THIẾT KẾ CHATBOT SỬ DỤNG THUẬT TOÁN KHOẢNG CÁCH LEVENSHTEIN TRÊN RASPBERRY

A CHATBOT USING LEVENSHTEIN DISTANCE ALGORITHM FOR RASPBERRY BOARD

Trương Ngọc Sơn, Võ Thành Nhân, Lê Minh, Lê Minh Thành, Nguyễn Văn Phúc, Đặng Phước Hải Trang
Trường Đại học Sư phạm Kỹ thuật TP.HCM, Việt Nam

Ngày tòa soạn nhận bài 2/3/2020, ngày phản biện đánh giá 19/3/2020, ngày chấp nhận đăng 5/6/2020

TÓM TẮT

Trong bài viết này, tác giả giới thiệu về thuật toán khoảng cách Levenshtein và ứng dụng thuật toán tìm kiếm dựa trên khoảng cách Levenshtein để thiết kế chatbot, thay thế cho các chatbot sử dụng mạng nơ-ron nhân tạo. Chatbot sử dụng thuật toán Levenshtein đơn giản và hiệu quả khi thực thi trên máy tính nhúng Raspberry cho các robot. Các thông tin được lưu trong cơ sở dữ liệu làm cơ sở cho chatbot trả lời câu hỏi từ người dùng. Để so sánh thời gian đáp ứng giữa chatbot sử dụng thuật toán tìm kiếm và chatbot sử dụng mạng nơ-ron, tác giả thiết kế mạng nơ-ron tích chập và mạng Long-Short-Term Memrory được huấn luyện với cùng tập dữ liệu. Các mô đun được thực thi trên hệ thống nhúng Raspberry. Kết quả thực nghiệm cho thấy, chatbot sử dụng thuật toán tìm kiếm dựa trên khoảng cách Levenshtein có thời gian đáp ứng nhanh với cùng độ chính xác cho các câu hỏi có trong cơ sở dữ liệu. Kiểm tra trên 10 câu hỏi ngẫu nhiên, chatbot sử dụng thuật toán Levenshtein cho kết quả nhanh hơn 15 lần so với dùng mạng CNN và 75 lần so với dùng mạng LSTM. Chatbot sử dụng giải thuật Levenshtein là một ứng dụng tối ưu nhằm làm giảm tối đa tài nguyên cho các máy tính nhúng có kiến trúc thấp được sử dụng trong các robot di động.

***Từ khóa**: Chatbot; Khoảng cách Levenshtein; Thuật toán tìm kiếm; Mạng nơ-ron; Mạng nơ-ron tích chập.*

ABSTRACT

In this paper, we present a chatbot based on the Levenshtein Distance for low-cost embedded systems. The state-of-the art chatbots are based on deep neural networks, however, such chatbots cannot be deployed on the low-cost embedded system, such as Raspberry board for mobile robots. Chatbot based on Levenshtein Distance requires fewer resources and can be deployed on low-cost embedded systems efficiently. The Levenshtein distance represents the similarity between the two strings. The similarity between the input question and all the stored questions in the database are measured. A winner is a stored question that is the best similar to the input question. Having recognized the question, chatbot can decide the output by querying from the database. Chatbot using (a) search algorithm based on Levenshtein distance is faster by 15 times and 75 times than the Convolutional Neural Network and the LSTM network. The chatbot based on Levenshtein Distance is suitable to be deployed on the low-cost embedded systems for mobile robots.

***Keywords:** Chatbot; Levenshtein distance; Search algorithm; Neural network; Convolutional neural networks;*

1. GIỚI THIỆU

Chatbot là một phần mềm hỗ trợ giao tiếp giữa người và máy [1]–[3]. Các chatbot được cài đặt một lượng thông tin và có khả năng đưa ra câu trả lời hoặc các quyết định khi người dùng truy vấn [4]. Với chatbot, người dùng có thể giao tiếp với các hệ thống bằng ngôn ngữ tự nhiên (natural language) mà không cần phải sử dụng các ngôn ngữ lập

trình phức tạp. Chatbot đóng vai trò như một trợ lý ảo, có thể được sử dụng trong nhiều lĩnh vực khác nhau như kinh doanh, y tế và cả trong giáo dục [5]–[8]. Các chatbot hiện nay được xây dựng chủ yếu sử dụng các mạng nơ-ron nhân tạo, trong đó chủ yếu là các mạng học sâu (deep leanring) [9], [10]. Mặc dù các mạng học sâu cho kết quả khá tốt nhưng nó chỉ thực sự phát huy hiệu quả khi được thực thi trên các máy tính có cấu hình mạnh bởi vì các mạng học sâu dựa trên một số lượng lớn các phép toán, xử lý, và các thông số mô hình. Khi triển khai các mạng học sâu cho các thiết bị nhỏ như các robot chúng ta gặp nhiều khó khăn. Các hệ thống nhúng ngày nay mặc dù có khả năng thực thi các mạng học sâu, tuy nhiên nó luôn bị hạn chế về tốc độ và khả năng lưu trữ. Trong các ứng robot cần phần cứng nhỏ gọn nhưng vẫn đáp ứng yêu cầu về độ chính xác và tốc độ đáp ứng. Đối với các tập dữ liệu nhỏ, chúng ta có thể sử dụng các thuật toán so sánh để thiết kế chatbot thay cho các mạng học sâu nhằm tăng độ chính xác và thời gian đáp ứng. Các thuật toán so sánh sẽ đánh giá mức độ giống nhau giữa câu hỏi người dùng đưa ra và các câu hỏi có trong cơ sở dữ liệu (CSDL), từ đó đưa ra câu trả lời phù hợp được cài đặt trước. Và một trong những thuật toán so sánh thường được sử dụng hiện nay là thuật toán tính khoảng cách Levenshtein. Thuật toán khoảng cách Levenshtein có thể áp dụng để so sánh cho hai từ hoặc câu không cùng độ dài. Chatbot được xây dựng dựa trên thuật toán Levenshtein và được triển khai trên máy tính nhúng Raspberry Pi cho robot.

2. ỨNG DỤNG THUẬT TOÁN KHOẢNG CÁCH LEVENSHTEIN THIẾT KẾ CHATBOT

Thuật toán khoảng cách Levenshtein là một phương pháp để đánh giá mức độ giống nhau giữa hai chuỗi [11]–[13]. Khoảng cách Levenshtein giữa hai từ hoặc câu là tính toán số thay đổi nhỏ nhất để chuyển đổi từ hoặc câu này thành từ hoặc câu còn lại, dựa trên ba phép biến đổi là: xóa, thêm, thay từng thành phần trong từ hoặc câu [13]. Thuật toán khoảng cách Levenshtein cũng còn được biết đến với tên gọi "khoảng cách chỉnh sửa" [12].

Công thức toán học của thuật toán khoảng cách Levenshtein giữa hai chuỗi a và b [14]–[16]:

$$lev_{a,b}(i,j)\begin{cases}\max(i,j) \; if \; \min(i,j)=0 \\ \min\begin{cases} lev_{a,b}(i-1,j)+1 \\ lev_{a,b}(i,j-1)+1 \\ lev_{a,b}(i-1,j-1)+1_{(a_i\neq b_i)} \end{cases} \quad otherwise\end{cases} \tag{1}$$

Trong đó, $1_{(a_i\neq b_j)}$ bằng 0 khi $a_i \neq b_j$và bằng 1 trong các trường hợp còn lại. $lev_{a,b}(i,j)$ là khoảng cách giữa i ký tự đầu tiên của chuỗi a và j ký tự đầu tiên của chuỗi b.

Chatbot được trình bày trong bài báo này là một ứng dụng trong robot trợ lý giảng dạy. Chatbot được cài đặt các câu hỏi và câu trả lời liên quan đến một môn học cụ thể. Chúng tôi chọn môn Ngôn ngữ lập trình C với các kiến thức cơ bản làm nội dung cài đặt cho chatbot. Tập dữ liệu được tạo với 100 câu hỏi và 100 câu trả lời tương ứng. Tập dữ liệu này được dùng để cài đặt cho chatbot. Tập dữ liệu này cũng được sử dụng để huấn luyện cho mạng nơ-ron tích chập và mạng nơ-ron hồi quy để so sánh kết quả. Chúng ta có thể gọi tập dữ liệu này là tập huấn luyện. Đối với tập dữ liệu phục vụ cho việc đánh giá mô hình, chúng tôi cũng tạo một tập dữ liệu bao gồm 100 câu hỏi và 100 câu trả lời tương ứng. Tập dữ liệu này được lấy từ tập dữ liệu huấn luyện nhưng lỗi ngẫu nhiên được thêm vào. Cụ thể chúng ta lấy 100 câu hỏi từ tập huấn luyện và cho phép sai ngẫu nhiên với số từ sai nhỏ hơn hoặc bằng 2. Chúng ta gọi tập dữ liệu này là tập kiểm tra. Tập huấn luyện được mô tả trong bảng 1.

Bảng 1. *Tập huấn luyện*

Câu hỏi: Biến là gì
Câu trả lời: Biến tượng trưng cho một ô nhớ để lưu trữ
Câu hỏi: Kiểu dữ liệu số nguyên là kiểu nào
Câu Trả lời: int, long, unsigned int, và unsigned long
Câu hỏi: while và do while khác nhau thế nào
Câu Trả lời: do while thực hiện ít nhất 1 lần, while thì có thể không

Độ dài lớn nhất cho các câu hỏi là 15 từ không bao gồm các ký tự đặt biệt như dấu chấm hỏi. Các câu hỏi và câu trả lời được lưu trong cơ sở dữ liệu dưới dạng các tập tin định dạng JSON (JavaScript Object Notation) với từng cặp câu hỏi và câu trả lời. Các câu hỏi và câu trả lời được phân biệt bằng ký tự đầu tiên là "Q" cho câu hỏi và "A" cho câu trả lời. Với thư viện tiếng Việt, ta phải chuyển đổi các ký tự tiếng Việt sang bảng mã để chương trình có thể dễ dàng nhận biết. Ví dụ một cặp câu hỏi và câu trả lời được thể hiện trong cơ sở dữ liệu như bảng 2.

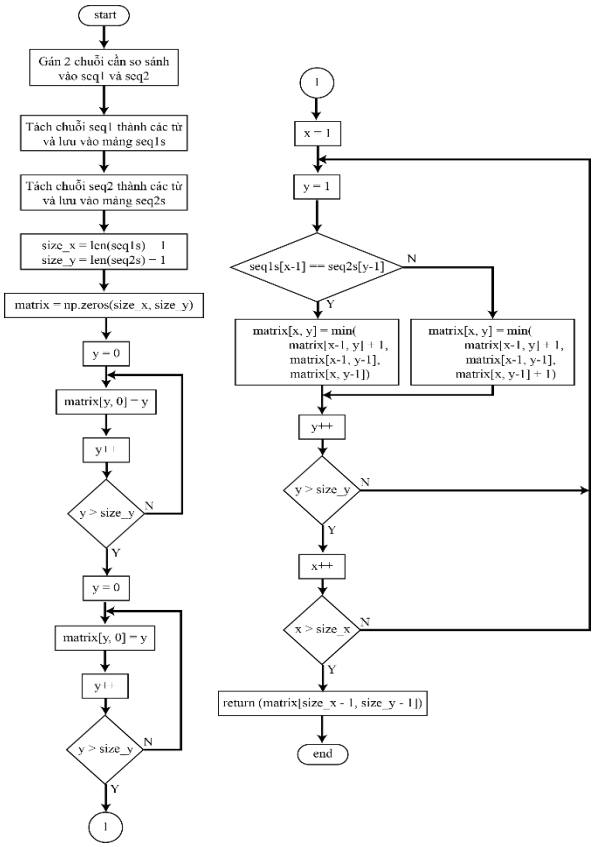
Bảng 2. *Định dạng câu hỏi và câu trả lời trong cơ sở dữ liệu*

Câu hỏi và câu trả lời bằng tiếng Việt	Câu hỏi và câu trả lời được mã hóa
Biến là gì	"Q": " bi\u1ebfn l\u00e0 g\u00ec "
Biến tượng trưng cho một ô nhớ để lưu trữ	"A": " Bi\u1ebfn t\u01b0\u1ee3ng tr\u01b0ng cho m\u1ed9t \u00f4 nh\u1edb \u0111\u1ec3 l\u01b0u tr\u1eef "

Trong đó, câu hỏi là "*Biến là gì*" và câu trả lời là "*Biến tượng trưng cho một ô nhớ để lưu trữ*". Theo đó, ký tự "ế" sẽ được lưu dưới dạng mã hex là \u1ebf, ký tự "à" sẽ là \u00e0, và tương tự cho các ký tự có dấu tiếng Việt còn lại.

Để tìm kiếm câu trả lời cho một câu hỏi trong cơ sở dữ liệu (CSDL), ta sử dụng thuật toán tính khoảng cách Levenshtein để so sánh mức độ giống nhau giữa câu hỏi được đưa ra và các câu hỏi có trong CSDL. Nếu giá trị khoảng cách Levenshtein nhỏ hơn giá trị được cài đặt trước, trong bài báo này giá trị này bằng 2, thì hai chuỗi đó được xem là tương đồng nhau. Trong ứng dụng này chúng ta lập trình tìm khoảng cách Levenshtein của 2 chuỗi ở mức từ (word) thay vì ở mức ký tự (character). Nếu 2 chuỗi giống nhau, giá trị Levenshtein sẽ là 0. Ngược lại, giá trị đo được cho biết số từ khác nhau giữa 2 chuỗi. Bằng thực nghiệm trên tập dữ liệu nhỏ, chúng tôi chọn giá trị ngưỡng là 2. Với giá trị này, chương trình vẫn đảm bảo độ chính xác khi có 2 từ trong câu hỏi không giống với câu hỏi trong tập huấn luyện.

Lưu đồ chương trình tính khoảng cách giữa 2 chuỗi được trình bày trong hình 1. Trước hết chương trình sẽ điền giá trị chỉ mục vào hàng đầu tiên và cột đầu tiên. Sau đó sẽ lần lượt tính giá trị các phần tử trong ma trận sử dụng công thức Levenshtein (1). Kết thúc quá trình chúng ta thu được giá trị thể hiện sự tương đồng của 2 chuỗi.

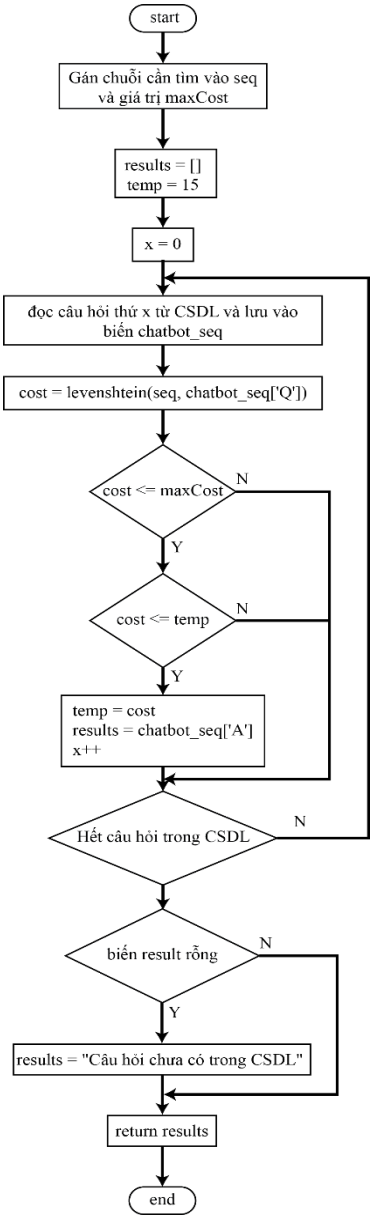


Hình 1. *Lưu đồ giải thuật tính khoảng cách Levenshtein giữa hai chuỗi*

Chương trình được viết bằng ngôn ngữ Python và thực thi trên hệ thống nhúng Raspberry Pi. Chuỗi đầu vào được lấy từ mô đun chuyển từ giọng nói sang văn bản và chuỗi còn lại được lấy từ cơ sở dữ liệu.

Lưu đồ chương trình chính được mô tả trong hình 2. Câu hỏi nhận được từ mô đun chuyển lời nói sang văn bản được so sánh với tất cả các câu hỏi trong cơ sở dữ liệu. Lưu đồ bài toán là giải thuật tìm lần so sánh có giá trị nhỏ nhất và nhỏ hơn giá trị ngưỡng. Giá trị

ngưỡng được thiết lập trong biến MaxCost. Giá trị cost ban đầu được khởi tạo là giá trị lớn nhất trong trường hợp 2 câu khác nhau hoàn toàn. Trong quá trình duyệt hết cơ sở dữ liệu, giá trị cost sẽ được cập nhật và câu hỏi có giá trị cost nhỏ nhất sẽ được lưu lại.



Hình 2. *Lưu đồ giải thuật tìm kiếm dựa trên khoảng cách Levenshtein*

3. KẾT QUẢ THỰC NGHIỆM VÀ THẢO LUẬN

Mục đích sử dụng thuật toán khoảng cách Levenshtein thay cho các mạng nơ-ron là hướng đến một hệ thống nhỏ, gọn, có khả năng thực thi tốt trên các hệ thống nhúng có cấu hình phần cứng thấp cho các thiết kế robot di động. Để so sánh thời gian đáp ứng của thuật toán Levenshtein với các mạng học sâu, tác giả thực thi 3 mô đun: thuật toán tìm câu trả lời dựa trên Levenshtein, mạng nơ-ron tích chập (CNN), và mạng nơ-ron hồi qui được cải thiện (Long-short-term memory). Mạng nơ-ron CNN và LSTM được thiết kế sử dụng thư viện Keras. Các module được thực thi trên phần cứng nhúng Raspberry Pi để đo tốc độ đáp ứng khi đưa 10 câu hỏi ngõ vào. Hệ thống nhúng Raspberry Pi 3 sử dụng bộ xử lý ARM Cortex-A53 với bộ nhớ RAM có dung lượng 1GB thích hợp cho các ứng dụng di động như robot và các hệ thống tự động điều khiển [17], [18].

Chatbot sử dụng giải thuật khoảng cách Levenshtein được so sánh với chatbot sử dụng mạng CNN và mạng LSTM về thời gian đáp ứng. Mạng CNN được thiết kế bao gồm 1 lớp Convolution với hàm kích hoạt ReLU, 1 lớp Maxpooling, 1 lớp kết nối đầy đủ với 256 nơ-ron và 1 lớp ngõ ra với 100 nơ-ron để phân lớp. Các câu ngõ vào được mã hóa sử dụng mã one-hot cho từng ký tự. Mỗi ký tự được biểu diễn dưới dạng một vector. Các vector của một câu tạo thành một ma trận 2 chiều cho ngõ vào của mạng CNN.

Đối với mạng LSTM, sử dụng kiến trúc Encoder-Decoder ở mức ký tự. Ngõ vào và ngõ ra của mạng Encoder-Decoder cũng sử dụng mã one-hot.

Đối với tập kiểm tra được tạo ra từ tập huấn luyện, trong đó các câu sai ngẫu nhiên 0, 1, hoặc 2 từ. Độ chính xác của 3 mô hình được liệt kê trong bảng 3.

Bảng 3. *So sánh độ chính xác của chatbot sử dụng thuật toán Levenshtein, mạng CNN và mạng mạng LSTM.*

Levenshtein	Mạng CNN	LSTM
100%	99%	88%

Đặc điểm của các mạng tích chập là cho kết quả tốt khi ngõ vào biến thiên nhẹ bởi các lớp maxpooling chỉ lấy kết quả lớn nhất trong cửa sổ mà không quan tâm vị trí của phần tử lớn nhất. Trong khi đó mạng LSTM dựa trên dự đoán các ký tự tiếp theo dựa trên các ký tự

hiện tại và trước đó lại cho ra sai số lớn khi ngõ vào thay đổi. Kỹ thuật so sánh dùng Levenshtein khá đơn giản và trong trường hợp sai số nhỏ có thể đảm bảo được độ chính xác khá tốt. Với tập nhỏ và sai số đặc dưới mức ngưỡng thiết lập cho giải thuật thì chúng ta vẫn đạt tỷ lệ 100%. Kết quả độ chính xác để kiểm chứng rằng đối với tập dữ liệu nhỏ và cho ứng dụng robot trả lời một số câu có trong kịch bản trước thì có thể sử dụng giải thuật Levenstein thay cho các cấu trúc phức tạp như CNN và LSTM. Tỷ lệ nhận dạng của các mạng phụ thuộc nhiều yếu tố như số nơ-ron, số lớp mạng, các thông số cài đặt mô hình.

Thời gian đáp ứng với mẫu thử là 10 câu hỏi trong có trong CSDL của chatbot xây dựng với thuật toán tìm kiếm dựa trên khoảng cách Levenshtein so với chatbot được xây dựng với mạng nơ-ron CNN và LSTM được thể hiện trọng bảng 4.

Bảng 4. *So sánh thời gian đáp ứng của thuật toán so sánh theo khoảng cách Levenshtein, mạng CNN và LSTM*

	Mô hình và thời gian đáp ứng (s)		
Mẫu thử	Levenshtein	Mạng CNN	LSTM
1	0.001	0.303	1.3521
2	0.003	0.301	1.345
3	0.005	0.301	1.342
4	0.007	0.310	1.534
5	0.028	0.309	1.432
6	0.034	0.302	1.476
7	0.024	0.300	1.421
8	0.026	0.302	1.486
9	0.029	0.300	1.422
10	0.034	0.305	1.496

Thời gian gian đáp ứng của chatbot được xây dựng với thuật toán tìm kiếm dựa trên khoảng cách Levenshtein nhanh hơn so với chatbot xây dựng bằng mạng CNN và mạng LSTM, khi có cùng số câu hỏi trong CSDL. Với những câu hỏi càng có nhiều từ trong câu thì chatbot sẽ càng mất nhiều thời gian để so sánh và tìm ra câu trả lời thích hợp. Tính trung bình, chatbot sử dụng thuật toán tìm kiếm khoảng cách Levenshtein nhanh hơn 15 lần so với mạng CNN và 75 lần so với mạng LSTM.

Các chatbot hiện nay đa phần dựa vào các mạng học sâu và xử lý ngôn ngữ tự nhiên, trong đó phổ biến là kiến trúc mạng Long-Short-Term Memory. Kết quả của các chatbot sử dụng các mạng nơ-ron cho kết quả tốt hơn, tìm kiếm chính xác và thông minh hơn. Tuy nhiên để thực thi các mạng nơ-ron CNN và mạng LSTM cần nhiều tài nguyên và thích hợp trên các máy tính có tốc độ xử lý cao và dung lượng lưu trữ lớn. Các mạng CNN và LSTM khi được triển khai dưới các hệ thống nhúng có tài nguyên giới hạn sẽ không hiệu quả. Trong khi đó, thực thi chatbot dựa trên giải thuật tìm kiếm Levenshtein đơn giản, sử dụng ít tài nguyên, thích hợp với việc triển khai trên kiến trúc hệ thống nhúng nhỏ như Raspberry Pi. Như vậy việc xây dựng kỹ thuật tìm kiếm câu trả lời với Levenshtein đơn giản, tài nguyên còn lại của hệ thống nhúng Raspberry có thể sử dụng cho các mục đích khác cho robot như xử lý ảnh ngõ vào, nhận dạng tiếng nói, và quyết định, điều khiển ngõ ra.

4. KẾT LUẬN

Chatbot được xây dựng trên thuật toán tìm kiếm bằng khoảng cách Levenshtein có thời gian phản hồi nhanh hơn 15 lần so với mạng CNN và 75 lần so với LSTM. Các câu hỏi và câu trả lời được thiết kế trước và lưu vào cơ sở dữ liệu. Thuật toán tính khoảng cách Levenshtein đơn giản, sử dung ít tài nguyên và hiệu quả khi được thực thi trên hệ thống nhúng Raspberry phục vụ cho việc thiết kế các robot di động. Chatbot sử dụng khoảng cách Levenshtein là một mô đun trong thiết kế robot di động có khả năng giao tiếp với con người bằng giọng nói.

LỜI CẢM ƠN

Kết quả nghiên cứu và ứng dụng là sản phẩm của Đề tài Nghiên cứu Khoa học Cấp Bộ, mã số **B2019-SPK-05,** được hỗ trợ bởi Bộ Giáo dục và Đào tạo và chủ trì bởi Trường Đại học Sư phạm Kỹ thuật TP.HCM.

TÀI LIỆU THAM KHẢO

[1] B. A. Shawar and E. Atwell, "Different measurement metrics to evaluate a chatbot system," in Proceedings of the workshop on bridging the gap: Academic and industrial research in dialog technologies, 2007, pp. 89–96.

[2] A. M. Rahman, A. Al Mamun, and A. Islam, "Programming challenges of chatbot: Current and future prospective," in 2017 IEEE Region 10 Humanitarian Technology Conference (R10-HTC), 2017, pp. 75–78.

[3] J.-C. Gu, Z.-H. Ling, and Q. Liu, "Utterance-to-Utterance Interactive Matching Network for Multi-Turn Response Selection in Retrieval-Based Chatbots," IEEE/ACM Trans. Audio, Speech, Lang. Process., vol. 28, pp. 369–379, 2020.

[4] B. Setiaji and F. W. Wibowo, "Chatbot Using a Knowledge in Database: Human-to-Machine Conversation Modeling," in 2016 7th International Conference on Intelligent Systems, Modelling and Simulation (ISMS), 2016, pp. 72–77.

[5] G. M. D'silva, S. Thakare, S. More, and J. Kuriakose, "Real world smart chatbot for customer care using a software as a service (SaaS) architecture," in 2017 International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC), 2017, pp. 658–664.

[6] M. Bates, "Health Care Chatbots Are Here to Help," IEEE Pulse, vol. 10, no. 3, pp. 12–14, May 2019.

[7] D. Madhu, C. J. N. Jain, E. Sebastain, S. Shaji, and A. Ajayakumar, "A novel approach for medical assistance using trained chatbot," in 2017 International Conference on Inventive Communication and Computational Technologies (ICICCT), 2017, pp. 243–246.

[8] A. Mondal, M. Dey, D. Das, S. Nagpal, and K. Garda, "Chatbot: An automated conversation system for the educational domain," in 2018 International Joint Symposium on Artificial Intelligence and Natural Language Processing (iSAI-NLP), 2018, pp. 1–5.

[9] M. Nuruzzaman and O. K. Hussain, "A Survey on Chatbot Implementation in Customer Service Industry through Deep Neural Networks," in 2018 IEEE 15th International Conference on e-Business Engineering (ICEBE), 2018, pp. 54–61.

[10] H. Honda and M. Hagiwara, "Question Answering Systems With Deep Learning-Based Symbolic Processing," IEEE Access, vol. 7, pp. 152368–152378, 2019.

[11] A. Ene and A. Ene, "An application of Levenshtein algorithm in vocabulary learning," in 2017 9th International Conference on Electronics, Computers and Artificial Intelligence (ECAI), 2017, pp. 1–4.

[12] G. Navarro, "A guided tour to approximate string matching," ACM Comput. Surv., vol. 33, no. 1, pp. 31–88, Mar. 2001.

[13] V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions, and reversals," in Soviet physics doklady, 1966, vol. 10, no. 8, pp. 707–710.

[14] A. Andoni, R. Krauthgamer, and K. Onak, "Polylogarithmic Approximation for Edit Distance and the Asymmetric Query Complexity," in Proceedings - Annual IEEE Symposium on Foundations of Computer Science, FOCS, 2010, pp. 244–252.

[15] D. Q. Thang and P. T. Huy, "Determining restricted Damerau-Levenshtein editdistance of two languages by extended automata," in 2010 IEEE-RIVF International Conference on Computing and Communication Technologies: Research, Innovation and Vision for the Future, RIVF 2010, 2010.

[16] K. U. Schulz and S. Mihov, "Fast string correction with Levenshtein automata," Int. J. Doc. Anal. Recognit., vol. 5, no. 1, pp. 67–85, Nov. 2002.

[17] X. Wen and Y. Wang, "Design of smart home environment monitoring system based on raspberry Pi" 2018 Chinese Control And Decision Conference (CCDC), Shenyang, 2018, pp. 4259-4263.

[18] S. Jain, A. Vaibhav and L. Goyal, "Raspberry Pi based interactive home automation system through E-mail," 2014 International Conference on Reliability Optimization and Information Technology (ICROIT), Faridabad, 2014, pp. 277-280.

Tác giả chịu trách nhiệm bài viết:
TS. Trương Ngọc Sơn
Trường Đại học Sư phạm Kỹ thuật TP.HCM
Email: sontn@hcmute.edu.vn