر" [B: "Albr"] when the intended word was "التربية" [B: "Altrbyp"]. When the erroneous word contained more than three types of errors, the Sahah system could easily detect the error, but could not correct it, for example "الياملاي" [B: "AlylAmlAy"], which was used instead of "الإملائية" [B: "Al<mlA}yp"], which contained five errors that were detected by the Sahah system, which then exchanged it with the incorrect alternative "النلام لأي" [B: "Al}lAm l>y"].
- Type II: An incorrect alternative occurred when the wrong candidates were chosen on the basis of the codelength of the trigram according to the statistical language model. For example, for "الصوص" [E: "The thieves" B: "AlSwS"], the candidates' list included "للصوص" [E: "The thieves" B: "AllSwS"] (94.727 bits) and "الصوت" [E: "The voice" B: "AlSwt"] (89.462 bits). The candidate list contained the intended word, but the smallest codelength was for [B: "AlSwt"], which is an incorrect alternative in this case.
- Type II: Addition words, deletion words or incorrect synonyms written for a word during dictation time such as "البيت" [E: "Home" B: "Albyet"] instead of "المنزل" [E: "House" B:

"Almnzel"] fall outside the scope of this study as they are not contains errors and very rare in the BDAC corpus.

4. Conclusions

This paper addressed the problem of the automatic correction of errors in Arabic text written by dyslexic writers. It introduced the Sahah system that automatically detects and corrects error words in dyslexic text. The Sahah system has three stages: a pre-processing stage, which corrects split words and repeated characters; the second stage, which uses the character-based PPM language model to identify all possible alternatives for the erroneous words and also uses an edit operation to generate a candidate list with a compression codelength calculation to rank possible candidates for error words; and the post-processing stage that addresses the spaces that had been omitted. It does this by using a character-based PPM method in order to segment the errors caused by dyslexia correctly.

An Arabic corpus containing errors made by dyslexic writers was used to evaluate the performance of the Sahah system presented in this study. The spelling correction used in this study significantly outperformed the Microsoft Word and Ayaspell systems for the detection and the Farasa tool in the correction.

Author Contributions: Methodology, M.M.A., W.J.T.; validation, M.M.A., W.J.T.; formal analysis, M.M.A.; investigation, M.M.A.; resources, M.M.A.; writing, original draft preparation, M.M.A.; writing, review and editing, W.J.T.; supervision, W.J.T.

Funding: The Saudi Arabian government sponsored the PhD scholarship for Maha M. Alamri.

Acknowledgments: We deeply thank the teachers, parents and all the children for providing Arabic texts written by dyslexics.

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

References

1. International Dyslexia Association. *Definition of Dyslexia*; International Dyslexia Association: Baltimore, MD, USA, 2002.
2. Nasen. *Supporting Pupils with Specific Learning Difficulties (Dyslexia) in Secondary Schools*; PDF file; Nasen: Tamworth, UK, 2015.
3. Kuwait Dyslexia Association. *A Survey Study of Dyslexia in Kuwait*; Kuwait Dyslexia Association: Kuwait City, Kuwait, 2002.
4. Elbeheri, G.; Everatt, J.; Al Malki, M. The incidence of dyslexia among young offenders in Kuwait. *Dyslexia* **2009**, *15*, 86–104. [CrossRef] [PubMed]
5. Aboudan, R.; Eapen, V.; Bayshak, M.; Al-Mansouri, M.; Al-Shamsi, M. Dyslexia in the United Arab Emirates university—A study of prevalence in English and Arabic. *Int. J. Engl. Linguist.* **2011**, *1*, 64. [CrossRef]
6. Goodwin, V.; Thomson, B. *Dyslexia Toolkit: A Resource for Students and Their Tutors*; The Open University Press: Milton Keynes, UK, 2007.
7. Washburn, E.K.; Binks-Cantrell, E.S.; Joshi, R.M. What do preservice teachers from the USA and the UK know about dyslexia? *Dyslexia* **2014**, *20*, 1–18. [CrossRef] [PubMed]
8. Al Rowais, F.; Wald, M.; Wills, G. An Arabic framework for dyslexia training tools. In Proceedings of the 1st International Conference on Technology for Helping People with Special Needs (ICTHP-2013), Riyadh, Saudi Arabia, 18–19 February 2013; pp. 63–68.
9. Abu-Rabia, S.; Taha, H. Reading and spelling error analysis of native Arabic dyslexic readers. *Read. Writ.* **2004**, *17*, 651–690. [CrossRef]
10. Burhan, H.; Al-Salahat, M.M.; Al-Shradgeh, M.T.; Alali, W.A. Degree of Common Misspellings of Students with Learning Disabilities. *Int. Interdiscip. J. Educ.* **2014**, *3*, 1–11.
11. Alamri, M.M.; Teahan, W.J. A New Error Annotation for Dyslexic texts in Arabic. In Proceedings of the Third Arabic Natural Language Processing Workshop, Valencia, Spain, 3–4 April 2017; pp. 72–78.
12. Ali, M. *Learning Difficulties Between Skills and Disorders*, 1st ed.; DaR SaFa: Amman, Jordan, 2011.

13. Rello, L.; Ballesteros, M.; Bigham, J.P. A spellchecker for dyslexia. In Proceedings of the 17th International ACM SIGACCESS Conference on Computers & Accessibility, Lisbon, Portugal, 26–28 October 2015; pp. 39–47.
14. Graham, S.; Harris, K.R.; Larsen, L. Prevention and intervention of writing difficulties for students with learning disabilities. *Learn. Disabil. Res. Pract.* **2001**, *16*, 74–84. [CrossRef]
15. Hiscox, L.; Leonavičiūtė, E.; Humby, T. The effects of automatic spelling correction software on understanding and comprehension in compensated dyslexia: Improved recall following dictation. *Dyslexia* **2014**, *20*, 208–224. [CrossRef] [PubMed]
16. Liensberger, C. Context Sensitive Auto-Correction. U.S. Patent 20140067371A1, 22 December 2015.
17. Douglas, S. The Intelligent Spell Checker. *Blog*, 17 December 2015.
18. Berninger, V.W.; Nielsen, K.H.; Abbott, R.D.; Wijsman, E.; Raskind, W. Writing problems in developmental dyslexia: Under-recognized and under-treated. *J. Sch. Psychol.* **2008**, *46*, 1–21. [CrossRef] [PubMed]
19. Berninger, V.W.; Wolf, B.J. *Dyslexia, Dysgraphia, OWL LD, and Dyscalculia*; Brookes Publishing: Baltimore, MD, USA, 2016; p. 227.
20. MacArthur, C.A.; Graham, S.; Haynes, J.B.; DeLaPaz, S. Spelling checkers and students with learning disabilities: Performance comparisons and impact on spelling. *J. Spec. Educ.* **1996**, *30*, 35–57. [CrossRef]
21. Montgomery, D.J.; Karlan, G.R.; Coutinho, M. The effectiveness of word processor spellchecker programs to produce target words for misspellings generated by students with learning disabilities. *J. Spec. Educ. Technol.* **2001**, *16*, 27–42. [CrossRef]
22. Leahy, M. Spelling, Spelling-Checkers and Dyslexia. In Proceedings of the CESI Conference, St. Patrick's College, Dublin, Ireland, January 2002.
23. Damerau, F.J. A technique for computer detection and correction of spelling errors. *Commun. ACM* **1964**, *7*, 171–176. [CrossRef]
24. Kernighan, M.D.; Church, K.W.; Gale, W.A. A spelling correction program based on a noisy channel model. In Proceedings of the 13th Conference on Computational Linguistics, Helsinki, Finland, 20–25 August 1990; Volume 2, pp. 205–210.
25. Church, K.W.; Gale, W.A. Probability scoring for spelling correction. *Stat. Comput.* **1991**, *1*, 93–103. [CrossRef]
26. Kukich, K. Techniques for automatically correcting words in text. *ACM Comput. Surv. (CSUR)* **1992**, *24*, 377–439. [CrossRef]
27. Brill, E.; Moore, R.C. An improved error model for noisy channel spelling correction. In Proceedings of the 38th Annual Meeting on Association for Computational Linguistics, Hong Kong, China, 3–6 October 2000; pp. 286–293.
28. Peterson, J.L. Computer programs for detecting and correcting spelling errors. *Commun. ACM* **1980**, *23*, 676–687. [CrossRef]
29. Attia, M.; Pecina, P.; Samih, Y.; Shaalan, K.; Genabith, J. Improved spelling error detection and correction for Arabic. In Proceedings of the COLING 2012 Posters, Bombay, India, 8–15 December 2012; pp. 103–112.
30. Shaalan, K.F.; Magdy, M.; Fahmy, A. Analysis and feedback of erroneous Arabic verbs. *Nat. Lang. Eng.* **2013**, *21*, 271–323. [CrossRef]
31. Mars, M. Toward a Robust Spell Checker for Arabic Text. In Proceedings of the International Conference on Computational Science and Its Applications, Beijing, China, 4–7 July 2016; pp. 312–322.
32. AlShenaifi, N.; AlNefie, R.; Al-Yahya, M.; Al-Khalifa, H. Arib QALB-2015 Shared Task: A Hybrid Cascade Model for Arabic Spelling Error Detection and Correction. In Proceedings of the Second Workshop on Arabic Natural Language Processing, Beijing, China, 30 July 2015; pp. 127–132.
33. Zerrouki, T.; Alhawiti, K.; Balla, A. Autocorrection of Arabic common errors for large text corpus. In Proceedings of the EMNLP Workshop on Arabic Natural Language Processing (ANLP), Doha, Qatar, 25 October 2014; pp. 127–131.
34. Mubarak, H.; Darwish, K. Automatic correction of Arabic text: A cascaded approach. In Proceedings of the EMNLP 2014 Workshop on Arabic Natural Language Processing (ANLP), Doha, Qatar, 25 October 2014; pp. 132–136.
35. Alkanhal, M.I.; Al-Badrashiny, M.A.; Alghamdi, M.M.; Al-Qabbany, A.O. Automatic stochastic Arabic spelling correction with emphasis on space insertions and deletions. *IEEE Trans. Audio Speech Lang. Process.* **2012**, *20*, 2111–2122. [CrossRef]

36. Zaghouani, W.; Zerrouki, T.; Balla, A. SAHSOH QALB-2015 Shared Task: A Rule-Based Correction Method of Common Arabic Native and Non-Native Speakers Errors. In Proceedings of the Second Workshop on Arabic Natural Language Processing, Beijing, China, 30 July 2015; pp. 155–160.
37. Nawar, M.; Ragheb, M. Fast and robust Arabic error correction system. In Proceedings of the EMNLP 2014 Workshop on Arabic Natural Language Processing (ANLP), Doha, Qatar, 25 October 2014; pp. 143–147.
38. Nawar, M.; Ragheb, M. CUFE QALB-2015 Shared Task: Arabic Error Correction System. In Proceedings of the Second Workshop on Arabic Natural Language Processing, Beijing, China, 30 July 2015; pp. 133–137.
39. Mubarak, H.; Darwish, K.; Abdelali, A. QCRI QALB-2015 Shared Task: Correction of Arabic Text for Native and Non-Native Speakers' Errors. In Proceedings of the Second Workshop on Arabic Natural Language Processing, Beijing, China, 30 July 2015; pp. 150–154.
40. Shaalan, K.F.; Attia, M.; Pecina, P.; Samih, Y.; van Genabith, J. Arabic Word Generation and Modelling for Spell Checking. In Proceedings of the LREC, Istanbul, Turkey, 21–27 May 2012; pp. 719–725.
41. Noaman, H.M.; Sarhan, S.S.; Rashwan, M. Automatic Arabic spelling errors detection and correction based on confusion matrix-noisy channel hybrid system. *Egypt. Comput. Sci. J.* **2016**, *40*, 54–64.
42. Pedler, J. Computer Correction of Real-Word Spelling Errors in Dyslexic Text. Ph.D. Thesis, Birkbeck College, University of London, London, UK, 2007.
43. Alamri, M.M.; Teahan, W.J. Towards a New Arabic Corpus of Dyslexic Texts. In Proceedings of the 2nd Workshop on Arabic Corpora and Processing Tools Theme: Social Media, Portorož, Slovenia, 27 May 2016; pp. 11–15.
44. Teahan, W.J. A Compression-Based Toolkit for Modelling and Processing Natural Language Text. *Information* **2018**, *9*, 294. [CrossRef]
45. Al-Wabil, A.; Meldah, E.; Al-Suwaidan, A.; AlZahrani, A. Designing Educational Games for Children with Specific Learning Difficulties: Insights from Involving Children and Practitioners. In Proceedings of the Fifth International Multi-Conference on Computing in the Global Information Technology (ICCGI), Valencia, Spain, 20–25 September 2010; pp. 195–198.
46. Al-Edaily, A.; Al-Wabil, A.; Al-Ohali, Y. Dyslexia Explorer: A Screening System for Learning Difficulties in the Arabic Language Using Eye Tracking. In *Human Factors in Computing and Informatics*; Springer: Berlin/Heidelberg, Germany, 2013; pp. 831–834.
47. El Kah, A.; Lakhouaja, A. Developing effective educative games for Arabic children primarily dyslexics. In *Education and Information Technologies*; Springer: Berlin/Heidelberg, Germany, 2018; pp. 1–20.
48. Coleman, C.; Gregg, N.; McLain, L.; Bellair, L.W. A comparison of spelling performance across young adults with and without dyslexia. *Assess. Effect. Interv.* **2008**. [CrossRef]
49. Fischer, F.W.; Shankweiler, D.; Liberman, I.Y. Spelling proficiency and sensitivity to word structure. *J. Mem. Lang.* **1985**, *24*, 423–441. [CrossRef]
50. Moats, L.C. Spelling error interpretation: Beyond the phonetic/dysphonetic dichotomy. *Ann. Dyslexia* **1993**, *43*, 174–185. [CrossRef] [PubMed]
51. Larkey, L.S.; Ballesteros, L.; Connell, M.E. Improving stemming for Arabic information retrieval: Light stemming and co-occurrence analysis. In Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Tampere, Finland, 11–15 August 2002; pp. 275–282.
52. Hassan, Y.; Aly, M.; Atiya, A. Arabic spelling correction using supervised learning. *arXiv* **2014**, arXiv:1409.8309.
53. Cleary, J.; Witten, I. Data compression using adaptive coding and partial string matching. *IEEE Trans. Commun.* **1984**, *32*, 396–402. [CrossRef]
54. Teahan, W.J. Text classification and segmentation using minimum cross-entropy. *Content-Based Multimed. Inf. Access* **2000**, *2*, 943–961.
55. Teahan, W.J.; Inglis, S.; Cleary, J.G.; Holmes, G. Correcting English text using PPM models. In Proceedings of the IEEE Data Compression Conference, Snowbird, UT, USA, 30 March–1 April 1998; pp. 289–298.
56. Teahan, W.J. Modelling English Text. Ph.D. Thesis, University of Waikato, Hillcrest, New Zealand, 1998.
57. Alhawiti, K.M. Adaptive Models of Arabic Text. Ph.D. Thesis, The School of Computer Science, Bangor University, Bangor, UK, 2014.
58. Alkahtani, S. Building and Verifying Parallel Corpora Between Arabic and English. Ph.D. Thesis, The School of Computer Science, Bangor University, Bangor, UK, 2015.

59. Alrabiah, M.; Al-Salman, A.; Atwell, E. The design and construction of the 50 million words KSUCCA. In Proceedings of the WACL2 Second Workshop on Arabic Corpus Linguistics, Lancaster, UK, 22–26 July 2013; pp. 5–8.
60. Al-kazaz, N.R.; Irvine, S.A.; Teahan, W.J. An automatic cryptanalysis of simple substitution ciphers using compression. *Inf. Secur. J. Glob. Perspect.* **2018**, *27*, 57–75. [CrossRef]
61. Viterbi, A. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Trans. Inf. Theory* **1967**, *13*, 260–269. [CrossRef]
62. Norvig, P. Natural language corpus data. In *Beautiful Data*; O'Reilly Media: Sebastopol, CA, USA, 2009; pp. 234–239.