A Typo Correction System Using Artificial Neural Networks for a Text-based Ornamental Fish Search Engine

Hyunhak Song[1], Sungyoon Cho[1], Wongi Jeon[1], Kyungwon Park[1], Jaedong Shim[2], and Kiwon Kwon[1*]

[1]Korea Electronics Technology Institue (KETI)
Seoul, South Korea
[e-mail: {rainy930, sycho, jeonwg, kwpark, kwonkw}@keti.re.kr]
[2]Korean Aquatic animal Disease Inspector Association (KADIA)
Busan, South Korea
[e-mail: daone101@naver.com]
[*]Corresponding author: Kiwon Kwon

Received February 28, 2023; accepted June 12, 2023; published August 31, 2023

Abstract

Imported ornamental fish should be quarantined because they can have dangerous diseases depending on their habitat. The quarantine requires a lot of time because quarantine officers collect various information on the imported ornamental fish. Inefficient quarantine processes reduce its work efficiency and accuracy. Also, long-time quarantine causes the death of environmentally sensitive ornamental fish and huge financial losses. To improve existing quarantine systems, information on ornamental fish was collected and structured, and a server was established to develop quarantine performance support software equipped with a text search engine. However, the long names of ornamental fish in general can cause many typos and time bottlenecks when we type search words for the target fish information. Therefore, we need a technique that can correct typos. Typical typo character calibration compares input text with all characters in a calibrated candidate text dictionary. However, this approach requires computational power proportional to the number of typos, resulting in slow processing time and low calibration accuracy performance. Therefore, to improve the calibration accuracy of characters, we propose a fusion system of simple Artificial Neural Network (ANN) models and character preprocessing methods that accelerate the process by minimizing the computation of the models. We also propose a typo character generation method used for training the ANN models. Simulation results show that the proposed typo character correction system is about 6 times faster than the conventional method and has 10% higher accuracy.

Keywords: Artificial Neural Network, Ornamental Fish, Typo Correction, Typo Data Generation, Quarantine

A preliminary version of this paper was presented at ICONI 2022, and was selected as an outstanding paper.
This research was a part of the project titled 'Development of electronic illustrated book for ornamental fish', funded by the Ministry of Oceans and Fisheries, Korea.

http://doi.org/10.3837/tiis.2023.08.018 **ISSN : 1976-7277**

1. Introduction

COVID-19 has caused many changes in existing life, reduced contact between outside people, and created various indoor hobbies [1-3]. Among the hobbies, pet raising in house is increasing worldwide. Although, most of the pets are dogs and cats, the number of non-mainstream animals such as small mammals and ornamental fish is also increasing every year [4]. In addition, as pet exchanges between countries increase, the importance of quarantine to prevent risk factors such as the inflow and outflow of diseases is being emphasized [5-6].

South Korea is also conducting quarantine at the time of importation to block risk factors for all imported ornamental fish [7]. Quarantine of ornamental fish requires quarantine officers to carry out quarantine accurately and quickly based on various information. However, the actual quarantine environment is very poor. Since a quarantine site is a closed space, the Internet is not available, so ornamental fish information cannot be searched through a FishBase or Google search engine that provide fish species information [8]. Therefore, the quarantine officers collect prior information before quarantine to secure information on the ornamental fish. Quarantine officers prints a list of names of imported ornamental fish and photo documents taken in various directions for each ornamental fish. And the officers perform quarantine by visually comparing the target using the output. The heavy workload like preparing fish information can reduce the quarantine efficiency of the quarantine officers. Also, due to the long quarantine time, fish die under stress, resulting in financial loss [9]. These inefficient ornamental fish quarantine systems need to be improved.

In order to eliminate the time-consuming preparation of imported ornamental fish in South Korea, various information such as scientific names, names, diseases, and photos were collected, and servers were built to develop graphical user interface software. A search engine is needed to quickly find the required target among many ornamental fish stored on the server [10]. In text search, it is a common way to find the same match by comparing the input text with all the texts in the server. However, this method has two main disadvantages. First, as the number of pre-stored data increases, the texts to be compared increases, and real-time processing can be burdensome depending on the computing power of each device. Second, because each character of the two texts is compared, it is not possible to find an item that matches when typo characters occur during the typing process. In particular, the names of the ornamental fish are very long compared to ordinary words. We have 2,000 ornamental fish data, of which the longest ornamental fish name length is 62. There is a very high probability of typos in the process of typing long characters. Therefore, there is a need for a technology capable of accurately correcting typos while minimizing the amount of computation.

In this paper, we propose an ANN-based typo correction system that can correct typo characters that may occur when typing ornamental fish varieties. Furthermore, we propose an algorithm that can generate typo character data for learning ANN models. The goal is to achieve better performance than the existing typo correction accuracy and enable real-time processing like the commonly used typo correction method.

2. General Typo Correction Methods

General typo correction consists of comparing text and text with each other, as shown in **Fig. 1**. Assume that you enter text containing a specific typo character. Preprocessing performs standardization, tokenization, and vectorization on the input text. The standardization

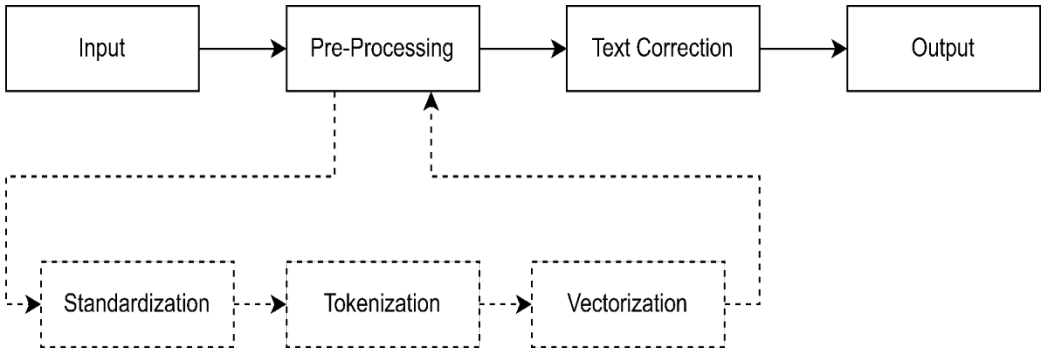


Fig. 1. Flowchart of a general typo character correction algorithm

modifies the text you enter to match specific character calibration criteria. Tokenization divides a given corpus into token units. Vectorization quantifies each character for character comparison. Text correction compares the prebuilt text list with the preprocessed input text. Comparison outputs the result of correcting the closest typo character. And sections 2.1 ~ 2.3 are algorithms based on common typo correction methods.

2.1 Peter Norvig's spelling corrector

Peter Norvig's spelling corrector is a typical simple typo correction algorithm [11]. When you receive a typo text, it outputs a text that corrects the typo keyword. The detailed operating principles are shown in **Fig. 2**. Assume that you entered text containing typo characters. The entered text utilizes the edit distance to generate several new texts. Edit distance is a way of comparing two texts, which is a numerical representation of how many edits are made so that when there are different texts, they are equal to each other. There are four ways to edit: 'Deletion', 'Substitution', 'Transposition' and 'Insertion'. 'Deletion' removes any one character from the text. "Substitution" changes any one character in the text to another. "Transposition" reverses two adjacent characters in a text. "Insertion" adds a new character to any location in the text. Compare the newly created text to a list of all texts in the prebuilt dictionary. When you use the above method to proofread English characters, you will get the following counts. Suppose there are 26 alphabets and a particular piece of text has a length of T_n. Each edit method produces the following number of texts: $Deletion = T_n, Subtitution = 26(T_n + 1), Transposition = T_n - 1, Insertion = 26(T_n + 1)$. Adding them all together, we get $54T_n + 25$. Finds matching similar text in the dictionary from the typo text entered, in order of decreasing edit distance. It typically targets text with an edit distance of 2 or less, providing good typo correction accuracy and processing speed. However, text with a lot of typo characters is difficult for edit distance to correct. Because of the large editing distance, the text generated by editing will grow rapidly, making real-time processing impossible. And the accuracy of correcting typo characters may gradually decrease.

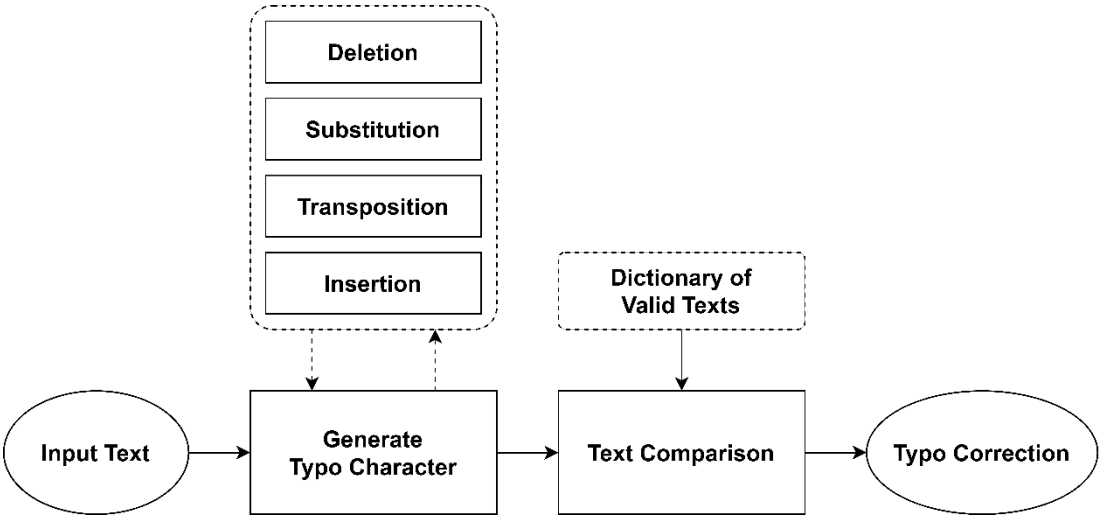


Fig. 2. The flow chart of Peter Norvig's spelling correction

2.2 Symspell

The Symspell algorithm improves on the shortcomings of Peter Norvig's spelling corrector and increases performance [12-13]. The Symspell algorithm uses "Deletion" only. Because Symspell generates $T_n - 1$ texts based on the alphabet, it is much faster to process than Peter Norvig's spelling corrector. The principle of operation is shown in **Fig. 3**. Build a proofreading dictionary for a prebuilt list of texts using the "Deletion" edit method. And we store them in a hash table so we can find them quickly. Apply the same editing methods to the input text and search for matching text in the proofreading dictionary.

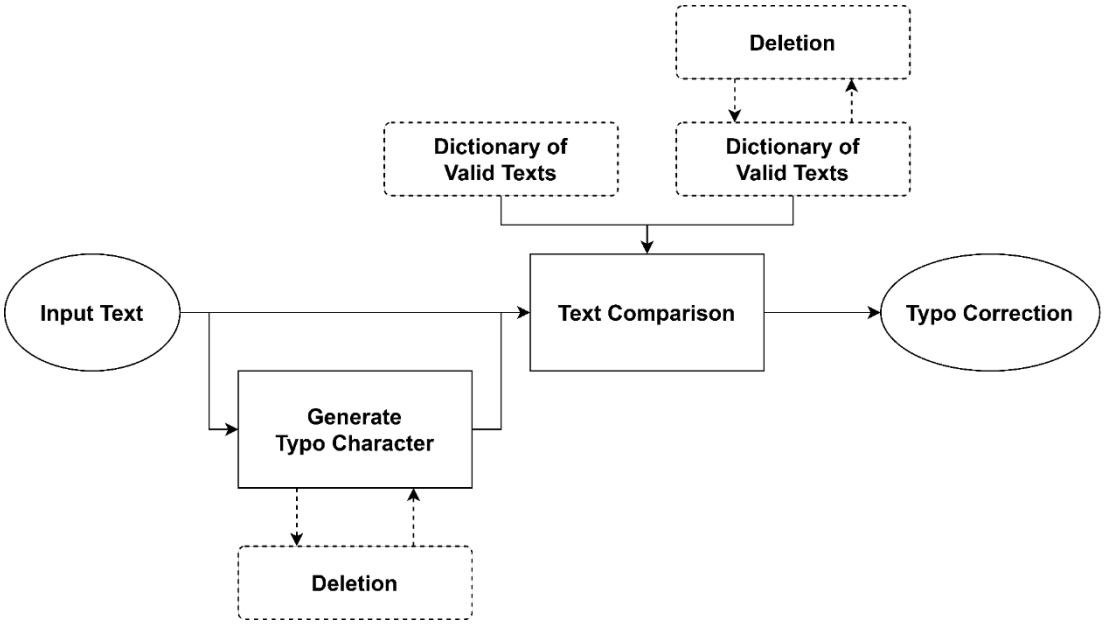


Fig. 3. The flow chart of Symspell

2.3 ANN-based text correction

ANN models can be used to correct misspelled characters in text, such as sentences and words, more accurately than methods sections 2.1 and 2.3. Artificial Neural Network (ANN) is a machine learning algorithm that mimics the principles and structure of a human neural network and can be applied to text proofreading [14]. We can train an ANN on word-in-sentence or character-in-word data to generate weights, and then use the neural network to make corrections [15-16]. The structure of an ANN model is related to its processing speed and accuracy. More layers and more neurons in a model generally means better typo correction accuracy, but it also means more computation and slower processing speed. This makes it difficult to apply to situations that require fast processing and accurate typo character correction.

Therefore, in section 3, we propose an ANN model-based typo correction system for long ornamental fish names. We built an ANN model with a simple structure for real-time processing and configured the system with a text preprocessing method to improve accuracy.

3. Proposed Ornamental Fish Name Typo Character Correction System

Ornamental varieties have a large maximum character length compared to common words. In our data set of 2,000 ornamental fish, the maximum name length is 62. Long character lengths make it very easy for users to make accidental typos when entering ornamental parts of speech into search engines. The longer the length of the text, the more likely it is that typos will occur during the typing process. It can also contain a large number of typos. The higher the number of typos, the more difficult it can be to correct them. In our proposed typo correction process, we designed the system based on text preprocessing and ANN model, considering both

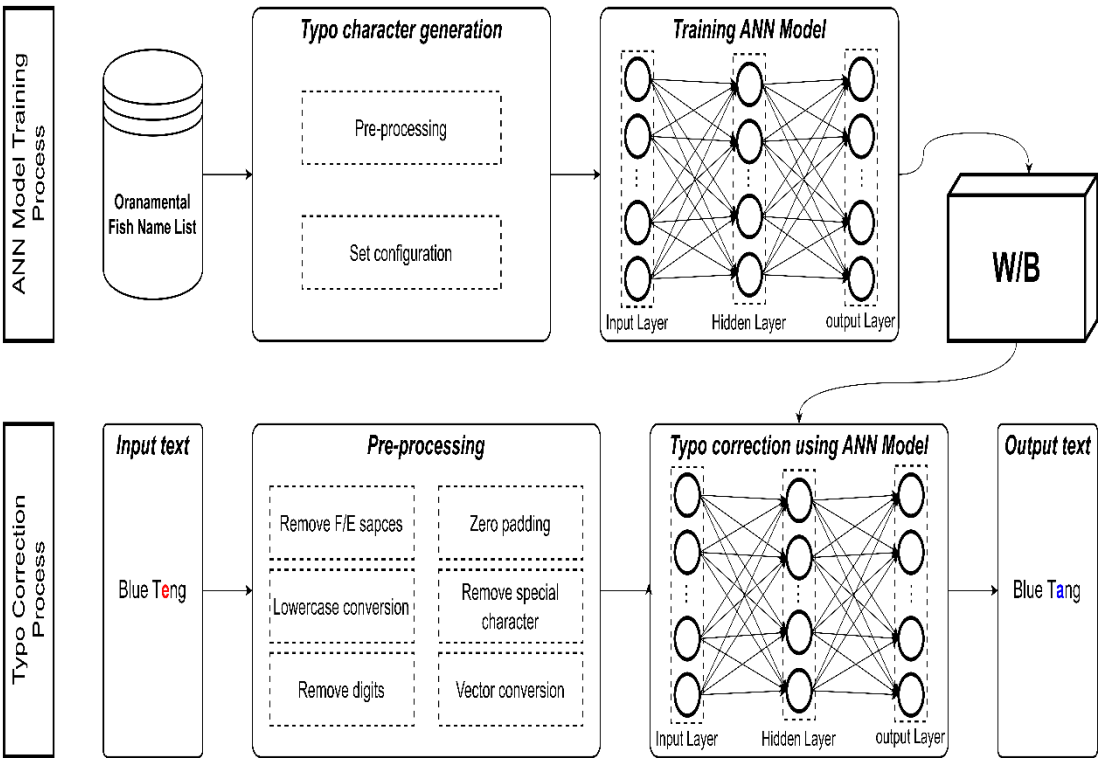


Fig. 4. Typo character correction system based on preprocessing and ANN model

accuracy and processing speed performance. As shown in **Fig. 4**, the ornamental name misspelling correction system consists of misspelling data generation, pre-training, preprocessing, and misspelling inference. Generate typo character data builds a training dataset and a validation dataset by generating multiple typo character data for the ANN model to train on. Pre-training builds the ANN model and feeds it training and validation datasets. Preprocessing converts characters for different contexts to fit the input characters into the ANN model input format. Typo character inference uses pre-trained weights to find the best correction character for an input character. And the details of the above parts are discussed in sections 3.1 ~ 3.3.

3.1 Generate typo character data

This section describes how to create training and validation datasets that contain typos. Because the number of characters in a search term is variable, we assume that N_C and N_T are the maximum number of characters in a search term that can be entered at once and the actual number of characters entered, respectively. Then N_T will be less than N_C. Also, assuming that each character maps to an L-bit vector, the size of one sample vector would be $D = N_C \times L$. **Table 1** shows an example mapping for lowercase alphabet English with $L = 5$. If the number of sample vectors in the training dataset is M, then the size of the training dataset is $M \times D$. This training matrix can be represented as $S = [c(0)\, c(1) \cdots c(M-1)]^T$. Where $c(i)$ is the ith sample vector, which can be represented as $c(i) = [b(0)\, b(1) \cdots\ b(N_C - 1)]^T$. Where $b(j)$ denotes each sample vector must contain at least one typo, and it may be advantageous to reduce the correlation between sample vectors to speed up the training convergence of the ANN. Therefore, in this paper, we assume that up to $N_E(i)$ typos can occur at any position of $c(i)$, when $N_E(i) \leq R \leq N_C$ for any natural number R.

Table 1. Vector conversion lookup table of 26 characters

Character	vector	Character	vector
space	[0,0,0,0,0]	n	[0,1,1,1,0]
a	[0,0,0,0,1]	o	[0,1,1,1,1]
b	[0,0,0,1,0]	p	[1,0,0,0,0]
c	[0,0,0,1,1]	q	[1,0,0,0,1]
d	[0,0,1,0,0]	r	[1,0,0,1,0]
e	[0,0,1,0,1]	s	[1,0,0,1,1]
f	[0,0,1,1,0]	t	[1,0,1,0,0]
g	[0,0,1,1,1]	u	[1,0,1,0,1]
h	[0,1,0,0,0]	v	[1,0,1,1,0]
I	[0,1,0,0,1]	w	[1,0,1,1,1]
j	[0,1,0,1,0]	x	[1,1,0,0,0]
k	[0,1,0,1,1]	y	[1,1,0,0,1]
l	[0,1,1,0,0]	z	[1,1,0,1,0]
m	[0,1,1,0,1]	-	-

3.2 Building an ANN model

This section describes modeling a training ANN for typo character correction using data. South Korea's ornamental fish quarantine is using tablet PCs without a Graphics Processing Unit (GPU). We designed our ANN model to take this existing environment into account. General-purpose ANN models for traditional natural language processing are deeply layered and computationally intensive. We needed a model that could exhibit high accuracy with low computation. We utilized a fully connected layer, which represents the most basic form of a neural network. The built neural network structure and hyper-parameters per layer are shown in **Fig. 5**. As a fully connected layer, each output node of Dense can represent the form of a matrix as $y_0 = f(W_0^T x + b_0)$. $y_1 = f(W_1^T x + b_1) \cdots y_{M-1} = f(W_{M-1}^T x + b_{M-1})$ with $y = f(Wx + b)$ as its form. Where y is the output vector, x is the input vector, b is the bias vector, W is the weight matrix, and M is the number of output vectors. Multiply the input by the weight, add the bias vector, and apply the activation function, f. The ANN model was built by sequentially stacking an Input Layer and three Dense Layers. The output of each layer becomes the input of the next layer. Input Layer takes in training data represented as vectors. It has a fixed input shape with a maximum character length multiplied by the length of the character representation vector. The number of neurons and the activation function should be set as hyper-parameters of Dense Layer. The higher the number of neurons, the better the learning performance, but the more computation it requires. Therefore, we set the number of neurons in each layer to 32, 16, and the number of classes. For activation functions, we used Relu and Softmax. Relu solves the slope problem for activation functions such as sigmoid and tanh, and is fast to learn and low in computational cost. Softmax was applied to the last output layer by normalizing the input value to a value between 0 and 1. For model training, we set the Optimizer and Loss. For the Optimizer, which can determine how the learning progresses, we applied Adam, which is a fusion of Momentum and RMSProp. Adam is an efficient computation and has very little memory requirements. Loss, which defines the feedback to be used for learning, applies Categorical Crossentropy, which is suitable for multiple classifications with multiple classes. The designed ANN model is used in the same form for learning and inference.

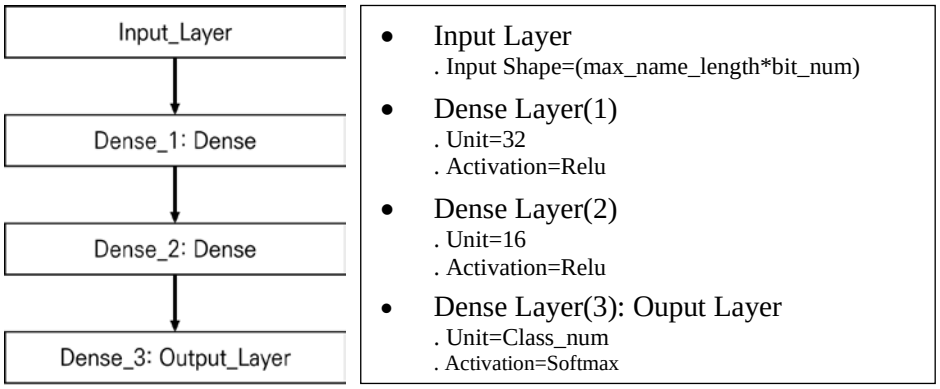


Fig. 5. ANN model structure and hyper-parameter

3.3 Character preprocessing

This section describes several methods for processing input characters. In order to input typo characters into an ANN model, they need to be transformed to match the model input format. The ANN model proposed in this paper can only process text with a total of 27 characters in the lowercase alphabet plus spaces, as shown in **Table 1**. This is why we need to remove unprocessable characters from the text before feeding it into the ANN model. We can preprocess text for any character that might be typed on a keyboard with a 101-key layout, which is basically what we use a lot. For the input text, we remove the leading and trailing spaces, because if the same word contains spaces, the entire word is shifted and the ANN model will take it as a completely different word. We convert uppercase English to lowercase by searching for uppercase alphabets, special characters, and numbers in the text, and remove special characters and numbers. We generate a vector with a value of zero equal to the length of the longest word in the entire data multiplied by the value of the representation vector, the same as the training data, and convert all the input characters into a vector according to the lookup table

Text preprocessing is necessary to reduce the number of training classes in an ANN model. By reducing the number of classes, you can reduce the amount of computation. Without text preprocessing, the performance results might be different if the ANN model generated classes for every character that would be input and processed them. We checked the performance results with and without text preprocessing.

Table 2. Vector conversion lookup table of 94 characters

Character	vector	Character	vector	Character	vector	Character	vector
space	[0,0,0,0,0,0,1]	9	[0,0,1,1,0,0,1]	Q	[0,1,1,0,0,0,1]	i	[1,0,0,1,0,0,1]
!	[0,0,0,0,0,1,0]	:	[0,0,1,1,0,1,0]	R	[0,1,1,0,0,1,0]	j	[1,0,0,1,0,1,0]
#	[0,0,0,0,0,1,1]	;	[0,0,1,1,0,1,1]	S	[0,1,1,0,0,1,1]	k	[1,0,0,1,0,1,1]
$	[0,0,0,0,1,0,0]	<	[0,0,1,1,1,0,0]	T	[0,1,1,0,1,0,0]	l	[1,0,0,1,1,0,0]
%	[0,0,0,0,1,0,1]	=	[0,0,1,1,1,0,1]	U	[0,1,1,0,1,0,1]	m	[1,0,0,1,1,0,1]
&	[0,0,0,0,1,1,0]	>	[0,0,1,1,1,1,0]	V	[0,1,1,0,1,1,0]	n	[1,0,0,1,1,1,0]
'	[0,0,0,0,1,1,1]	?	[0,0,1,1,1,1,1]	W	[0,1,1,0,1,1,1]	o	[1,0,0,1,1,1,1]
(	[0,0,0,1,0,0,0]	@	[0,1,0,0,0,0,0]	X	[0,1,1,1,0,0,0]	p	[1,0,1,0,0,0,0]
)	[0,0,0,1,0,0,1]	A	[0,1,0,0,0,0,1]	Y	[0,1,1,1,0,0,1]	q	[1,0,1,0,0,0,1]
*	[0,0,0,1,0,1,0]	B	[0,1,0,0,0,1,0]	Z	[0,1,1,1,0,1,0]	r	[1,0,1,0,0,1,0]
+	[0,0,0,1,0,1,1]	C	[0,1,0,0,0,1,1]	[	[0,1,1,1,0,1,1]	s	[1,0,1,0,0,1,1]
,	[0,0,0,1,1,0,0]	D	[0,1,0,0,1,0,0]	\	[0,1,1,1,1,0,0]	t	[1,0,1,0,1,0,0]
-	[0,0,0,1,1,0,1]	E	[0,1,0,0,1,0,1]	]	[0,1,1,1,1,0,1]	u	[1,0,1,0,1,0,1]
.	[0,0,0,1,1,1,0]	F	[0,1,0,0,1,1,0]	^	[0,1,1,1,1,1,0]	v	[1,0,1,0,1,1,0]
/	[0,0,0,1,1,1,1]	G	[0,1,0,0,1,1,1]	_	[0,1,1,1,1,1,1]	w	[1,0,1,0,1,1,1]
0	[0,0,1,0,0,0,0]	H	[0,1,0,1,0,0,0]	`	[1,0,0,0,0,0,0]	x	[1,0,1,1,0,0,0]
1	[0,0,1,0,0,0,1]	I	[0,1,0,1,0,0,1]	a	[1,0,0,0,0,0,1]	y	[1,0,1,1,0,0,1]
2	[0,0,1,0,0,1,0]	J	[0,1,0,1,0,1,0]	b	[1,0,0,0,0,1,0]	z	[1,0,1,1,0,1,0]
3	[0,0,1,0,0,1,1]	K	[0,1,0,1,0,1,1]	c	[1,0,0,0,0,1,1]	{	[1,0,1,1,0,1,1]
4	[0,0,1,0,1,0,0]	L	[0,1,0,1,1,0,0]	d	[1,0,0,0,1,0,0]	|	[1,0,1,1,1,0,0]
5	[0,0,1,0,1,0,1]	M	[0,1,0,1,1,0,1]	e	[1,0,0,0,1,0,1]	}	[1,0,1,1,1,0,1]
6	[0,0,1,0,1,1,0]	N	[0,1,0,1,1,1,0]	f	[1,0,0,0,1,1,0]	~	[1,0,1,1,1,1,0]
7	[0,0,1,0,1,1,1]	O	[0,1,0,1,1,1,1]	g	[1,0,0,0,1,1,1]	-	-
8	[0,0,1,1,0,0,0]	P	[0,1,1,0,0,0,0]	h	[1,0,0,1,0,0,0]	-	-

Table 2 shows the vector conversion table for all the characters that can be input. There are 94 in total, which is 67 more than before. Also, we need 7 bits to represent 94 characters, whereas before 27 characters could be represented with 5 bits. We built different ANN models with the same hyper-parameters except for the input form. **Table 3** shows the structure and parameters of the two ANN models for 27 classes in (a) and 94 classes in (b). The ANN model parameter for (b) is less than (a) because of the fewer classes and shorter character vector lengths. For the comparison of (a) and (b), we generated data for the typed characters in accordance with the method described in Section 3.1, and the experimental results are shown in **Table 4**. The accuracy is about 99%, but the processing speed is about 3.4 times different. The reason for the approximately three-fold difference in processing speed is that (b) has more computation in the character vectorization and inference model based on the number of classes than (a). This result shows that text preprocessing can reduce the computation of the ANN model and thus increase the processing speed.

Table 3. Comparison of the number of ANN model parameters between Table 1 and Table 2

Layer (type)	Activation	Output Shape	Param
Input	Linear	310	0
Dense	ReLU	64	19,904
Dense	ReLU	32	2,080
Dense	Softmax	2,000	66,000
Total Params: 87,984			

(a) Applying the character in **Table 1**

Layer (type)	Activation	Output Shape	Param
Input	Linear	434	0
Dense	ReLU	64	27,840
Dense	ReLU	32	2,080
Dense	Softmax	2,000	66,000
Total Params: 95,920			

(b) Applying the character in **Table 2**

Table 4. Comparison of processing time and accuracy performance of (a) and (b)

ANN Model Type	Number of typo characters									
	1 Typo		2 Typo		3 Typo		4 Typo		5 Typo	
	time (ms)	accuracy (%)	time (ms)	accuracy (%)	time (ms)	accuracy (%)	time (ms)	accuracy (%)	time (ms)	accuracy (%)
(a)	1.41	99.99	1.51	99.96	1.39	99.86	1.44	99.72	**1.39**	99.23
(b)	4.59	99.98	4.81	99.93	4.66	99.87	4.71	99.68	**4.69**	99.43

4. Experiment Results

In this section, we analyzed the performance of an existing character correction method, the Symspell algorithm, and our proposed method, the Typo Correction System, using data generated by the typo character generation method in section 3.1. We have not analyzed the performance for other ANN models with deep neural network structure, as real-time processing performance is required on low-end tablet PCs without GPUs. The test data used in the experiment consisted of 2,000 types of coronary names. The maximum number of characters in a dual name is 62. Therefore, I arbitrarily set the maximum number of characters to $N_C = 66$ to accommodate the extra typos. The remaining $N_C - N_T$ bits of each sample vector are filled with zeros. For training, 180 typo samples per 2,000 varieties are generated. Also, R is set to 10 so that $1 \leq N_E(i) \leq 10$. The size of the training dataset is 360,000. The

ANN model used in this paper is section 3.2. We implemented our model using the KERAS framework, an open source neural network library based on Python [17]. The size of the input layer is 310. The hyper-parameters for training are 20 for batch size and 20 for epoch. An AMD Ryzen 9 5900HS with Radeon Graphics 3.30GHz, 32GN RAM, and NVIDIA GeForce RTX 3080 is used for the hardware environment.

We evaluated the model with Accuracy, Precision, Recall, and F1-Score, which can be defined as the relationship between the answer output by the model and the actual correct answer, as shown in **Fig. 6**. TP predicts true answers as True, FP predicts false answers as True, FN predicts true answers as False, and TN predicts false answers as False. This allows us to express the classification performance metric as equations (1) through (4). **Fig. 7** shows the classification performance metric as a function of the number of epochs during the training process. We entered training and validation data at each epoch and graphed the results. Accuracy and Precision converged to approximately 99% at epoch 3, Recall converged to 99% at epoch 20, F1-Score converged to 99% at epoch 10, and the loss on the validation data converged to zero.

		Predicted	
		Positive	Negative
Actual	Positive	TP (True Positive)	FN (False Negative)
	Negative	FP (False Positive)	TN (True Negative)

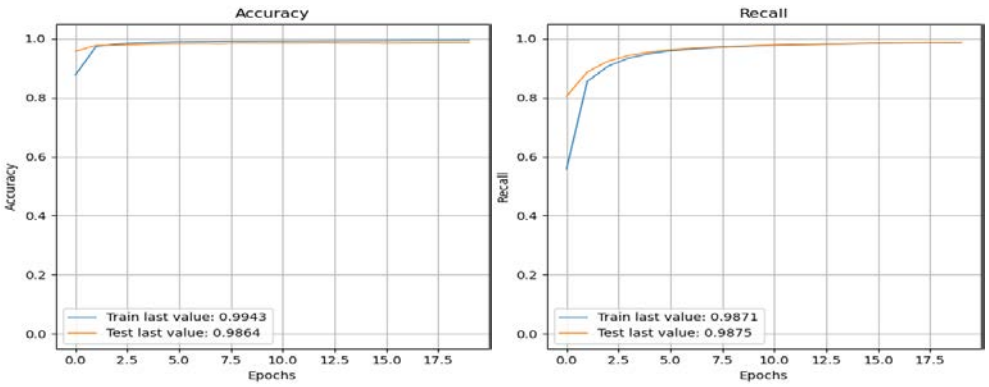
Fig. 6. Performance evaluation metrics of ANN classification models

$$\text{Precision} = TP/(TP + FP) \tag{1}$$

$$\text{Recall} = TP/(TP + FN) \tag{2}$$

$$\text{Precision} = TP + TN/(TP + FN + FP + TN) \tag{3}$$

$$\text{F1 Score} = 2 * Precision * Recall/(Precision + Recall) \tag{4}$$



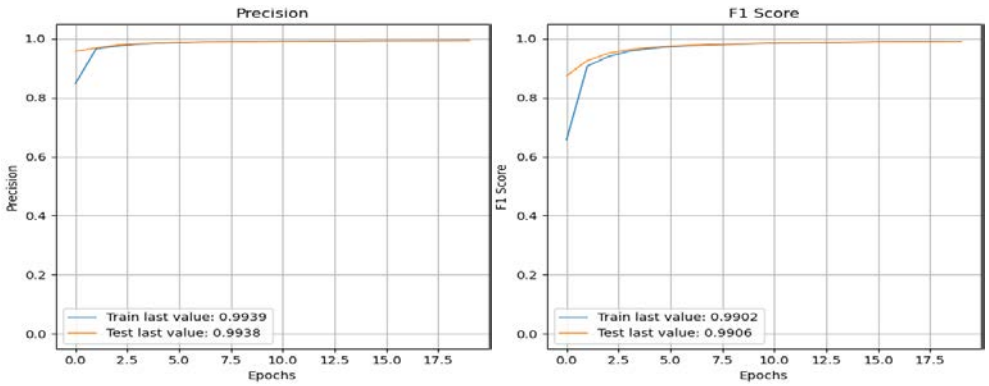


Fig. 7. ANN model performance by number of trainings

Table 5 shows the proofreading performance metrics based on the number of typos for the traditional proofreading method, Symspell, and the proposed proofreading system. For this simulation, we generated 10 of data for each class, totaling 20,000 of test data. Processing speed and accuracy were calculated by varying the number of typos contained in $c(i)$ in each sample vector from 1 to 10. Processing speed is the average of the time it takes for text to be input and output. Accuracy is the number of correct corrections as a percentage of the total number of input data.

Fig. 8 is a graphical representation of the **Table 5**. Both methods are approximately 99% accurate up to four typos, but the metrics diverge at five typos. When text containing 10 misspelled characters is entered into the system for suggestions, the accuracy is 98%, which is approximately 12% more accurate than the Symspell algorithm. The model we propose is a fixed neural network, which means that the processing speed is roughly the same regardless of the number of character typos. However, Symspell's processing speed increases dramatically as the number of typo characters increases. Up to three typo characters, the proposed model is slower than the Symspell algorithm, but faster from four. On 10 typo characters, the proposed model is 1.434 ms, roughly 6 times faster than Symspell.

Table 5. Comparison of processing time and accuracy of each typo correction algorithm

Method	**Number of typo characters**									
Proposed System	1		2		3		4		5	
	time (ms)	accuracy (%)	time (ms)	accuracy (%)	time (ms)	accuracy (%)	time (ms)	accuracy (%)	time (ms)	accuracy (%))
	1.14	99.99	1.51	99.96	1.39	99.86	1.44	99.72	1.39	99.23
	6		7		8		9		10	
	time (ms)	accuracy (%))	time (ms)	accuracy (%)	time (ms)	accuracy (%)	time (ms)	accuracy (%)	time (ms)	accuracy (%)
	1.42	99.35	1.43	98.96	1.43	97.89	1.42	96.75	1.43	**95.14**
Symspell	1		2		3		4		5	
	time (ms)	accuracy (%)	time (ms)	accuracy (%)	time (ms)	accuracy (%)	time (ms)	accuracy (%))	time (ms)	accuracy (%)
	0.34	99.95	0.76	99.81	1.22	99.61	1.83	99.18	2.59	98.09
	6		7		8		9		10	
	time (ms)	accuracy (%)	time (ms)	accuracy (%)	time (ms)	accuracy (%)	time (ms)	accuracy (%)	time (ms)	accuracy (%)
	3.55	96.76	4.36	95.03	5.31	92.6	6.69	89.71	8.69	**85.96**

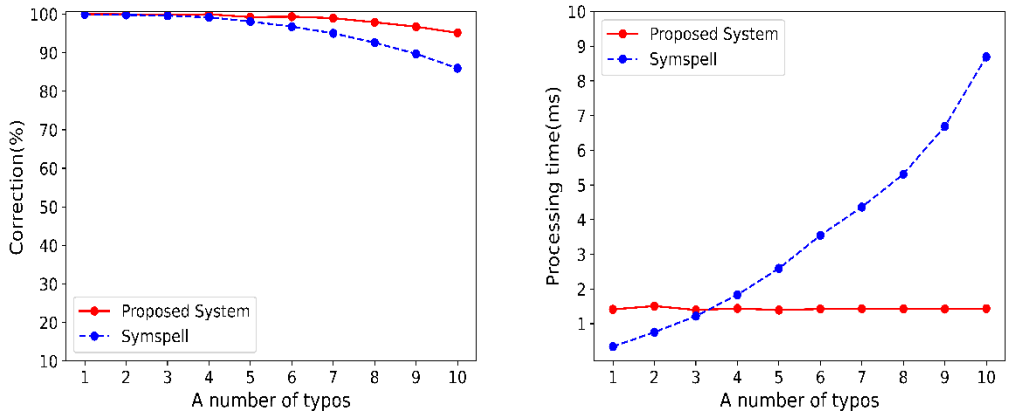


Fig. 8. Graph of time and accuracy values in Table 5

5. Conclusion

In this paper, we proposed a typo correction system for ornamental fish data search engine to quickly and accurately quarantine ornamental fish imported into South Korea. In our experiments, the proposed character correction system was about 6 times faster and 10% more accurate than the existing Symspell algorithm for words with many misspelled characters. This proves that the proposed typo correction system is suitable for ornamental varieties, where typing with long characters can lead to many typos. In this study, the system was built using 2,000 varieties of ornamental fish collected in advance, but it can be applied quickly by updating the weights for newly introduced ornamental fish through a simple structure of ANN model and training data generation method. It is a faster and more accurate character correction method than the existing one, and it can be processed in real time on a tablet PC used at the quarantine site for ornamental fish in South Korea. This study will update the system to provide similar word as well as typo character correction, considering natural patterns such as adjacent keys on keyboards where typo characters will occur probably. And we were performed on laptop PCs using artificially generated typo character datasets, but in the future, we plan to incorporate a system into an ornamental search engine and apply it to quarantine sites.

References

[1] Z. Ng, T. Griffin, and L. Braun, “The new status quo: enhancing access to human–animal interactions to alleviate social isolation & loneliness in the time of COVID-19,” *Animals*, vol. 11, no. 10, p. 2769, Sep. 2021. Article (CrossRef Link)

[2] A. Vincent, H. Mamzer, Z. Ng, and K. J. Farkas, “People and their pets in the times of the COVID-19 pandemic,” *Society Register*, vol. 4, no. 3, pp. 111-128, Apr. 2020. Article (CrossRef Link)

[3] U. Hunjan and J. Reddy, “Why companion animals are beneficial during COVID-19 pandemic,” *Journal of Patient Experience*, vol. 7, no. 4, pp. 430-432, Jul. 2020. Article (CrossRef Link)

[4] J. Ho, S. Hussain, and O. Sparagano, “Did the COVID-19 pandemic spark a public interest in pet adoption?,” *Frontiers in Veterinary Science*, vol. 8, May. 2021. Article (CrossRef Link)

[5] R. Whittington and R. Chong, "Global trade in ornamental fish from an australian perspective: the case for revised import risk analysis and management strategies," *Preventive Veterinary Medicine*, vol. 81, no. 1-3, pp. 92-116, Sep. 2007. Article (CrossRef Link)
[6] C. McDermott and B. Palmeiro, "Updates on Selected Emerging Infectious Diseases of Ornamental Fish," *Veterinary Clinics of North America: Exotic Animal Practice*, vol. 23, no. 2, pp. 413-428, May. 2020. Article (CrossRef Link)
[7] M. Cho, K. Kim, E. Jung and S. Jung, "Global outbreaks and strategies to control the emerging diseases in aquaculture farms in Korea," *Korea Maritime Institute*, vol. 34, no. 1, pp. 67-88, Jun. 2019. Article (CrossRef Link)
[8] C. Boettiger, D. Lang, and P. Wainwright, "Rfishbase: exploring, manipulating and visualizing FishBase data from R," *Journal of Fish Biology*, vol. 81, no. 6, pp. 2030-2039, Nov. 2012. Article (CrossRef Link)
[9] N. Masud, A. Ellison, and J. Cable, "A neglected fish stressor: mechanical disturbance during transportation impacts susceptibility to disease in a globally important ornamental fish," *Diseases of Aquatic Organisms*, vol. 134, no. 1, pp. 25-32, Apr. 2019. Article (CrossRef Link)
[10] T. Seymour, D. Frantsvog, and S. Kumar, "History of search engines," *International Journal of Management & Information Systems (IJMIS)*, vol. 15, no. 4, pp. 47-58, Sep. 2011. Article (CrossRef Link)
[11] T. Fahrudin, I. Sa'diyah, L. Latipah, I. Atha Illah, C. Bey Lirna, and B. Acarya, "KEBI 1.0: indonesian spelling error detection system for scientific papers using dictionary lookup and Peter Norvig spelling corrector," *Lontar Komputer: Jurnal Ilmiah Teknologi Informasi*, vol. 12, no. 2, pp. 78-90, Aug. 2021. Article (CrossRef Link)
[12] E. Mon, Y. Thu, T. Yu, and A. Wai Oo, "SymSpell4Burmese: symmetric delete Spelling correction algorithm (SymSpell) for burmese spelling checking," in *Proc. of 16th International Joint Symposium on Artificial Intelligence and Natural Language Processing (iSAI-NLP)*, IEEE, pp. 1-6, Dec. 2021. Article (CrossRef Link)
[13] M. Ndiaye and A. Faltin, "A spell checker tailored to language learners," *Computer Assisted Language Learning*, vol. 16, no. 2-3, pp. 213-232, Jul. 2003. Article (CrossRef Link)
[14] M. Madhiarasan and M. Louzazni, "Analysis of artificial neural network: architecture, types, and forecasting applications," *Journal of Electrical and Computer Engineering*, vol. 2022, pp. 1-23, Apr. 2022. Article (CrossRef Link)
[15] A. Ahmadzade and S. Malekzadeh, "Spell correction for azerbaijani language using deep neural networks." *arXiv*, Feb. 2021. Article (CrossRef Link)
[16] J. Lee, M. Kim, and H. Kwon, "Deep learning-based context-sensitive spelling typing error correction," *IEEE Access*, vol. 8, pp. 152565-152578, Jul. 2020. Article (CrossRef Link)
[17] J. Moolayil, "An introduction to deep learning and keras," *Learn Keras for Deep Neural Networks*, pp. 1-16, Dec. 2018. Article (CrossRef Link)

Hyunhak Song received the B.S. and M.S. degrees in Information & Communication Engineering from Hoseo University, Asan, Korea, in 2019 and 2021. He has been a researcher in Korea Electronics Technology Institute (KETI), Korea, since 2021. His research interests include computer vision, deep learning, and artificial intelligence.

Sungyoon Cho received the B.S., M.S., and Ph.D. degrees in electrical and electronic engineering from Yonsei University, Seoul, Korea, in 2006, 2008, and 2013, respectively. From 2013 to 2020, he was with Samsung Electronics, Korea, as a Staff Engineer to research cellular communication systems and develop 4G and 5G modem chipset. Since 2020, he has been with Korea Electronics Technology Institute (KETI) as a Principal Researcher to develop the advanced technologies for embedded network. His research interests are in the fundamental aspects of wireless communication and signal processing, and learning algorithms for practical application.

Wongi Jeon received the B.S., M.S., and Ph.D. degrees in electronic engineering from Chung-Ang University, Seoul, Korea, in 1994, 1996, and 1999, respectively. In 2001, he joined in the Korea Electronics Technology Institute (KETI), Korea, where he is presently a senior researcher. His research interests are in the area of intelligent signal processing and wireless digital communications, especially for multicarrier transmission systems.

Kyungwon Park received the B.S. and M.S. degrees in electrical engineering from Chung-Ang University, Seoul, Korea, in 1999 and 2001, respectively, and Ph.D. degrees in the school of electrical and electronic Engineering from Chung-Ang University in 2005. Since 2005, he has been working at Korea Electronics Technology Institute (KETI), Korea, where he is presently the Principal Researcher of Smart Network Research Center. His research interests are in the areas of wireless/wired communications system design and multiple antenna systems including multiple-input multiple-output systems and smart antenna systems.

Jaedong Shim received the B.S. degrees in department of aquatic life medicine from Pukyong National university, Busan, Korea, in 1997, He also received M.S. degrees in Gyeongsang National University, tongyeong, Korea, in 2017. Since 2016, he has been inaugurated as president in Korea Aquatic animal Disease Inspector Association (KADIA). His research interests are in the area of aquatic organisms, ornamental fish, and disease diagnosis, aquatic medicines.

Kiwon Kwon received B.S. and M.S. degrees in computer engineering from Kwangwoon University, Korea, in 1997 and 1999, He also received the Ph.D. degree in the School of Electrical & Electronics Engineering from Chung-Ang University, Korea, in 2011. In 1999, he joined in KETI, Korea, where he is currently a Group Leader with Oceans and Fisheries ICT Group. His research interests are in the area of advanced broadcasting/communication system, digital twin, oceans and fisheries ICT.