International Journal of Combinatorial Optimization Problems and Informatics, 13(2), May-Aug 2022, 65–75. ISSN: 2007-1558

www.editada.org

English mispronunciation detection module using a Transformer network integrated into a chatbot

Marcos E. Martinez-Quezada, J. Patricia Sánchez-Solís, Gilberto Rivera, Rogelio Florencia, Francisco López-Orozco*
División Multidisciplinaria de Ciudad Universitaria / Universidad Autónoma de Ciudad Juárez, Chih 32500, Mexico
*Correspondence: rogelio.florencia@uacj.mx

Abstract. Today it is crucial to have up-to-date information for companies to be more competitive in this business world. There are applications based on speech recognition that allows access to data stored in databases. However, the proper functioning of these applications lies in good pronunciation, a skill that most people do not have. In this paper, the architecture of an English mispronunciation detection module integrated into a chatbot is proposed. It allows users to enter the audio of the phrases in which they want to evaluate their pronunciation. The output is the mispronounced words, thus helping the user to practice their English language pronunciation. The proposed architecture consists of an Automatic Speech Recognizer (ASR) model based on a Transformer network that converts the audio signal to text and an algorithm for string alignment that identifies mispronounced words using the Levenshtein distance. The Transformer network was trained using the LibriSpeech and L2-ARTIC datasets. The module was evaluated using the Accuracy metrics, reaching 90%, and the Character Error Rate metric, reaching 9.5%. Additionally, its performance was evaluated on a group of real users, showing promising results.
Keywords: Mispronunciation detection, Automatic Speech recognition, Transformer Network.

Article Info

Received: August 31, 2021
Accepted: October 23, 2021

1 Introduction

Business Intelligence refers to the tools and strategies used in the processing, analysis, and visualization of data to support decision-making in companies [1]. Accessing up-to-date information in real time, stored on company servers or in the cloud, could allow decision-makers to have the certainty of carrying out business operations and obtaining favorable dividends for their companies. In this sense, applications based on speech recognition are intended to answer queries expressed in natural language by users [2]. However, on the one hand, for these applications to achieve a good performance, the pronunciation of the users is a key element, a skill that most people do not have. On the other hand, most of these applications have been developed for English, which is the predominant language in this globalized world.

Pronunciation is often the most difficult skill to develop when learning a second language. Interaction with other people is a key point in developing speech skills. However, sometimes learning partner is not available, which may delay the improvement of this skill [3]. There are several tools that can help learners in language learning, such as websites or apps. One of those tools is chatbots, which has been well received in the second language learning task [4]. Additionally, Automatic Speech Recognition (ASR) systems are often used in the mispronunciation detection task [5]. Considering that frequent interaction with a chatbot could allow users to improve their pronunciation skills, the integration of both chatbots and ASR systems could be useful to emphasize the pronunciation of the language.

Different techniques have been implemented to satisfy the mispronunciation detection task using ASR systems. For example, in [6], a system was proposed that obtains scores for mispronounced English words. Such a system was developed using an algorithm called Goodness of Pronunciation (GOP). This algorithm produces scores of each one of the phonemes found in the speech signal. Phonemes below a threshold are considered mispronounced. The algorithm uses an ASR system developed using the Hidden Markov Model (HMM). HMM represents phonemes like a set of states and the probability between them like a transition. However, HMM has the disadvantage that it only updates one state at each timestamp. Due to the above, Recurrent Neural Networks (RNNs) were implemented. RNNs can use their internal state (memory) to remember long-term dependencies. However, its high computational cost becomes a disadvantage [7].

The rapid growth in the use of Deep Learning has led to the development of new techniques to build ASR systems with better performance; among these are End-to-End (E2E) models; these models transform an input sequence into an output sequence [8]. A novel technique to process and transform sequence was proposed in [9], the Transformer network. It was initially used in automatic translation tasks, but its rapid growth and performance permitted its implementation in ASR tasks. This network has a better performance compared to RNNs [10]. It allows processing sequences in larger quantities because it does not process one sequence at a time but many of them in parallel. This technique was used in [11] for the mispronunciation detection task.

The mispronunciation detection systems require two inputs that the user must enter: the voice audio to be processed and its respective text transcription. Having the transcript makes it possible to use other methods for mispronunciation detection. One of these commonly used methods is the string alignment algorithm [12]. This algorithm is used to find discrepancies between the text recognized by the ASR and the text transcription entered by the user.

This paper describes the implementation of an English mispronunciation detection system into a chatbot. The system is composed of two components: an ASR module and a string alignment module. The ASR module is composed of a Transformer network trained in two datasets: LibriSpeech and L2-ARTIC. LibriSpeech is composed of native English audios, and L2-ARTIC is composed of non-native English audios with different accents and mispronunciations. For the detection of mispronunciations, Levenshtein distance was used. Finally, the mispronunciation detection system was integrated into the architecture of an AIML-based chatbot so that users could interact with the system more naturally.

The paper is structured as follows: Section 2 presents a background of all the techniques used in this work. Section 3 describes the general architecture of the pronunciation error detection system. Section 4 shows the experiments carried out as well as the results obtained. Section 5 presents the conclusions and findings of this research.

2 Background

2.1 Transformer network

The Transformer network was initially used for the machine translation task and other NLP problems [9]. However, due to its good performance, it began to be used in ASR systems. In [13], it was used for the first time for this task, receiving the name of the *Speech-Transformer* network. Like all ASR E2E systems, the objective of this network is to transform a speech audio signal input into its corresponding character sequence output. The architecture of the network is shown in Figure 1.

This network architecture consists of two parts: an *encoder* and a *decoder*. The encoder transforms the audio feature sequences $(x_1, \dots, x_T)$ into a hidden representation $h = (h_1, \dots, h_L)$. Next, h is supplied to the decoder to generate an output sequence $(y_1, \dots, y_S)$ character by character. The decoder uses the previous character as additional input to output the next character in each time sequence. The encoder is composed of two blocks: *Multi-Head Attention* and *Feed-Forward Network*, and the decoder is composed of three blocks: *Masked Multi-Head Attention*, *Multi-Head Attention*, and *Feed-Forward Network*. Encoder and decoder blocks are composed of sub-blocks; these are:

a) Scaled Dot-Product Attention

First, a process called the self-attention mechanism is calculated to relate different positions of an input sequence of audio features. It helps to understand how the sequences are related to each other. For this, three inputs are necessary: *queries*, *keys* of dimension d_k, and *values* with dimension d_v. All these three inputs are composed of audio features. As Figure 2 shows, the Dot-Product between the queries and the keys is calculated, resulting in a matrix that determines the degree of relation between the

segments. Next, the matrix elements are divided by $\sqrt{d_k}$ so that the *softmax* function avoids regions with small gradients. The attention function is calculated on a set of queries packed into a matrix *Q*. The keys and values are also packed in matrices *K* and *V*, respectively. Finally, the resulting matrix is multiplied by the values matrix. The output of this process is calculated as:

$$Attention(Q,K,V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \quad (1)$$

Where:

- $Q \in \mathbb{R}^{t_q \, x \, d_q}$ represent the queries
- $K \in \mathbb{R}^{t_k \, x \, d_k}$ represent the keys
- $V \in \mathbb{R}^{t_v \, x \, d_v}$ represent the values

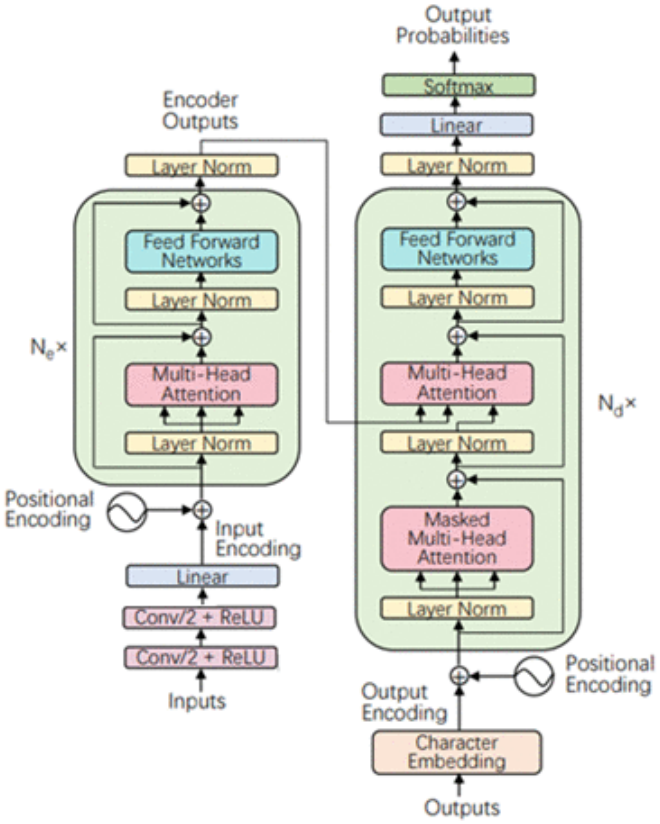

Fig. 1. Transformer architecture

b) Multi-Head Attention

This process calculates *h* times the attention of Scaled Dot-Product (Figure 3). Before performing each attention, three linear layers transform the queries, keys, and values into more discriminative representations. These layers are different for each of the heads and have their own parameters. In this way, each of the heads calculates a series of different values. The outputs of the *h* heads are concatenated and fed into another linear layer to obtain an output of dimension d_{model} finally. It is calculated by:

$$MultiHead(Q,K,V) = Concat(head_1, \ldots, head_h)W^O \quad (2)$$

$$head_i = Attention(QW_i^Q, KW_i^K, VW_i^V) \quad (3)$$

Where:

- Q, K, and V have dimension d_{model}

- $W_i^Q \in \mathbb{R}^{d_{model} \, x \, d_q}$, $W_i^K \in \mathbb{R}^{d_{model} \, x \, d_k}$, $W_i^V \in \mathbb{R}^{d_{model} \, x \, d_v}$ y $W_i^O \in \mathbb{R}^{hd_v \, x \, d_{model}}$ are arrays of parameters.

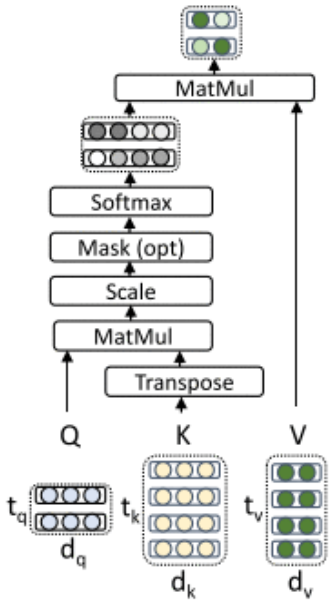

Fig. 2. Scaled Dot-Product Attention

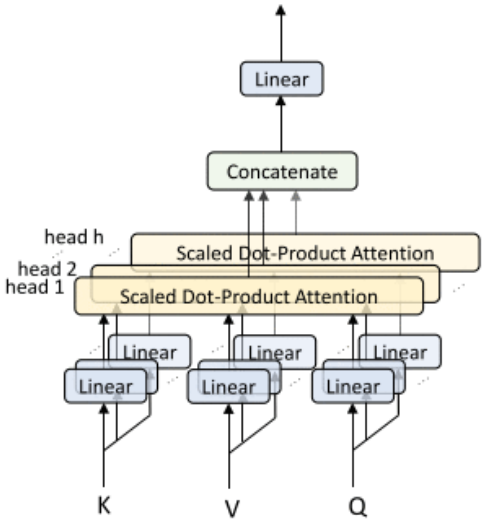

Fig. 3. Multi-Head Attention

c) Feed-Forward Networks

A fully connected *feed-forward* network is added to the encoder and decoder layers to transform the output to dimension d_{model}. It consists of two layers with a linear *Rectified Linear Unit* (ReLU) activation function. The dimension of the input and output is d_{model}, while the inner layer has dimension d_{ff}. It is defined as:

$$FFN(x) = max(0, xW_1 + b_1)\, W_2 + b_2 \tag{4}$$

Where:

- $W_1 \in \mathbb{R}^{d_{model} \, x \, d_{ff}}$ y $W_2 \in \mathbb{R}^{d_{ff} \, x \, d_{model}}$ are matrices of weights parameters.
- $b_1 \in \mathbb{R}^{d_{ff}}$ y $b_2 \in \mathbb{R}^{d_{model}}$ are bias parameters.

2.1.1 Encoder

The encoder input is a speech audio signal converted to a vector numeric representation using a feature extraction technique, commonly is employed the *Mel Frequency Cepstral Coefficients* (MFCC) for this purpose. Because these feature sequences are commonly longer than character sequences, two 3x3 CNN layers are used to treat the length mismatch between both sequences. This is achieved through the *Acoustic Event Detection* technique, which detects and classifies acoustic events where there is no voice [14]. The output of the above is a flattened feature vector to which a linear transformation is performed to obtain the vectors of dimension d_{model}.

Because there is no recurrence process in the Transformer network, as used in an RNN, it is necessary to add information about the positions of the input segments. For this reason, a positional encoding is used for the input vectors before they are sent to the encoder block. This is represented by the following:

$$PE(pos, 2i + 1) = \cos\left(pos \frac{pos}{10000^{2i/d_{model}}}\right) \tag{5}$$

$$PE(pos, 2i) = \text{sen}\left(pos \frac{pos}{10000^{2i/d_{model}}}\right) \tag{6}$$

Where:

- pos represents the position in the sequence.
- i represents the dimension.

For each even timestamp, a vector is created using Equation 5, while for each odd timestamp, a vector is created using Equation 6. Because both new vectors have the same dimension d_{model}, it is possible to sum the input vectors and the positional vectors, which are going to be the input to the *N* encoder blocks. These vectors are processed by the sub-blocks mentioned above: *Multi-Head Attention* and *Feed Forward Network*. A normalization layer and residual connection are introduced to each sub-block to standardize the neuron activation for effective training.

2.1.2 Decoder

The decoder employs a learned character-level embedding to convert the character sequence to the output encoding, which has a dimension d_{model}. These vectors are also added with positional encoding. The vectors are input to a stack of *N* decoder blocks to obtain the final decoder outputs. Each decoder has three sub-blocks: *Masked Multi-Head Attention*, *Multi-Head Attention*, and *Feed-Forward Network*. The *Masked Multi-Head Attention* sub-block, in the same way, receives *queries*, *keys*, and *values* as input. It is used to ensure the predictions for position *j* only depend on known outputs at positions less than *j*. It permits each segment probability in the attention matrix to be ignored by itself (Figure 4). The *Multi-Head Attention* sub-block feeds on keys and values from the encoder output and from the previous sub-blocks output queries. The output goes through the last sub-block, the *Feed-Forward Network*. As in the encoder, normalization and residual connection layers are added between the sub-blocks. Finally, a linear layer and a *softmax* function transform the decoder outputs into class probabilities output. This output is used again as input to the decoder. This process is repeated to calculate the probability of the next character to the end of the speech audio signal.

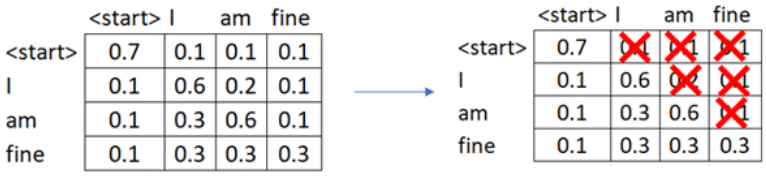

Fig. 4. Example of Attention matrix with probabilities deleted

2.2 Sequence alignment

String-matching algorithms are used to find possible mispronunciations. Two strings are required: the canonical transcription and the string inferred by ASR. They are compared to find differences between the words in both strings, which would indicate

a mispronunciation. In [12], the Needleman-Wunch algorithm was used to calculate the *insertion*, *deletion*, and *substitution* errors of two sequences. Subsequently, mispronunciation detection can be carried out. This algorithm was also used in [15], achieving an accuracy of 87.93%. The algorithm can be used in different linguistic units, *characters*, or *phonemes* which are the most common.

In this research, we used the *Levenshtein distance algorithm* to find the similarity measure between two strings: the canonical string(s) and the target string(t). The total distance is the number of *insertions*, *deletions*, *or substitutions* required to transform *s* in *t*. The larger the distance, the greater the difference between both strings [16]. Algorithms like Levenshtein distance are used to find possible mispronunciations. Mispronunciation detection systems commonly use a speech audio signal and a text transcription of that audio. Then, the differences between the text inferred by ASR and the given text transcription denote a mispronunciation.

After applying the Levenshtein distance algorithm and finding the distance (number of *insertions*, *deletions*, and *substitutions*), a metric can be used to obtain the error rate. The *Character Error Rate* (CER) is used to evaluate ASR performance, but it can also be used to determine how much correct or incorrect the pronunciation is. CER is calculated using the equation:

$$CER = \frac{S+D+I}{N} \tag{7}$$

There exist different techniques to achieve the mispronunciation detection task. The state-of-the-art mention the Goodness of Pronunciation [17] and Extended Recognition Networks [18] as techniques to implement the mispronunciation detection. However, it also is mentioned that those techniques have lower performance compared with Transformer networks [11]. For that reason, another mispronunciation detection technique to make a comparison was not implemented.

2.3 AIML chatbot

The mispronunciation detection module was integrated into a chatbot to create a more natural interaction between the user and the system. It was decided to develop its knowledge base using the *Artificial Mark-Up Language* (AIML) to facilitate the integration into the chatbot. AIML is based on XML and is used to describe the linguistic knowledge of conversational agents. Knowledge is managed with textual units called *categories* [19]. When the user sends input text to an AIML chatbot, it finds the most similar pattern and responds. A pattern is composed as follows:

```
<category>
      <pattern>WHAT IS THE LONGEST STATE IN US</pattern>
            <template>
                  THE LONGEST STATE IN US IS ALASKA.
            </template>
</category>
```

The pattern '*WHAT IS THE LONGEST STATE IN US*' returns the text between the <template> tags. Patterns also can be built in a general form. An example is the use of the symbol '*'. Unlike the example above, in which the user must enter the exact pattern so that the chatbot can associate the correct answer, this symbol allows the input of any text. In that case, the pattern is structured as the following way:

```
<category>
      <pattern>WHY * IS YOUR FAVORITE FOOD</pattern>
            <template>
                  BECAUSE IT IS DELICIOUS.
            </template>
</category>
```

A graphic interface was created for users to interact with the mispronunciation detection system (Figure 5). It is composed of:

1. Activate or deactivate pronunciation training. If it is active, the chatbot answers and returns possible mispronunciations; otherwise, the chatbot just answers.
2. Text field to enter messages to the chatbot.

3. Conversation history.
4. Area in which mispronounced words are displayed.

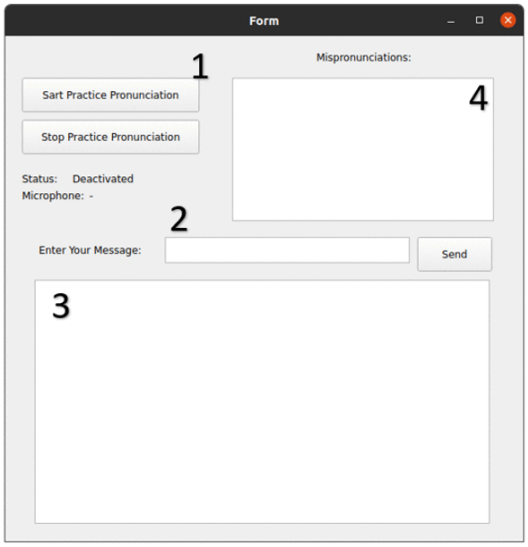

Fig. 5. Mispronunciation detection graphic interface

3 Mispronunciation detection architecture

The entire system pipeline is shown in Figure 6. Firstly, it was necessary to collect the dataset, which is composed of speech audio signals and their respective text transcriptions. Then, feature extraction is used to convert the audio signal to a numeric representation ready to fit the model. The Transformer network is trained with all the dataset and, once the process is done, results in the ASR E2E model. After training the ASR model, the user must enter the audio of the phrase to be analyzed and its respective text transcription. Audio signal features are extracted and sent to the ASR model to infer the text. The recognized text is compared to the text transcription entered by the user using the Levenshtein distance algorithm to find possible mispronunciations.

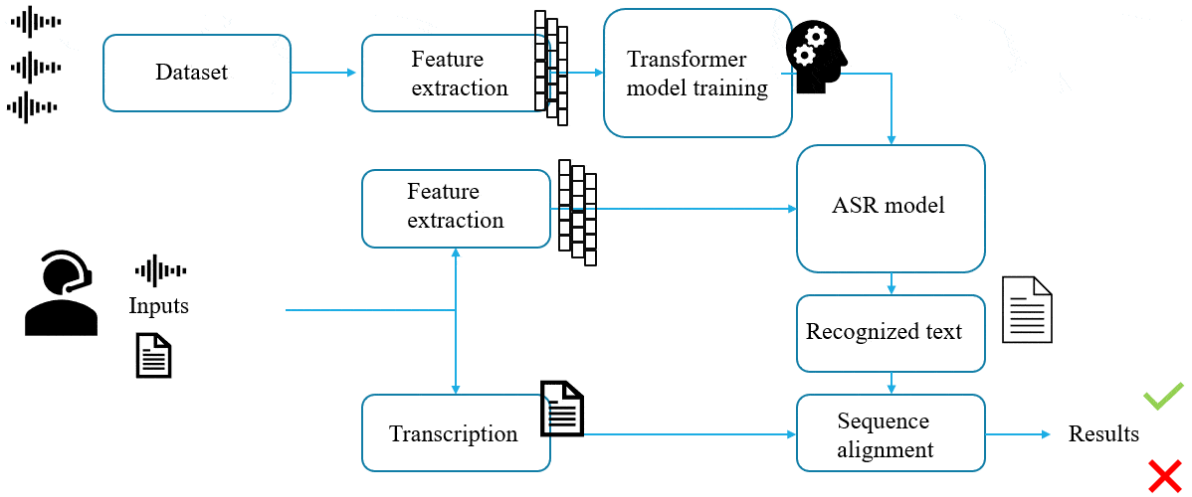

Fig. 6. Mispronunciation detection architecture

4 Experiments and results

Two data sets, LibriSpeech [20] and L2-ARCTIC [21], were used to evaluate the performance of the mispronunciation detection module. LibriSpeech contains 1,000 hours of audio recorded by people whose native language is English. It consists of two sets:

train set with 960 hours and test set with 40 hours. L2-ARCTIC contains 11.2 hours of audio recorded by non-native speakers in English. This dataset includes audio with different accents and even mispronunciations. For this research, only audios with a Mexican Spanish accent were used. The idea is to contribute with the results of this research to encourage the development of tools to improve the pronunciation of the English language of Mexicans since pronunciation is a difficult skill to develop [22]. From this dataset, 4 hours of audios were taken and integrated into the LibriSpeech train and test sets, thus forming the dataset of audios used to evaluate the ASR model. The metric to assess performance was the CER metric, described in Section 2.2.

While learning a second language, assigning L1 as the native language and L2 as the target language is common. The reason for using both datasets is because there is some relationship between L1 and L2. Some phonemes share similarities in their pronunciation. However, while learning a second language, phoneme insertions, deletions, or substitutions are common. In [23], the relationship between L1 and L2 is described when adding or modifying a phoneme through the learning process. Because of this, both data sets were used.

Table 1. Different models trained

	Accuracy train set	**Accuracy validation set**	**CER**	**Data used**	**Expected epoch**	**Max epoch**
Model 1	8.3%	8.5%	77%	194 hours	20	20
Model 2	65%	52%	43%	388 hours	20	18
Model 3	66%	55%	43%	388 hours	60	20
Model 4	88%	79%	31%	776 hours	60	14
Model 5	**90%**	**86%**	**9.5%**	**1000 hours**	**100**	**100**

For the model training, a parameter configuration based on the *big model* implemented in [13] was used. The model was trained in approximately ten days on 1 NVIDIA 2600 Super with 8 GB GPU. In total, five models were trained, varying the data used and the training epochs. Table 1 shows the different models used.

As can be seen in Table 1, Model 5 obtained the best performance, reaching an accuracy of 90% in the train set and 86% in the validation set. On the other hand, the model reached 9.5% in the CER metric. It is important to mention that the lower the value of the CER metric, the more similar was the comparison between the canonical text and the inferred text by the ASR model.

Models 1 to 4 were trained with an early stopping parameter to avoid overfitting. However, the expected epoch was not reached in some models. The train set accuracy would tend to increase and decrease constantly, but after decreasing, the early stopping parameter was triggered. After deleting the early stopping parameter, the low accuracy would tend to grow up and never decrease back. Model 5 was the only one that reached the max epoch expected.

Table 2. Participants' characteristics

	Age	**Gender**	**Nationality**	**English level**	**Important characteristic**
Participant 1	27	Male	Mexican	Intermediate	Student
Participant 2	25	Female	Mexican	Advanced	English teacher
Participant 3	24	Male	Mexican	Intermediate	Student
Participant 4	24	Female	Mexican	Advanced	Native pronunciation
Participant 5	27	Male	Mexican	Intermediate	Student
Participant 6	35	Male	American	Advanced	Native pronunciation
Participant 7	34	Female	Mexican	Intermediate	Student
Participant 8	14	Male	Mexican	Beginner	Student
Participant 9	28	Male	American	Advanced	Native pronunciation
Participant 10	30	Female	Mexican	Intermediate	Student

After identifying the best model, the performance of the mispronunciation detection module was evaluated by a group of ten real users. Table 2 shows the characteristics of the participants. Each of the participants wrote and pronounced, one by one, a set of 10 phrases. The phrases contained words that are commonly mispronounced, according to a study conducted in [24]. Phrases

were selected from the *DeepL Translator* company *Linguee* website [25]. The phrases are shown below, highlighting words that are commonly difficult to pronounce:

1. *She has **enough** time.*
2. *It is a **pleasure** to meet you.*
3. *The first one is to **measure** the monetary value.*
4. *Their duties **should** be specified.*
5. *But I **think** it is hardly.*
6. *I cannot feed every **beast** of the field.*
7. *Classify **objects** based on their fragility.*
8. *The price per **hour** may vary.*
9. *It is a **slow** swimmer.*
10. *Create **elegant** online albums.*

Table 3 shows the results of the evaluation. The rows represent the participants, and the columns represent the phrases. The data correspond to the values of the CER metric obtained by each participant when pronouncing each phrase. The data shows how similar the recognized text and the input transcription are; a lower CER means better pronunciation. Participants 4 and 6 (dark gray) had the lowest CER values. That is, they achieved good pronunciation, so the string alignment algorithm found few discrepancies. It is worth mentioning that these people's characteristics can be seen that they have a native accent. On the other hand, participants 1 and 8 (light gray) obtained the highest CER values. It is worth mentioning that participant 8 has an initial level of English.

Table 3. Evaluation results

Participant	Phrase									
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1	16%	18%	30%	31%	08%	00%	34%	22%	25%	36%
2	00%	00%	11%	13%	17%	05%	44%	07%	05%	14%
3	00%	07%	09%	13%	00%	11%	07%	15%	10%	39%
4	00%	00%	02%	00%	00%	13%	27%	15%	10%	29%
5	00%	14%	15%	44%	00%	26%	22%	11%	25%	14%
6	00%	00%	07%	00%	08%	00%	10%	07%	00%	07%
7	21%	00%	13%	38%	00%	05%	20%	19%	30%	11%
8	63%	00%	43%	34%	17%	16%	24%	44%	35%	18%
9	00%	04%	30%	19%	00%	03%	22%	15%	20%	32%
10	00%	00%	02%	41%	00%	03%	15%	19%	40%	21%

There are two types of outputs in the mispronunciation detection module after the string-matching algorithm is applied. The first is when there are no differences between the strings, and the other is when there are differences. Below is an example of each of these types.

- Type 1:
  - Original transcription: *She has enough time (length 19)*
  - ASR text inference: *She has enough time (length 19)*
  - Output: `[ ]`

- Type 2:
  - Original transcription: *She has enough time (length 19)*
  - ASR text inference: *She fast and no attain (length 22)*
  - Output: `[('replace',4, 4), ('insert',7, 7), ('replace',8, 9), ('replace',10, 11), ('replace',11, 12), ('replace',12, 13), ('replace',13, 14), ('insert',15, 16), ('insert',15, 17), ('replace',16, 19), ('replace',17, 20), ('replace',18, 21)]`

The Type 1 example corresponds to participant 6. In this case, both sequences are the same length, and the output does not have any delete, insert, or substitution operations. The example of Type 2 corresponds to participant 8. In this case, the output has several insert and substitution operations. Also, the length between the strings is different.

As can be seen, the output consists of fragments that represent the operations performed. Each fragment is made up of three arguments. The first argument is the type of operation. The second argument is the index position of the first phrase. The third argument is the index position of the second phrase. The third argument is the second phrase index position. For example, the output `('replace',4, 4)` can be read as '*the character with index four in the first string was replaced with the character with index four in the second string*'.

5 Conclusion

In this work, a mispronunciation detection module integrated into the architecture of a chatbot was presented to achieve a natural interaction with the user. It receives as input the audio of the phrase in which the user wants to evaluate its pronunciation and receives its corresponding text transcription. The module outputs the mispronounced words.

It consists of two components, an ASR model, and a string alignment algorithm. The ASR converts the audio signal to text using a Transformer network, which was trained from the LibriSpeech and L2-ARCTIC data sets. To detect mispronounced words, the Levenshtein distance was used as a string alignment algorithm.

The ASR model reached an accuracy of 90% and a CER of 9.50%, showing promising results in detecting pronunciation errors. Additionally, it was also evaluated using a set of ten real users, resulting in good pronunciation in those users with native pronunciation, so the string alignment algorithm found few discrepancies. Users who obtained high CER values have an initial level of English.

Based on the results of the tests carried out, it can be concluded that the mispronunciation detection module achieved a good performance. It is because participants with a high level of English tended to obtain the best results with low CER values, while the rest achieved high CER values. Another factor that influenced the results was the intonation of the users at the time of the tests. Additionally, the results were positively affected when users spoke with a high tone of voice and made short pauses between words. Due to the above, it can be said that the mispronunciation detection module is helpful for users to practice until they improve their pronunciation.

As future work, we have in mind to focus on increasing the chatbot's knowledge base to achieve a greater degree of naturalness in the interaction with users.

References

1. Pedrycz, W., Martínez, L., Espin-Andrade, R. A., Rivera, G., Gómez, J. M (Eds.), "Computational Intelligence for Business Analytics," Springer, Cham, 2021. https://doi.org/10.1007/978-3-030-73819-8
2. Pazos-Rangel, R. A., Florencia-Juarez, R., Paredes-Valverde, M. A., Rivera, G. (Eds.), "Handbook of Research on Natural Language Processing and Smart Service Systems," IGI Global, 2021. https://doi.org/10.4018/978-1-7998-4730-4
3. Gilakjani, S. Ahmadi, and M. Ahmadi, "Why is Pronunciation so Difficult to Learn?," *English Lang. Teach.*, vol. 4, no. 3, 2011, https://doi.org/10.5539/elt.v4n3p74.
4. N. Haristiani, "Artificial Intelligence (AI) Chatbot as Language Learning Medium: An inquiry," *J. Phys. Conf. Ser.*, vol. 1387, no. 1, 2019, https://doi.org/10.1088/1742-6596/1387/1/012020.
5. K. Anwar and R. Husniah, "Evaluating Integrated Task Based Activities and Computer Assisted Language Learning (CALL)," *English Lang. Teach.*, vol. 9, no. 4, p. 119, 2016, https://doi.org/10.5539/elt.v9n4p119.
6. H. Wang, J. Xu, H. Ge, and Y. Wang, "Design and implementation of an English pronunciation scoring system for pupils based on DNN-HMM," *Proc. - 10th Int. Conf. Inf. Technol. Med. Educ.* ITME 2019, pp. 348–352, 2019, https://doi.org/10.1109/ITME.2019.00085.
7. Y. Zhao, J. Li, X. Wang, and Y. Li, "The Speechtransformer for Large-scale Mandarin Chinese Speech Recognition," *ICASSP, IEEE Int. Conf. Acoust. Speech Signal Process. - Proc.*, vol. 2019-May, pp. 7095–7099, 2019, https://doi.org/10.1109/ICASSP.2019.8682586.
8. S. Li, D. Raj, X. Lu, P. Shen, T. Kawahara, and H. Kawai, "Improving transformer-based speech recognition systems with compressed structure and speech attributes augmentation," *Proc. Annu. Conf. Int. Speech Commun. Assoc. INTERSPEECH*, vol. 2019-Septe, no. September, pp. 4400–4404, 2019, https://doi.org/10.21437/Interspeech.2019-2112.
9. Vaswani et al., "Attention is all you need," *Adv. Neural Inf. Process. Syst.*, vol. 2017-Decem, no. Nips, pp. 5999–6009, 2017, https://dl.acm.org/doi/10.5555/3295222.3295349.
10. S. Karita et al., "A Comparative Study on Transformer vs RNN in Speech Applications," *2019 IEEE Autom. Speech Recognit. Underst. Work. ASRU 2019 - Proc.*, no. December, pp. 449–456, 2019, https://doi.org/10.1109/ASRU46091.2019.9003750.

11. Z. Zhang, Y. Wang, and J. Yang, "Text-conditioned transformer for automatic pronunciation error detection," *arXiv*, 2020, https://doi.org/10.1016/j.specom.2021.04.004.
12. L. Zhang et al., "End-to-end automatic pronunciation error detection based on improved hybrid ctc/attention architecture," *Sensors* (Switzerland), vol. 20, no. 7, pp. 1–24, 2020, https://doi.org/10.3390/s20071809.
13. l. dong, s. xu, and b. xu, "speech-transformer: A no-recurrence sequence-to-sequence model for speech recognition," *ICASSP, IEEE Int. Conf. Acoust. Speech Signal Process. - Proc.*, pp. 5884–5888, 2018, [Online]. Available: http://150.162.46.34:8080/icassp2018/ICASSP18_USB/pdfs/0005884.pdf.
14. M. Espi, M. Fujimoto, K. Kinoshita, and T. Nakatani, "Exploiting spectro-temporal locality in deep learning based acoustic event detection," *Eurasip J. Audio, Speech, Music Process.*, vol. 2015, no. 1, 2015, https://doi.org/10.1186/s13636-015-0069-2.
15. W. K. Leung, X. Liu, and H. Meng, "CNN-RNN-CTC Based End-to-end Mispronunciation Detection and Diagnosis*," ICASSP, IEEE Int. Conf. Acoust. Speech Signal Process.* - Proc., vol. 2019-May, pp. 8132–8136, 2019, https://doi.org/10.1109/ICASSP.2019.8682654.
16. R. Haldar and D. Mukhopadhyay, "Levenshtein Distance Technique in Dictionary Lookup Methods: An Improved Approach," no. January 2011, 2011, [Online]. Available: http://arxiv.org/abs/1101.1232.
17. S. Kanters, C. Cucchiarini, and H. Strik, "The goodness of pronunciation algorithm: a detailed performance study.," *Speech Lang. Technol. Educ. -SLaTE*, no. 2, pp. 2–5, 2009, [Online]. Available: http://www.eee.bham.ac.uk/SLaTE2009/papers%5CSLaTE2009-33.pdf.
18. K. Li, X. Qian, and H. Meng, "Mispronunciation Detection and Diagnosis in L2 English Speech Using Multidistribution Deep Neural Networks," *IEEE/ACM Trans. Audio Speech Lang. Process.*, vol. 25, no. 1, pp. 193–207, 2017, https://doi.org/10.1109/TASLP.2016.2621675.
19. G. De Gasperis, I. Chiari, and N. Florio, AIML knowledge base construction from text corpora, vol. 427, no. June 2014. 2013, https://doi.org/10.1007/978-3-642-29694-9_12.
20. V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: An ASR corpus based on public domain audio books," *ICASSP, IEEE Int. Conf. Acoust. Speech Signal Process. - Proc.*, vol. 2015-Augus, pp. 5206–5210, 2015, https://doi.org/10.1109/ICASSP.2015.7178964.
21. G. Zhao, S. Sonsaat, A. Silpachai, and I. Lucic, "L2-ARCTIC : A Non-Native English Speech Corpus," *Proc. Annu. Conf. Int. Speech Commun. Assoc. INTERSPEECH*, no. September, pp. 2783–2787, 2018, https://doi.org/10.21437/Interspeech.2018-1110.
22. Borjian, "Learning English in Mexico: Perspectives From Mexican Teachers of English.," *CATESOL J.*, vol. 27, no. 1, pp. 163–173, 2015.
23. J. E. Flege, "Second Language Speech Learning: Theory, Findings, and Problems," *Speech Percept. Linguist. Exp. Issues Cross-Language Res.*, no. January 1995, pp. 233–277, 1995.
24. Komariah, "Problems in Pronouncing the English Sounds Faced by the Students of SMPN 2 Halong, Banjar," *J. English Lang. Pedagog.*, vol. 1, no. 2, pp. 1–10, 2019, https://doi.org/10.36597/jelp.v1i2.4127.
25. Schmidhofer and N. Mair, "Machine Translation in Translator Education," *CLINA*, vol. 4, no. 2, pp. 163–180, 2018, [Online]. Available: https://doi.org/10.14201/clina201842163180.