*electronics*

MDPI

*Article*

# Lex-Pos Feature-Based Grammar Error Detection System for the English Language

**Nancy Agarwal, Mudasir Ahmad Wani * and Patrick Bours**

Department of Information Security and Communication Technology, Norwegian University of Science and Technology (NTNU), 2815 Gjøvik, Norway; nancy.agarwal@ntnu.no (N.A.); patrick.bours@ntnu.no (P.B.)
* Correspondence: mudasir.a.wani@ntnu.no

Received: 6 September 2020; Accepted: 1 October 2020; Published: 14 October 2020


**Abstract:** This work focuses on designing a grammar detection system that understands both structural and contextual information of sentences for validating whether the English sentences are grammatically correct. Most existing systems model a grammar detector by translating the sentences into sequences of either words appearing in the sentences or syntactic tags holding the grammar knowledge of the sentences. In this paper, we show that both these sequencing approaches have limitations. The former model is over specific, whereas the latter model is over generalized, which in turn affects the performance of the grammar classifier. Therefore, the paper proposes a new sequencing approach that contains both information, linguistic as well as syntactic, of a sentence. We call this sequence a Lex-Pos sequence. The main objective of the paper is to demonstrate that the proposed Lex-Pos sequence has the potential to imbibe the specific nature of the linguistic words (i.e., lexicals) and generic structural characteristics of a sentence via Part-Of-Speech (POS) tags, and so, can lead to a significant improvement in detecting grammar errors. Furthermore, the paper proposes a new vector representation technique, Word Embedding One-Hot Encoding ($W_E O_E$) to transform this Lex-Pos into mathematical values. The paper also introduces a new error induction technique to artificially generate the POS tag specific incorrect sentences for training. The classifier is trained using two corpora of incorrect sentences, one with general errors and another with POS tag specific errors. Long Short-Term Memory (LSTM) neural network architecture has been employed to build the grammar classifier. The study conducts nine experiments to validate the strength of the Lex-Pos sequences. The Lex-Pos -based models are observed as superior in two ways: (1) they give more accurate predictions; and (2) they are more stable as lesser accuracy drops have been recorded from training to testing. To further prove the potential of the proposed Lex-Pos -based model, we compare it with some well known existing studies.

**Keywords:** Natural Language Processing; deep learning; grammar error detection; word embedding

## 1. Introduction

With the advent and continuous advancement in Natural Language Processing (NLP) that aims to enable a machine to understand the human language, the problem of designing a grammar error detector for the natural language is also gaining much attention from researchers [1–4]. The non-native speakers of a language find a hard time in writing grammatically correct sentences. For example, there is a large section of English language learners who need a tool to check if their written content contains grammatical errors [3]. The primary task of a grammar classifier is to predict whether a sentence is grammatically valid or not. The automatic grammar detector can also be applied to grade the writing style of a person by counting the incorrect sentences in their content [1]. Furthermore, a grammar detector can be employed to evaluate the output of Machine Translation (MT) systems which

are designed to produce grammatically correct sentences, by highlighting the translated sentences which contain errors [2].

The language error detection problem is mostly considered as a sequence labeling task where a supervised learning approach is adopted to predict whether the input sequence is grammatically correct or not. Most of the existing studies use one out of two approaches to convert an English sentence into a sequence for the classification task. In the first approach, the sentence is processed as a sequence of words as they appear in the text [1,5]. We refer to this sequence as a lexical sequence. For example, the sentence *"I am reading a book"* will be transformed into the sequence $<I>$ $<am>$ $<reading>$ $<a>$ $<book>$. In the second approach, a sentence is converted into the sequence of tokens which indicate its structural or syntactic information [6,7]. We call these types of sequences syntactic. For example, the syntactic sequence of the same sentence will be $<subject>$ $<helping-verb>$ $<verb>$ $<article>$ $<object>$. This is more like specifying the grammar-domain of words used in a sentence. Researchers use various tools such as dependency parser and Part-Of-Speech (POS) tagger to obtain the structural information of a sentence.

However, we observe that both types of sequences have their inherent limitations. The model trained on lexical sequences is highly specific to the domain of vocabulary of the sentences. Therefore, these models do not generalize well. This implies that, if the sentences in a training set are not enough to cover the large aspect of the English language, the words in test sequences would appear strange to the model. On the other hand, the model trained on syntactic sequences overcomes this limitation by providing the structural characteristics of the sentences, and hence, allow the model to generalize the rules. However, too much generalization is also not good for the model as it often provides insufficient knowledge about the grammar used in a sentence. For example, both words *"a"* and *"an"* are articles but they are used in a different context (e.g., *"an apple"*, *"a banana"*) which cannot be reflected by a syntactic sequence only.

We address this problem by proposing a novel sequence named as Lex-Pos sequence that attempts to capture the specific nature of the lexical sequence and generic nature of the syntactic sequence of a sentence. The structural organization of a sentence in the Lex-Pos format is represented using Part-Of-Speech tags. The required linguistic knowledge is added to the structural knowledge of the sentence to prevent the grammar error classifier from over-generalization.

Since the proposed Lex-Pos sequence contains both lexical tokens and POS-tag tokens, we introduce a new vector representation to represent this sequence in a machine-understandable format. We infused two vector representation techniques viz; word embedding and one-hot encoding to draw the vector of Lex-Pos sequences. We named this representation Word Embedding One-Hot Encoding ($W_EO_E$). In this $W_EO_E$vector representation, the lexical tokens in a sequence are converted into embedding vectors, whereas syntactic tokens are converted into binary vectors.

In order to design the grammar error detector algorithm, a large corpus containing a satisfactory quantity of both correct and incorrect sentences is required. The correct sentences are acquired from the Lang-8 English learner corpus (https://sites.google.com/site/naistlang8corpora/). However, for designing a dataset of grammatically invalid sentences, an artificial error corpus is created by inducing the grammatical errors into the correct sentences of the Lang-8 dataset. Talking about the grammar error types, there are a variety of errors in English language and we distribute them in two categories, viz, syntactic errors and semantic errors. Syntactic errors are caused due to varied reasons for example, a word in a sentence does not spell right (misspelling error), a verb does not conform to the subject (subject-verb agreement error) or a preposition is incorrectly used(preposition error), etc. On the other hand, the sentences with semantic errors are structurally correct but does not make any sense in real life, for example, 'I am eating water', 'we are running a banana', etc. The proposed approach can detect all the syntactic errors in an English sentence and verifies the grammatical structure of a sentence but does not ensure if a sentence is semantically valid, (i.e., if the sentence is meaningful).

Since our target is to train the classifier to differentiate between a valid or invalid Lex-Pos sequence which contain two kinds of tokens, i.e., lexical and POS-tag, two sets of incorrect sentences are designed, one with general errors and another with POS-tag specific errors. The general errors are induced to make the model aware of lexical specific mistakes. The existing error introduction techniques such as *missing verb errors*, *repeated word errors*, *subject-verb agreement errors*, etc. [1] have been used to create different types of such ungrammatical sentences. However, for designing the second set of error corpus, a new error induction method has been implemented that induces POS-tags specific errors in the correct English sentences.

In this paper, the major focus is to show that the proposed Lex-Pos sequence which incorporates both linguistic and structural information of a sentence can markedly enhance the performance of the grammar error detection classifier. The source code for the proposed approach has been made available for the researchers (https://github.com/Machine-Learning-and-Data-Science/Lex-POS-Approach). The main contributions of the work are summarised as follows.

- A new sequence of the English sentence named Lex-Pos is proposed, which tends to infuse the specificity of linguistic and generalization of syntactic characteristics of a sentence;
- A novel vector representation for Lex-Pos sequence of sentences named as Word Embedding One-Hot Encoding ($W_E O_E$) has been presented by combining the word embedding and one-hot encoded sequences;
- The novel error induction methods have been proposed to create negative samples containing POS-tag errors for training;
- The grammar classifier is designed using LSTM deep learning architecture;
- Overall, nine experiments have been conducted on three designed datasets to reveal the potential of Lex-Pos sequences; and
- A comparative study is presented where two replicas of existing grammar-aware systems are designed and experiments are conducted to further demonstrate the strength of Lex-Pos sequences.

The remaining of the paper is structured as follows: Section 2 discusses the literature about grammar detection and correction systems. The proposed Lex-Pos sequence is explained in Section 3 and the datasets and pre-processing are presented in Section 4. In Section 5, different error induction methods are discussed including the newly introduced tag specific error induction. Section 6 presents a novel sequence representation technique that has been used for designing a grammar error detector in this study. The experimental setup and results are discussed in Section 7. Section 8 provides a comparison with existing studies. In Section 9, we have discussed a few limitations of our study, and finally, Section 10 concludes the overall work of Lex-Pos feature-based Grammar Error Detection system for the English Language.

## 2. Background Study

In the grammar detection problem, the sentences are mostly converted into some sequence to obtain a feature set for experiments. Prior works have majorly focused on either considering the sentence itself as a sequence of words or extracting the sequence of tokens which depicts the structure of a sentence. For example, [1] has combined the POS tags of the sentence and the output of the XLE parser (https://ling.sprachwiss.uni-konstanz.de/pages/xle/) to extract the feature set for identifying grammatically ill-formed sentences. The authors also proposed the design of an artificial error corpus for training the model by introducing four types of grammatical mistakes including missing word errors, extra word errors, spelling errors, and agreement errors. The work is further extended in [2], where probabilistic parsing features are incorporated with the POS $n$-grams and XLE-based features to improve the results. In [6], the authors propose a classifier to detect grammatical mistakes in the output produced by Statistical Machine Translation (SMT) systems. The structure of the sentences has

been captured using multi-hot encoding where the word vector represents three types of information: *POS tag*, *morphology* and *dependency relation*.

A large section of researchers has focused on representing the sentences using word embedding vectors. The authors of [8] propose the Grammar Error Corrector (GEC) model using the convolutional encoder-decoder architecture which was trained on word embeddings of the sentences. Another work [3] proposes word embeddings that considers both the grammaticality of the target word and the error patterns. To create incorrect sentences in the corpus, the target word in the sentence has been replaced with a similar but different word that often confuses the learners. For example, replacing 'peace' with 'piece'. Authors in [9] have designed a translation model that assists in understanding the unseen word using its context. The encoder-decoder model which is capable of handling the Out Of Vocabulary (OOV) words has been employed. [10] also utilizes the Convolutional Neural Network (CNN) to build a GEC model. However, the problem is considered as a binary classification rather than a sequence-to-sequence problem. The task of the model is to predict the grammatical correctness of a word based on the context where it has been used in the sentence. The authors also implement word embeddings to represent the sequence of a sentence and substitution error induction method to artificially create the negative samples in the training set.

There are also several studies that attempt to integrate a different level of information of the sentence in the sequence. For example, in [11], word-based sequences represented using word embedding are applied to build a neural GEC model. They also infuse character-level information in the neural network where the word embedding representation of OOV words depends on their character sequences. Study [12] attempts to detect the prepositional mistakes in the sentences by extracting the contextual information of the prepositions. The authors in this study integrated the prepositional words (e.g., *into* or *at*) with the noun or verb phrases to predict the probability of their correct usage in the sentences. Similarly, [13] worked on identifying prepositional errors by combining POS-tagged and parsed information with English words. In our work, we convert the complete sentence into a sequence that contains both structural as well as contextual information. The structural tokens are represented using one-hot encoding and context tokens are represented using word embedding.

Other studies on grammatical error detection focus only on specific errors, such as article errors, adjective errors or preposition errors [7,14,15]. The authors of [7] proposed four error generation methods to introduce article mistakes statistically in English sentences to create negative samples that resemble grammar errors naturally occurring in second language learner texts. A model has been designed to detect and correct article errors. Similarly, the authors in [16] put their efforts into selectively correcting article errors in the sentences. Instead of using all the words in sentences, the model is trained on the sequence of words surrounding the articles only, i.e., $n$ words before and after the article. Article [14] focuses on the mistakes committed by the learner while using adjectives with nouns in sentences. In our study, an attempt is made to target all kinds of errors with special attention to POS-tag specific errors. Therefore, our work utilizes two corpora for negative samples, one with general errors and another with tag specific errors.

## 3. Lex-Pos Sequence

Earlier studies have mainly focused on either lexical knowledge of the sentences such as words appearing in the text or the syntax knowledge of the sentences such as POS tags, as features for training the grammar detection model. In a lexical-based approach, an English sentence can mostly be directly converted into a sequence of words by splitting it with space. Whereas, in a syntactic-based approach, the sentence is first converted into the grammatical structure using tools like dependency parser (http://www.nltk.org/howto/dependency.html) or tagger (https://www.nltk.org/book/ch05.html) and then a sequence is designed by extracting the relevant information.

However, lexical-based models highly depend on the vocabulary of sentences in the training set, therefore, these models are difficult to generalize. For example, a model trained on sentence $S_1$:

*"I have an umbrella"* might fail to understand the grammaticality of the sentence $S_2$: *"I have a cat"* during testing as the words *"a"* and *"cat"* appear new to the model. Therefore, the model trained on the words vocabulary of the sentences is highly vulnerable to categorizing unseen sentences as incorrect.

On the other hand, the learning structure of the sentences allows the model to generalize the rules. For example, the *NLTK pos-tagger* converts both of the above sentences ($S_1$ and $S_2$) into the same sequence of POS tags, i.e., $<PRP>$ $<VBP>$ $<DT>$ $<NN>$ for denoting the *personal pronoun*, *present tense verb*, *determiner* and *noun* respectively. Therefore, the model trained on syntactic features of the sentence, *"I have an umbrella"* can easily predict the structure of the sentence, *"I have a cat"* as correct. However, too much generalization can also increase the false alarms. For example, the *pos-tagger* tool generates the same sequence for the two sentences *"I have a umbrella"* and *"I have an umbrella"*, i.e., $<PRP>$ $<VBP>$ $<DT>$ $<NN>$. Here the articles *a* and *an* are both categorized under same tag $<DT>$.

Therefore, in this paper, we introduced a new sequence, viz, Lex-Pos by combining the specificity level of the lexical approach and generalization of structural characteristics of sentences. In this feature set, we embed the required linguistic knowledge in the POS-tag sequence of the sentence so that the model can learn to generalize the structure of sentence *"I have an umbrella"* to *"I have a cat"*, and at the same time, also distinguish it from the sentence *"I have a umbrella"*.

In order to construct the Lex-Pos sequence, we first need to identify the problematic POS tags which overgeneralize the structure of a sentence. For example, in the sentences, *"I have an umbrella"* and *"I have a umbrella"*, $<DT>$ is the tag which causes the problem. Once we identify these problematic POS tags, we embed additional linguistic knowledge to such tags. For example, the $<DT>$ tag is integrated with two tokens; first the article (i.e., a/an/the) itself, and second the pronouncing alphabet of the word that follows the article as shown in sentences 1, 2 and 3 in Table 1. The *pronouncing* (https://pypi.org/project/pronouncing/) library of python has been used to obtain the pronounced letter of the word.

In case of the *NLTK pos-tagger*, the other tags which were found problematic include $<PRP>$ representing personal pronoun (e.g., *he*, *she*, *I*, *we*, or *you*), $<VBP>$ representing verb such as *am*, *are*, or *have*, and $<IN>$ representing preposition/subordinating conjunction e.g., *in*, *at*, or *on*. All these tags in the syntactic sequence of a sentence are provided with extra linguistic information. Algorithm 1 illustrates the step-wise designing of the Lex-Pos sequence.

**Algorithm 1:** Lex-Pos Sequence.

```
begin
    Input: Sequence;
    Output: Lex_Pos_Seq;
    Calculate pos-tag sequence of Sentence (Pos_Seq);
    Initialize the list of problematic tags (Prob_Tags);
    Initialize empty Lex_Pos sequence (Lex_Pos_Seq);
    foreach Pos_tag in Pos_Seq do
        if Pos_tag in Prob_Tags don then
            Append (Pos_tag + Linguistic information) to Lex_Pos_Seq;
        else
            Append Pos_tag to Lex_Pos_Seq;
        end
    end
end
```

**Table 1.** Examples of Lex-Pos sequences.

| ID | Sentence | Lexical Sequence | POS-tag Sequence | Lex-Pos Sequence | Label |
|---|---|---|---|---|---|
| 1 | I have an umbrella | <i><have><an> <umbrella> | <PRP><VBP><DT> <NN> | <PRP><I><VBP> <have><DT><an><A> <NN> | correct |
| 2 | I have a cat | <i><have><a> <cat> | <PRP><VBP><DT> <NN> | <PRP><I><VBP> <have><DT><a><K> <NN> | correct |
| 3 | I have a umbrella | <i><have><a> <umbrella> | <PRP><VBP><DT> <NN> | <PRP><I><VBP> <have><DT><a><A> <NN> | incorrect |
| 4 | You are here | <you><are><here> | <PRP><VBP><RB> | <PRP><you><VBP> <are><RB> | correct |
| 5 | I are here | <i><are><here> | <PRP><VBP><RB> | <PRP><I><VBP> <are><RB> | incorrect |
| 6 | I am here | <i><am><here> | <PRP><VBP><RB> | <PRP><I><VBP> <am><RB> | correct |
| 7 | I am sitting on the table | <i><am><sitting> <on><the><table> | <PRP><VBP> <VBG><IN><DT> <NN> | <PRP><I><VBP> <am><VBG><sitting> <IN><on><table><DT> <NN> | correct |

Table 1 shows a few instances of Lex-Pos sequences. It can be seen in Table 1 that the two tags $<PRP>$ and $<VBP>$ are appended with the information of personal pronoun and helping verb in sentences 4, 5 and 6. In the case of $<IN>$ tag, three lexical tokens, namely, preposition, word preceding the preposition and word following the preposition are appended. However, if the preceded or followed word comprises some $<DT>$ tag words such as *the*, or *some*, then these words are ignored and the next word in the sequence is appended as shown for the last sentence in Table 1.

## 4. Datasets and Pre-Processing

Training of grammar classifiers requires both correct and incorrect sentences in a dataset. We used the Lang-8 Corpus of the Learner English dataset as grammatically valid English sentences for our experiments. The dataset contains over 5 million sentences with the length of the sentences ranging from 1–80 words. We selected sentences with a length of less than 15 words in order to reduce the variation in the length of the sentences during the training of the model. Finally, we obtained around 1 million correct sentences. The incorrect sentences are obtained from the correct corpus by writing error induction programs which are explained in detail in Section 5.

Although the sentences in the Lang-8 corpus are already verified as grammatically correct, we performed a few pre-processing functions so as to design an efficient dataset for training. First, we converted the sentence into lower case. Then, we replaced the contracted form of auxiliaries in the sentences with their long-form (e.g., *"I'm not"* $\rightarrow$ *"I am not"*). Also, numbers in the sentences are replaced with the keyword *digit* to reduce variation (e.g., *"I am 16 years old"* $\rightarrow$ *"I am digit years old"*). However, we did not remove any punctuation marks from the sentences as they hold significant knowledge of the structure of the sentences. The python libraries, *nltk* (https://www.nltk.org/) and *re* (https://docs.python.org/3/library/re.html), were used to pre-process the sentences.

## 5. Error Induction Methods

In this section, we describe the procedure used to generate an artificial error corpus from the Lang-8 dataset which has been made available for the researchers (https://github.com/Machine-Learning-and-Data-Science/Lex-POS-Approach). Our target is to train the machine learning based model to differentiate the correct sequence of the sentence from the wrong ones. Various researchers have used the notion of breeding artificial error data for training the grammar detector model [1,2]. A sentence can be grammatically invalid due to varied reasons, for example, a word in a sentence does not spell right, a verb does not conform to the subject, or a preposition is incorrectly used. Training requires a large set of grammatically incorrect sentences containing enough samples for each kind of error, which is hard to collect in the sentences produced by native language speakers or writers. However, the dataset of grammatically incorrect sentences with a sufficient number of sentences can be created by performing certain transformations in the grammatically correct sentences (e.g., *inserting*, *replacing*, *repeating* or *deleting* words from the correct sentences). While inserting the errors, proper linguistic knowledge is required in order to ensure that the sentence produced by the script is grammatically unacceptable. For example, consider the sentence *"she bought two fresh apples"*, and only deleting the word *"fresh"* from the sentence does not make the sentence incorrect.

In this work, two types of error induction methods, namely General Error Induction and Tag-specific Error Induction, are employed. Sentences with general errors assist the detector in mainly learning the lexical mistakes and tag-based errors helps in making the model learn about POS-tag related mistakes. Both error induction methods are discussed in the following sub-sections.

### *5.1. General Error Induction Methods*

General errors contain those methods which have been mostly adopted by the earlier studies for creating incorrect sentences. In our dataset, we introduce 5 types of errors, i.e., *misspelled error*, *repeated word error*, *subject-verb agreement error*, *word order error*, and *missing verb error*. Table 2 provides a brief description of the list of these general errors.

In order to ensure that the sentences created by the error induction procedure are grammatically invalid, a few things were taken into consideration. First, we ensure that we do not misspell the words which are proper nouns. Nouns are something for which a dictionary is unlimited, such as the name of a person. For example, *"Alice is having tea"*, *"Aliceee is having tea"*, and *"Ali is having tea"* are all correct sentences. Therefore, we avoid misspelling proper nouns while creating negative references.

**Table 2.** Examples of General Errors.

| Type | Error Induction Procedure | Correct Sentence | Negative Sample |
|---|---|---|---|
| Misspelled | Misspell an appropriate word in a sentence | Boys are playing outside. | Boys are playing **outsde**. |
| Repeated | Duplicate an appropriate word in a sentence | Boys are playing outside. | Boys are **are** playing outside. |
| Subject-Verb Agreement | Replace the verb with a verb disagreeing with the subject | Boys are playing outside. | Boys **is** playing outside. |
| Word Order | Swap the position of two appropriate words in a sentence | Boys are playing outside. | Boys **playing are** outside. |
| Missing Verb | Delete a verb from the sentence | Boys are playing outside. | Boys playing outside. |

For creating sentences with subject-verb agreement errors, we replace the singular verbs with the plural verbs or the other way round to create incorrect sentences. For example *are* is replaced with *is* or *"has"* is replaced with *"have"*.

While generating the repeated errors, we avoided repeating words like *very* or *so*, as a repetition of such words does not make a sentence grammatically incorrect. For example, both sentences *"I like you very much"* and *"I like you very, very much"* are treated as correct in grammar.

While creating word-order errors, we avoid swapping helping-verb with its subject if the sentence is interrogative as both sentences *"am I working"* and *"I am working"* are correct in the English language.

Table 3 provides the distribution of errors in the incorrect dataset. It can be noted from the table that the number of sub-verb agreement and missing verb errors are less when compared to other types of errors as these errors are limited to verbs in the sentences, whereas the domain of other types of errors is not limited to verbs only. It should be noticed that multiple errors can be introduced in a single sentence, i.e., a sentence can have more than one kind of error. Even though the total number of errors created is 85,092, the total number of negative samples produced by the general error method is only 62,899.

**Table 3.** Distribution of General Errors.

| Type of Error | #Negative Samples |
|---|---|
| Misspelled | 31,067 |
| Repeated | 24,648 |
| Subject-Verb Agreement | 5756 |
| Word Order | 16,158 |
| Missing Verb | 7463 |
| Total | 85,092 |

### *5.2. Tag-Specific Error Induction Methods*

As discussed in the earlier sections, $< DT >$, $< PRP >$, $< VBP >$ and $< IN >$ are the tags which provide insufficient knowledge about the structure of a sentence, therefore, these tags must be provided with some additional linguistic knowledge for training a machine learning-based model to differentiate between grammatical correct and incorrect sentences. While creating the tag-specific

errors, we introduce errors particularly for these problematic tags to obtain enough negative examples for assisting the model to learn such errors in the sequence structure. Table 4 provides examples of tag specific instances of the sentences.

The total number of negative samples produced by the tag-specific error method is 50,015, which is less than the total number of errors in Table 5 for the same reason as for the general errors. Table 5 provides the distribution of errors in the incorrect dataset.

**Table 4.** Examples of Tag-specific Errors.

| Type | Correct Sentence | Negative Sample |
|---|---|---|
| $<DT>$ error | I am eating **an** apple. | I am eating **a** apple. |
| $<PRP>$ error | **He** is coming tomorrow. | **You** is coming tomorrow. |
| $<VBP>$ error | I **am** reading a book. | I **are** reading a book. |
| $<IN>$ error | I am sitting **on** the table. | I am sitting **to** the table. |

**Table 5.** Distribution of Tag-specific Errors.

| Type of Error | #Negative Samples |
|---|---|
| $<DT>$ error | 18,169 |
| $<PRP>$ error | 15,008 |
| $<VBP>$ error | 20,295 |
| $<IN>$ error | 9872 |
| Total | 63,344 |

## 6. Feature Representation

In the proposed work, we convert every sentence (correct and incorrect) in the dataset to the Lex-Pos sequence as discussed in the earlier section. However, for training the machine learning-based model, the Lex-Pos sequence needs to be converted into some machine-understandable (mathematical) form. Researchers have employed a variety of ways to represent a linguistic sequence into useful features, e.g., Bag of Words (BoW), $N$-grams, TF-IDF, word embedding, and one-hot encoding [17], etc. The approaches such as Bag of Words (BoW), $N$-grams and TF-IDF rely on the set of tokens and their frequency in the dataset and are therefore insufficient to capture the exact structure of a sentence.

However, in one-hot encoding representation, each word in the vocabulary is assigned a unique binary vector. Therefore, in this encoding, all the distinct words receive distinct representation and the length of the one-hot vector is decided by the number of words in the vocabulary. Usually, the size of POS-tags vocabulary is limited, and hence employing one-hot encoding is a good choice to represent the POS-tag sequences. But the one-hot vector to represent an English word seems an inefficient approach as the length of the binary vector could be extremely long due to the large size of English vocabulary.

Word embedding is another feature representation technique in which every distinct word in the vocabulary is mapped to a numeric vector so that semantically similar words share similar representations in the vector space. One good advantage of using word embedding is that the words can be represented in a much lower dimension than the one-hot encoding. Therefore, word embedding seems an optimal choice to represent the English tokens.

Earlier studies have represented the sequences using either the one-hot vector or embedded-word vector. Since the proposed Lex-Pos sequence consists of both POS tags and English words, we present the feature representation that combines both techniques, named as $W_E O_E$. In this technique, we first maintain a list called *tag-list* which contains all the POS-tag tokens generated by *NLTK pos-tagger* along with their index values. The *tag-list* assists in identifying the tokens in the Lex-Pos sequence which need to be represented in one-hot encoded form. We also appended pronouncing alphabets of the words to

the *tag-list* for adding the linguistic information to the $<DT>$ tag as mentioned in Section 3. In order to obtain the word embedding vectors of the English tokens in the Lex-Pos sequence, Google's pre-trained *Word2Vec* model has been utilized. The model includes 300-dimensional word-vectors for around 3 million English words. The *Gensim* library (https://pypi.org/project/gensim/) of python has been used to extract the embeddings from the *Word2Vec* model.

The Algorithm 2 explains the procedure explains the procedure of changing the Lex-Pos sequence into the Word Embedding and one-hot Encoding ($W_EO_E$) representation. Three arguments are passed to the algorithm as input: (1) Lex-Pos sequence; (2) *tag-list*; and (3) *Word2Vec* model. The *sentVector* variable is initialized to store the vector representation of each token in the sequence. Also, every token of the Lex-Pos sequence is initialized with a fixed length ($n$) vector having all zero entries. In our case, the size of the *tag-list* is less than the size of the embedding vectors of the *Word2Vec* model. Therefore, the value of $n$ ranges from *min* to *max*, where the minimum value is the number of tokens in the *tag-list*, and the maximum value is the length of the embedded vector of the *Word2Vec* model.

**Algorithm 2:** $W_EO_E$ representation of a Lex-Pos Sequence.

```
begin
    Input: Lex_Pos_Seq, POS_tag_list, word2vec;
    Output: Word Embedding and one-hot Encoding Vector W_EO_E_Sent_Vec;
    W_EO_E_Sent'_Vec = [];
    foreach token in Lex_Pos_Seq do
        Initialize a zero vector with n length (W_EO_E_token_Vec);
        if token in POS_tag_list then
            W_EO_E_token_Vec[POS_tag_list[token]] = 1;
        else
            if token in word2vec_model then
                W_EO_E_token_Vec = word2vec(token)[:n];
        end
        Append W_EO_E_token_Vec to W_EO_E_Sent_Vec;
    end
end
```

In order to generate the $W_EO_E$feature vector representation, every token of the Lex-Pos sequence is first passed through a filter to check if this token exists in *tag-list*. If found, the zero vector of the token is replaced with the respective binary vector, otherwise the token is searched in the *Word2Vec* model. If a token is found in the model, the zero vector is replaced with its embedded representation. The token is considered as unknown if it is not found in either of the lists. Finally, the vector values of a token are appended to the *sentVector*.

## 7. Experiments and Results

In Section 4, we discussed the corpus of grammatically correct sentences, and in Section 5, we presented two types of error induction methods for creating two different corpora of negative samples, i.e., one with incorrect sentences containing general errors and another with incorrect sentences having POS-tag specific errors. All three corpora are utilized to create three datasets in the following manner.

**Dataset 1:** Correct sentences + incorrect sentences with general errors;
**Dataset 2:** Correct sentences + incorrect sentences with tag-specific errors;
**Dataset 3:** Correct sentences + incorrect sentences with both general and tag-specific errors;

Earlier studies on grammar classifiers have employed either lexical sequences or POS-tag sequences of a sentence for grammar classification. This work presents a Lex-Pos sequence which tends

to imbibe the specificity quality of lexical sequences and generalization trait of POS-tag sequences. Therefore, we compare the efficiency of a classifier trained on Lex-Pos sequences with the classifiers modeled using lexical and POS-tag sequences. We evaluate the performance of the proposed work on detecting grammatical errors using the 3 datasets described above. Sentences of each dataset are converted into the three types of sequences, lexical sequence, POS-tag sequence and Lex-Pos sequence as shown earlier in Table 1. There are a total of 3 datasets and 3 types of sequences to represent a sentence in each dataset, thus, in total, nine experiments are conducted for comparing the performances of the proposed grammar detector model.

In Section 6, we discussed the one-hot encoding and word embedding representations to denote a linguistic sequence in numeric form. In the experiments, we represent lexical sequences of sentences using a word embedding vector as it allows to represent a word in lower dimensions. The POS-tag sequences are represented using one-hot encoded vectors as the list of POS-tags is very limited. The Lex-Pos sequences are represented using the $W_E O_E$-feature vector. Before training the model, all three datasets were balanced by randomly removing the extra instances from the dataset where it was required. The final size of each dataset used in the experiments is shown in Table 6.

**Table 6.** Statistics of Corpuses.

| Dataset | #Positive Samples | #Negative Samples | #Total Samples |
|---|---|---|---|
| 1 | 60,000 | 60,000 (general errors) | 120,000 |
| 2 | 50,000 | 50,000 (specific errors) | 100,000 |
| 3 | 60,000 | 30,000 (general) + 30,000 (specific) | 120,000 |

The Long Short-Term Memory (LSTM) neural network architecture has been employed to build a classifier. An LSTM network is a variant of a Recurrent Neural Network (RNN) which is extensively used in solving NLP problems as they are capable of learning the structure of sequential data. All the datasets are split in the ratio of 80:20 for training and testing respectively. The *Keras* framework has been used for implementation. In all of the nine experiments, we have used *sparse-categorical-crossentropy* as a loss function and *adam* as an optimizer with a batch size of 2000. The outmost layer of the network is a *dense* layer with 2 nodes and a *softmax activation* function. Since we are using balanced datasets, the accuracy metric has been evaluated to assess the performance of the models.

The results shown in Tables 7 and 8 are for the grammar classifiers which were trained on lexical and POS-tag sequences of the sentences respectively. If we compare the vocabulary size (i.e., unique number of tokens in the training sets) of datasets of both sequences, it can be seen that the vocabulary size of pos-tag sequences (38 or 39) is much smaller than the lexical ones (15,796 to 20,725). This indicates the generalization capability of keeping the structure of the sentences in its syntactic form. The accuracy obtained on the testing sets of lexical sequences is 80%, 96% and 80% for dataset 1, 2 and 3 respectively. On the other hand, accuracy values obtained on the testing sets of POS-tag sequences are 79%, 75% and 73%, which are significantly lower than the accuracy recorded for the lexical-based classifier. This indicates that the classifier performs better with lexical sequences than the POS-tag sequences in all the datasets.

**Table 7.** Lexical Sequence—Word Embedding Representation.

| | | Training | | Testing | |
|---|---|---|---|---|---|
| Dataset | Voc. Size | Loss | Accuracy | Loss | Accuracy |
| 1 | 20,725 | 0.30 | 0.87 | 0.46 | 0.80 |
| 2 | 15,796 | 0.55 | 0.98 | 0.19 | 0.96 |
| 3 | 19,777 | 0.29 | 0.87 | 0.48 | 0.80 |

**Table 8.** POS-tag Sequence—One-hot Encoding Representation.

| | | Training | | Testing | |
|---|---|---|---|---|---|
| **Dataset** | **Voc. Size** | **Loss** | **Accuracy** | **Loss** | **Accuracy** |
| 1 | 39 | 0.38 | 0.83 | 0.44 | 0.79 |
| 2 | 38 | 0.41 | 0.79 | 0.49 | 0.75 |
| 3 | 39 | 0.45 | 0.77 | 0.52 | 0.73 |

Also, while creating the specific-tag errors, we mention that these are basically those errors for which the POS-tag classifier finds it difficult to discriminate. The statement is also reflected in the results of Table 8, where it can be seen that the pos-tag classifier achieves better accuracy in dataset 1, which contains general errors in negative samples (79%) than dataset 2, which contains specific errors in negative samples (75%).

However, it can also be noticed in the results shown in Tables 7 and 8 that the accuracy-drops from training to testing sets are higher for lexical sequences by significant margins. For example, the accuracy obtained in the training set for dataset 3 of lexical sequences (87%) is reduced to 80% in the testing set, i.e., a 7% decrement in the accuracy. The value of loss also increases from 0.29 (training) to 0.48 (testing), i.e., a 19% increase in the loss value. On the other hand, while evaluating the performances of the POS-tag based classifiers on the training and testing sets of dataset 3, there is 4% reduction in the accuracy value and 7% increment in the loss values. This indicates that although the POS-based model is not as accurate as the lexical-based model, it is more stable than the lexical-based model.

The objective of this paper is to combine the effectiveness and stability characteristics into one model by converting English sentences into the Lex-Pos sequences. Table 9 shows the results of the classifiers trained on the Lex-Pos sequences of the sentences with $W_E O_E$feature representation. It can be seen that the vocabulary size of Lex-Pos sequences (1026 to 2122) in the training set lies between the vocabulary size of lexical (15,796 to 20,725) and POS-tag sequences (38 to 39). This indicates that the Lex-Pos sequences tend to maintain a balance between the generalization and specialization of the two sequence types. It is evident from the results that the Lex-Pos classifier outperforms both lexical and POS-tag based classifiers in all the three datasets. The accuracies obtained by the Lex-Pos models on datasets 1, 2 and 3 are 84%, 97% and 87% respectively.

The results also put the Lex-Pos sequences on top from the aspect of stability as they obtain lower values for both metrics, *increment in the loss* and *decrement in the accuracy* while deploying the classifiers from the training to the testing environment. For example, in dataset 3, the loss values of the Lex-Pos system for training and testing are 32% and 26% respectively (see Table 9), thereby, a total of 6% increment in the loss. The value is significantly lesser than the loss increment values for lexical (19%, see Table 7) and POS-tag systems (7%, see Table 8). A similar pattern is observed for the accuracy drop. In dataset 3, the value of accuracy decreases from 89% in training to 87% in testing in the case of Lex-Pos , a total of 2% drop in the accuracy. This accuracy drop of 2% is also markedly lower than the values obtained by lexical (7%) and POS-tag (4%) classifiers.

**Table 9.** Lex-Pos Sequence—$W_E O_E$ Representation.

| | | Training | | Testing | |
|---|---|---|---|---|---|
| **Dataset** | **Voc. Size** | **Loss** | **Accuracy** | **Loss** | **Accuracy** |
| 1 | 2122 | 0.29 | 0.88 | 0.39 | 0.84 |
| 2 | 1026 | 0.26 | 0.99 | 0.11 | 0.97 |
| 3 | 1918 | 0.26 | 0.89 | 0.32 | 0.87 |

## 8. Comparative Study

In this section, we compare the proposed work with two well known existing studies in order to further demonstrate the potential of Lex-Pos sequences. The experiment results show that the Lex-Pos

sequences represented using $W_E O_E$-feature vectors have more potential to capture the grammatical structure of English sentences than the POS-tag sequences and lexical sequences, and so, are more suitable for designing the grammar aware systems. We compare our work with two other existing studies, [5,16]. In each comparison, we replicate the models proposed by the authors in their work and conduct two sets of experiments. In the first experimental setup, we feed the sequence mentioned by authors in [5,16] as input to the implemented model, and in the another setup, we feed the Lex-Pos sequence as input to the implemented model to see and compare the results.

In [5], the authors designed an essay scoring system to evaluate the writing skills. The objective of the system is to assign a rating (i.e., 0–5) to an English essay that reflects the quality of its content based on various parameters including grammatical correctness. The authors experimented with several deep learning models such as CNN, RNN, LSTM and LSTM+CNN and observed that the LSTM-based system outperformed the others. For comparison, we implemented a similar LSTM-based system which was claimed by the authors as the best. The values of the hyper-parameters are set same as by the authors. Table 10 lists the settings of these hyper-parameters used for training the model, referred to here as Essay model.

**Table 10.** Hyper-parameters settings for Author Model [5].

| Parameters | LSTM Nodes | Dropout | Epoch | Optimizer | Learning Rate |
|---|---|---|---|---|---|
| Values | 300 | 0.5 | 50 | RMSProp | 0.001 |

The authors evaluated the quality of English essays, including short ones, on a scale of 0 to 5. However, here, we evaluate the quality of English sentences based on its grammatical structure on the scale of 0 or 1 where 0 score refers to a correct and 1 score refers to an incorrect sentence. The three datasets discussed in Section 7 have been used for training and testing the Essay model. For comparison, two experimental setups have been established. In the first round of experiments, the Essay model takes the word embeddings of the lexical sequences of sentences (as mentioned by the authors in their work) as input. In the other round, we provided our proposed $W_E O_E$-feature vectors of Lex-Pos sequences as input features to the model.

The results obtained from the two rounds of experiments are shown in Tables 11 and 12 respectively. It can be seen that on the training set, the author methodology (lexical sequence and LSTM model) with the accuracies 0.89, 0.98 and 0.91 shows slightly better performance than the same model trained using Lex-Pos sequences on datasets 1, 2 and 3 respectively. However, while testing the Lex-Pos sequences-based trained model outperforms in all three datasets with accuracy values 0.84, 0.96 and 0.87 respectively. This confirms that models learn more efficiently on Lex-Pos sequences of sentences. Also, if we notice the accuracy drops from training to testing, we observe that it is less in the author model trained from Lex-Pos based features. The accuracy drops are 7%, 3% and 6% for the author model trained on lexical sequences of three datasets respectively, whereas the values, 4%, 1% and and 3% have been recorded for the author model trained on Lex-Pos sequences. These results further confirm that the Essay model trained on Lex-Pos sequences are more capable of generalization and so, are more stable and efficient.

**Table 11.** Performance of Model [5] on lexical sequences.

| | Training | | Testing | |
|---|---|---|---|---|
| Dataset | Loss | Accuracy | Loss | Accuracy |
| 1 | 0.08 | **0.89** | 0.13 | 0.82 |
| 2 | 0.02 | **0.98** | 0.04 | 0.95 |
| 3 | 0.07 | **0.91** | 0.11 | 0.85 |

**Table 12.** Performance of Model [5] on Lex-Pos sequences.

| | Training | | Testing | |
|---|---|---|---|---|
| **Dataset** | **Loss** | **Accuracy** | **Loss** | **Accuracy** |
| 1 | 0.09 | 0.88 | 0.12 | **0.84** |
| 2 | 0.02 | 0.97 | 0.03 | **0.96** |
| 3 | 0.08 | 0.90 | 0.10 | **0.87** |

For the second comparison, we carry out experiments on the work proposed by the authors of [16]. In that paper, the authors developed a deep learning model with convolution and pooling layers for detecting article errors in the English sentences. We refer here to this model as the Article model. The Article model takes a sequence of $k$ words before and after the article as input in order to learn the surrounding context of the articles. The sequence is translated into a mathematical vector using pre-trained word embeddings. In order to replicate the Article model, we also design a similar CNN model with the same parameters as mentioned in the paper. Table 13 provides the values of these hyper-parameters.

**Table 13.** Hyper-parameters settings for Author Model [16].

| **Parameters** | **Context-Window Size (k)** | **Filter Windows** | **Dropout Rate** | **Regularizer** | **Classifier Layer** |
|---|---|---|---|---|---|
| Values | 6 | 3, 4, 5 | 0.5 | $l_2$-constraint | softmax |

The output of the Article model is multiclass with labels *a*, *an*, *the* and $\epsilon$ where $\epsilon$ indicates no article. The three datasets that have been used so far for training cannot be applied in training this author-model as these datasets have labels 0 and 1 for denoting the correct and incorrect sentences respectively. Therefore, for this comparative study, we design a new dataset from the correct sentences that have been earlier used for training the models and assign three labels 0, 1, and 2 depending on whether the sentences contain *a*, *an* or no article respectively. We do not consider "*the*" article for prediction as the dataset contains instances of single sentences only which are not sufficient to provide enough knowledge of specific and non-specific nouns. Similar to the first comparison study, we conduct two rounds of experiments. In the first experiment, we extract the context words from the sentences with window size 6 as mentioned by the authors, and translate this sequence into numeric vectors using word embeddings. Afterward, these feature values are supplied to the CNN-based author-model for training and testing. In the second set up, we provide the Lex-Pos sequences of sentences transformed using $W_E O_E$-feature vectors as input to the Article model.

Table 14 displays the results of both experiments where it can be clearly noticed that the author model performed extremely well on the Lex-Pos sequences of the new dataset by obtaining 99% accuracy, significantly higher than the accuracy yielded by the context-based sequence model, i.e., 90%. The high performance of the Lex-Pos model could be the result of adding phonetic information of the word used immediately after the article into the syntactic sequence of a sentence.

**Table 14.** Performance of Context-based and Lex-Pos Sequence on Author Model [16].

| | Training | | Testing | |
|---|---|---|---|---|
| **Sequence Type** | **Loss** | **Accuracy** | **Loss** | **Accuracy** |
| Context-based Lexical Sequence | 0.27 | 0.92 | 0.33 | 0.90 |
| Lex-Pos Sequence | 0.01 | **0.99** | 0.02 | **0.99** |

In order to make sure that the proposed approach is statistically significant, we further conducted a number of experimental trials to determine if the Lex-Pos-based classifier can be trusted over the author model. In this regard, 15 pairs of training and testing subsets were constructed by randomly selecting 10,000 and 2000 instances for each pair from the main training and testing set respectively.

Afterward, on each pair, both Lex-Pos-based and author model [5] were trained and the respective accuracy values have been recorded. Figure 1 presents a plot drawn from these accuracy values where the x-axis and y-axis represent values obtained by author classifier [5] and Lex-Pos-based classifier respectively. The graph clearly shows that for every pair of subset, Lex-Pos based classifier has performed better by obtaining a higher accuracy score. We also applied paired Student's *t*-tests (https://www.ruf.rice.edu/~bioslabs/tools/stats/ttest.html) on the two sets of accuracy scores to know if the distribution difference is statistically significant. We recorded the *t*-value as 11.516 with a *p*-value less than 0.05 which implies that the accuracy distribution of the two models is statistically different. Therefore, there is sufficient evidence to consider that the Lex-Pos model is better than the author model [5]. It is to be noted that statistical significance test was not conducted for comparing Lex-Pos-based grammar detector with author model [16] as we observed a considerably large improvement in the results, i.e., 9% increment in accuracy.

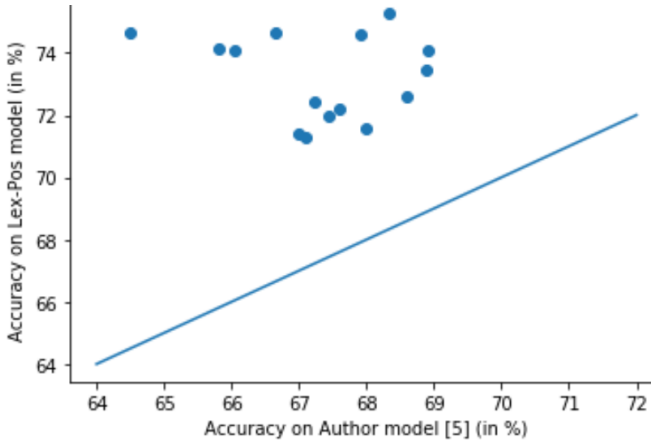


**Figure 1.** Accuracy values on 15 trials.

## 9. Discussion and Limitations

In this work, we have proposed the concept of converting an English sentence into a Lex-Pos sequence represented using a $W_E O_E$-feature vector in order to design a grammar detector that is capable of taking the advantage of both kinds of sequences, i.e., the specific nature of the lexical sequences and generic nature of syntactic sequences. We compare the performances of the Lex-Pos classifier with the models which are individually trained on lexical and POS-tag sequences of sentences. Lexical sequences were represented using word embedding vectors, while POS-tag sequences were represented using one-hot vector encoding. It is evident from the results that in terms of accuracy, the lexical-based models perform better than POS-tag-based models, whereas, in the context of stability, the POS-tag-based model proved to be more trustworthy. However, Lex-Pos sequence-based classifiers have proven to be the best systems in both aspects, accuracy and stability. This confirms the usefulness of providing additional linguistic knowledge to the POS-tag sequences of sentences and shows that the Lex-Pos sequences are more efficient in capturing the grammar structure of the English language.

In order to further demonstrate the potential of Lex-Pos , two grammar aware models of existing studies have been replicated. The first replica (LSTM-based Essay model) is designed to score the English sentence based on its correctness of grammar. And the second replica (CNN-based Article model) is modeled to classify the article errors in the sentence. The experiments show that both author models performed better on the Lex-Pos sequences than the sequences used in the respective papers. Furthermore, in these experiments also, Lex-Pos based trained author models are observed as more stable with lower accuracy-drops from training to testing.

Although the Lex-Pos models are found to be more efficient and trustworthy, there are also a few limitations associated with the present work. First, it does not ensure if a sentence is semantically valid, i.e., if the sentence is meaningful. The proposed model only verifies the grammatical structure of the

sentence, and therefore, it will not be able to discriminate the two sentences, $S_1$: *"I am eating a banana"* and $S_2$: *"I am running a banana"*. Both sentences are valid on syntax grounds but the second sentence fails on semantic context since *"I am running a banana"* does not make any sense in real life. Secondly, the proposed model is limited to individual sentences only and does not consider dependency between sentences. For example, consider the two sentences, $S_3$: *"I talked to a boy"* and $S_4$: *"She is great"*. If these two sentences are considered independently, then both are correct. But if these two sentences are considered in combination where the second sentence follows the first one, then instead of *"She"* as a subject in the $S_4$, *"He"* should have been used. These limitations have been considered as the future scope in the proposed work's direction.

## 10. Conclusions and Future Scope

In this paper, our main aim was to demonstrate that the proposed sequence, namely, Lex-Pos which incorporates both linguistic and structural information of a sentence, can lead to a significant improvement in the performance of grammar error detection. Since the Lex-Pos sequences contain both lexical and POS-tag tokens, these sequences have been translated into numerical values by providing a new embedding technique, i.e., $W_EO_E$-encoding. Also, the two types of error corpora have been designed for making the model learn about the lexical and POS-tag specific mistakes respectively. A total of three types of datasets have been used for conducting the experiments where an LSTM architecture was employed to design the grammar detection system.

In the experiments, we found that classifiers trained on lexical sequences yield more accurate results than the classifiers trained on POS-tag sequences. On the contrary, POS-tag-based models are observed as more stable than the lexical- ones. However, Lex-Pos based classifiers outperform the others in both parameters, accuracy and stability. Lex-Pos sequences are also found to be more efficient and trustworthy on the replica systems designed on the basis of existing studies. The comparative study shows that the Lex-Pos sequences can be further employed to design other grammar aware systems other than error detection, e.g., essay scoring system and grammar error correction system. The future scope can be to extend these sequences by imbibing semantic information using methods like named entity recognition in order to make the model learn about semantically valid or invalid sentences.

**Author Contributions:** N.A. and M.A.W. conceived and designed the experiments; N.A. performed the experiments; N.A. and M.A.W. analyzed the data; N.A. prepared the first draft of the paper; M.A.W. edited the paper; P.B. proofread the paper and supervised the overall work. All authors have read and agreed to the published version of the manuscript.

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

**Acknowledgments:** This work was carried out during the tenure of an ERCIM Alain Bensoussan Fellowship Program.

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

## References

1. Wagner, J.; Foster, J.; van Genabith, J. A comparative evaluation of deep and shallow approaches to the automatic detection of common grammatical errors. In Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL), Prague, Czech Republic, 28–30 June 2007; pp. 112–121.
2. Wagner, J.; Foster, J.; van Genabith, J. Judging grammaticality: Experiments in sentence classification. *Calico J.* **2009**, *26*, 474–490. [CrossRef]
3. Kaneko, M.; Sakaizawa, Y.; Komachi, M. Grammatical error detection using error-and grammaticality-specific word embeddings. In Proceedings of the Eighth International Joint Conference on Natural Language Processing, Tapei, Taiwan, 27 November–1 December 2017; Volume 1: Long Papers, pp. 40–48.
4. Xiong, D.; Zhang, M.; Li, H. Error detection for statistical machine translation using linguistic features. In Proceedings of the 48th annual meeting of the Association for Computational Linguistics, Uppsala, Sweden, 11–16 July 2010; pp. 604–611.

5. Taghipour, K.; Ng, H.T. A neural approach to automated essay scoring. In Proceedings of the 2016 Conference On Empirical Methods in Natural Language Processing, Austin, TX, USA, 1–5 November 2016; pp. 1882–1891.
6. Tezcan, A.; Hoste, V.; Macken, L. A neural network architecture for detecting grammatical errors in statistical machine translation. *Prague Bull. Math. Linguist.* **2017**, *108*, 133–145. [CrossRef]
7. Rozovskaya, A.; Roth, D. Training paradigms for correcting errors in grammar and usage. In Proceedings of the Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Los Angeles, CA, USA, 2–4 June 2010; pp. 154–162.
8. Chollampatt, S.; Ng, H.T. A multilayer convolutional encoder-decoder neural network for grammatical error correction. *arXiv* **2018**, arXiv:1801.08831.
9. Yuan, Z.; Briscoe, T. Grammatical error correction using neural machine translation. In Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego, CA, USA, 12–17 June 2016; pp. 380–386.
10. Liu, Z.R.; Liu, Y. Exploiting unlabeled data for neural grammatical error detection. *J. Comput. Sci. Technol.* **2017**, *32*, 758–767. [CrossRef]
11. Ji, J.; Wang, Q.; Toutanova, K.; Gong, Y.; Truong, S.; Gao, J. A nested attention neural hybrid model for grammatical error correction. *arXiv* **2017**, arXiv:1707.02026.
12. Chodorow, M.; Tetreault, J.; Han, N.R. Detection of grammatical errors involving prepositions. In Proceedings of the Fourth ACL-SIGSEM Workshop on Prepositions, Prague, Czech Republic, 28 June 2007; pp. 25–30.
13. Tetreault, J.; Foster, J.; Chodorow, M. Using parse features for preposition selection and error detection. In Proceedings of the ACL 2010 Conference Short Papers. Association for Computational Linguistics, Uppsala, Sweden, 11–16 July 2010; pp. 353–358.
14. Kochmar, E.; Briscoe, E. Detecting learner errors in the choice of content words using compositional distributional semantics. In Proceedings of the Association for Computational Linguistics, Baltimore, MD, USA, 22–27 June 2014.
15. Tetreault, J.; Chodorow, M. The ups and downs of preposition error detection in ESL writing. In Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008), Manchester, UK, 18–22 August 2008; pp. 865–872.
16. Sun, C.; Jin, X.; Lin, L.; Zhao, Y.; Wang, X. Convolutional neural networks for correcting English article errors. In *Natural Language Processing and Chinese Computing*; Springer: Berlin/Heidelberg, Germany, 2015; pp. 102–110.
17. Yang, Z.; Yang, D.; Dyer, C.; He, X.; Smola, A.; Hovy, E. Hierarchical attention networks for document classification. In Proceedings of the 2016 Conference of The North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego, CA, USA, 12–17 June 2016; pp. 1480–1489.


**Publisher's Note:** MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

© 2020 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (http://creativecommons.org/licenses/by/4.0/).