Journal of Artificial Intelligence and Technology, 2021, 1, 121-130
https://doi.org/10.37965/jait.2021.0003

ISTP
RESEARCH ARTICLE

Intelligent Voice Instructor-Assistant System for Collaborative and Interactive Classes

Matthew Baker,[1] Xiaohui Hu,[2] Gennaro De Luca,[3] and Yinong Chen[1]
[1]School of Computing, Informatics and Decision Systems Engineering, Arizona State University, Tempe, AZ 85281 USA
[2]School of Physics and Telecommunication Engineering, South China Normal University, Guangzhou, Guangdong 510006 China
[3]Polytechnic School, Arizona State University, Mesa, AZ 85212 USA

(Received 15 July 2020; Revised 14 March 2021; Accepted 21 March 2021; Published online 24 March 2021)

Abstract: College classes are becoming increasingly large. A critical component in scaling class size is the collaboration and interactions among instructors, teaching assistants, and students. We develop a prototype of an intelligent voice instructor-assistant system for supporting large classes, in which Amazon Web Services, Alexa Voice Services, and self-developed services are used. It uses a scraping service for reading the questions and answers from the past and current course discussion boards, organizes the questions in JavaScript object notation format, and stores them in the database, which can be accessed by Amazon web services Alexa skills. When a voice question from a student comes, Alexa is used for translating the voice sentence into texts. Then, Siamese deep long short-term memory model is introduced to calculate the similarity between the question asked and the questions in the database to find the best-matched answer. Questions with no match will be sent to the instructor, and instructor's answer will be added into the database. Experiments show that the implemented model achieves promising results that can lead to a practical system. Intelligent voice instructor-assistant system starts with a small set of questions. It can grow through learning and improving when more and more questions are asked and answered.

Key words: natural language processing; voice processing; machine learning; Long short-term memory (LSTM) network; questions and answers

I. INTRODUCTION

College classes are becoming online and becoming very large. For example, Arizona State University (ASU) computer science and engineering senior courses CSE445 (Distributed Software Development) and CSE446 (Software Integration and Engineering) enroll over 100 in-person students in each course. Both courses also have an in-person and an online section. The lower-division courses, such as CSE240 (Introduction to Programming Languages), have over 400 in-person students in each semester, plus even bigger online sections. At least a thousand questions are asked in each course in each semester. Each course has one instructor and a number of graduate teaching assistants (TAs) proportional to the number of students, which allows the classes to be scaled up.

A number of major problems emerge in the current system through dealing with large classes, which can affect the teaching quality:

- The current discussion boards we use in blackboard and canvas learning systems are not friendly for searching questions and answers.
- It is much easier for a student to type a new question than finding the same question in the discussion board that has been answered by the instructor, or by a TA, or by another student, resulting in many redundant questions and answers, which in turn demotivating students to read the discussion board effectively.
- Questions and answers of the same course in the previous semesters cannot be reused, as they are not accessible for current students.
- Instructor and TAs are responsible for answering student questions in a timely manner. As the class and the number of questions become bigger, the portion of questions answered by the instructor becomes smaller.
- TAs are assigned on semester basis, and they may or may not answer all the questions correctly or accurately, forcing the instructor to answer most questions.

Question answering is a topic in natural language processing, where a software system accepts a user question as the input, processes it, and returns what it believes to be the answer to that question. Question answering is categorized into two major paradigms: information-retrieval, which "relies on the vast quantities of textual information on the web or in collections like PubMed," and knowledge-based systems, which create a semantic representation of the query before attempting to answer [1]. Many general techniques and systems have been developed. Stanford question answering dataset (SQuAD) is a comprehensive system that consists of 100,000 + questions posed by crowd workers on a set of Wikipedia articles, where the answer to each question is a segment of text from the corresponding reading passage [1]. SQuAD is available for open access at https://rajpurkar.github.io/SQuAD-explorer/. A key component of the question answering system is a knowledge base that

Corresponding author: Xiaohui Hu (e-mail: huxh@scnu.edu.cn).

stores questions and their answers and facilitates searching and matching of the questions [2]. Ontology is often used for presenting knowledge bases for easy searching, matching, and reasoning [3], [4]. Many studies and commercial systems have been developed, such as Amazon web services (AWS) Alexa [5], Apple Siri [6], Google assistant [7], and Microsoft cortana [8].

Rather than focusing on developing a general question answering system, the proposed intelligent voice instructor-assistant system (IVIAS) focuses on a question answering system in a closed domain for specific courses. A quality course requires good answers to student questions. For large classes, we depend on the TA's skills. However, even good TAs often cannot answer student's questions in the way the instructors want. In many cases, their answers are inconsistent to what the instructors teach. Therefore, many instructors do not use TAs for answering student questions, except those questions that are related to the assignments that the TAs have prepared. As a result, the instructors have to answer many questions, which are the same from semester to semester and even in the same semester. This instructor-assistant system developed is intended for imitating the skills of the instructor for the same course across multiple semesters. The skills developed for the system are based on the instructor's answers to the questions in the past on the discussion boards.

Due to the source of the skill's data being quantities of textual information in the form of discussion board posts, we chose to focus the investigation on information retrieval. The purpose of this project is to achieve this balance by creating a skill that allows students to ask their questions about their courses and receive reasonably accurate answers drawn from their class discussion board. In a different application scenario, the IVIAS has been used for the communication between a vision-impaired person and a robot guide dog system [9], [10], where the relevant questions and answers have been prepared in the system.

The rest of the paper is organized as follows. Section II describes the related work. Section III introduces the IVIAS. Section IV shows the experiments, and Section V draws the conclusions.

II. RELATED WORK

The third-generation artificial intelligence (AI) today is based on a number of key technologies, including big data processing, cloud computing, service-oriented computing, natural language processing, and voice-based human–machine interface [3]. Fig. 1 shows the cutting-edge technologies this project is based on.

The system uses a number of cutting-edge technologies, including Internet of things (IoT), Internet of intelligent things (IoIT), artificial intelligence of things (AIoT), and physical artificial intelligence (PAI), that support the human–machine interfacing [14]. The system is implemented in web and cloud environment that can take advantages of big data, AI, and machine learning resources [3], [11]–[13]. PAI is the latest concept that differentiates Digital Artificial Intelligence (DAI) from PAI. DAI focuses on the part of AI that is based on cloud computing and big data processing, while PAI focuses on the interaction with the following domains of knowledge and their integration:

- Computer Science: programming, machine learning, data science, networking, etc.
- Biology: physiology, tissue engineering, biomechanics, phytology, etc.
- Chemistry: Chemical synthesis, analytical chemistry, organic chemistry, biochemistry, etc.
- Material Science: polymer science, composite materials, characterization, processing, etc.
- Mechanical Engineering: robotics, metratronics, manufacturing, design, etc.

As a PAI system, a voice-activated question answering system contains many key components, including question analysis, document retrieval, answer matching, and other modules. The answer matching module is an important component that is supported by natural language processing, service invocation, machine learning, and big data processing. For a given question, how well an answer matches the question is directly related to the quality and performance of the entire system. Essentially, answer matching can be regarded as a classification task. The probability of each answer in the candidate answer sequence is calculated, and then the answer with the highest probability is chosen as the correct answer [15].

As a part of the third-generation AI, the in-depth learning technology represented by convolutional neural network (CNN) is transforming the intelligent question answering system. Severyn *et al.* [16] showed that the multilayer CNN is effective in learning the vector representation of questions and answers. In their model, questions and answers are used as the input of the model, respectively; Yu *et al.* [17] proposed a deep CNN for sentences modeling and the answer searching of in the dataset; Dong L [18] built a CNN for a better vector model.

Iyyer [19] modeled the combined texts via recurrent neural network (RNN) and applied it to quiz bowl answering tasks. Wang *et al.* [20] used bi-directional long short-term memory (Bi-LSTM) network to learn the eigenvector representation of question–answer pairs through the context information of question–answer texts. In 2016, Tan *et al.* published "LSTM-based deep learning models for non-factoid answer selection," which elaborated Long Short-Term Memory (LSTM) algorithm in question answering system in detail [21], and Shi *et al.* [22] combined Bi-LSTM and CNN with attention mechanism for question categorization.

All the aforementioned studies and experiments show that with the help of a large number of corpus, the deep learning model can actively learn the potential syntactic and semantic features in the problem, so that the system can better understand the question and choose a good answer with flexibility and robustness. Although these studies have made some breakthroughs in answer selection in question answering systems, due to the complexity of the natural language, such as synonyms, antonyms, etc., used in question answering texts and the syntax, some key information in question answering text sequences have to be ignored. This paper attempt to implement a Siamese Bi-LSTM network based on attention mechanism [23] to solve this problem.

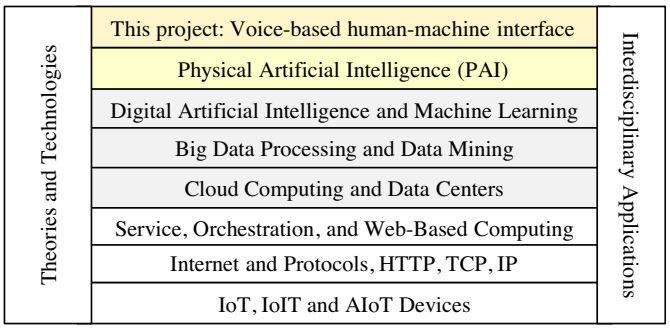

Fig. 1. Voice-based human–interface and related supporting technologies.

III. IVIAS

This section presents the IVIAS that we developed for supporting large classes, which is based on Amazon voice services and Alexa skills, as shown in Fig. 2.

Voice input is taken by device, such as Echo, Echo Dot, and Echo Show. An Alexa application can run on iOS or Android or any Windows or Unix-based operating system. Input is uploaded to Alexa Voice Service and Alexa Skills Kit Lambda Trigger takes input to AWS Lambda, which runs our IVIAS code and generates the response based on its database of questions and answers and machine learning model. AWS Lambda application will take the input from Alexa Skill Kit, then parse out irrelevant input such as "why," "does," etc. Relevant input such as subject concept terms will be taken then the relationship between these terms will be determined. After the search is done, results will be displayed or read back to the user. If no relevant answer is available in the database, then question will be sent to the unanswered question bank for instructor to answer.

A. SYSTEM ARCHITECTURE

The architecture of the proposed IVIAS question–answer system is given in Fig. 3. IVIAS offers types of interfaces for students, instructors, and administrators, respectively.

IVIAS consists of three major functional units:

1) Corpus management: As an experimental system, the corpus management module allows the questions and answers to be added, deleted, edited, and imported dynamically.
2) The question answering model: It is the core of the system, in which an improved siamese deep LSTM (SD-LSTM) model is introduced to compare the similarity among the preprocessed questions. We also compared this model with some other information retrieval methods.
3) The question answering services: They provide access to the voice inputs of the users. Alexa interaction model is applied to translate the voice into the text, and Alexa skill functions provide database application programming interface (API) and POST Provider API for developers to program the access to the database and the user interface.

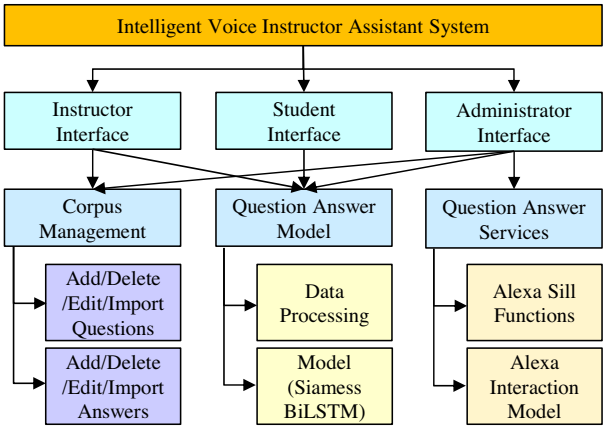

Fig. 3. IVIA system framework.

B. QUESTION ANSWERING SERVICES

IVIAS utilizes numerous services offered by AWS to receive, process, and answer questions. Fig. 4 shows the major modules and the data flow of the system, with different blocks representing the AWS service used. The Alexa interaction model is introduced for the interface of human and the computer. Alexa skill functions are lambda functions. APIs built using API gateway. The DynamoDB tables and the website are hosted by AWS S3.

How the components interact in a typical user case is described as follows. A user asks Alexa a question, and the Alexa interaction model interprets it and sends it to the supporting lambda function. This lambda function extracts the question text, lemmatizes it, and calls the POST Provider API to obtain the discussion board posts (in database) to select an answer. After lemmatizing the subject and body portions of the candidate posts, they and the lemmatized question are sent to the similarity model to obtain a vector of similarity values. The Alexa skill function then recites the replies obtained from the candidate post with the highest similarity value. If the user is not satisfied with the recited answer, or if no candidate post with a nonzero candidate score is found, the user is given the option of sending the post to the Unanswered Questions Database, where the professor and other students can view the members of the

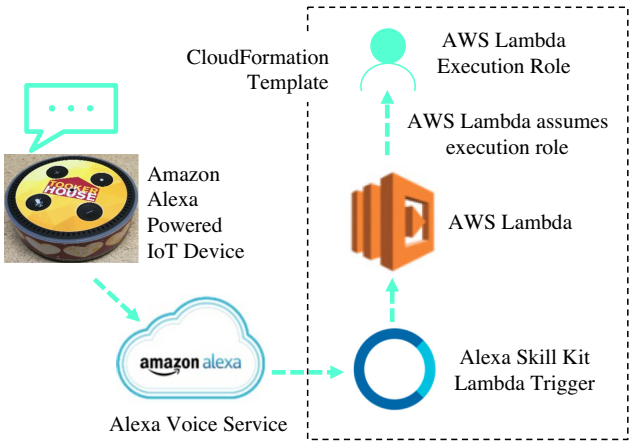

Fig. 2. Amazon voice services and Alexa skills.

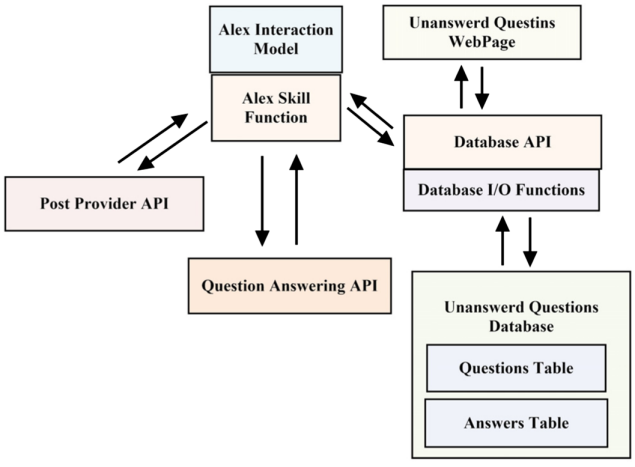

Fig. 4. Flow of the question answering services in IVIAS.

Unanswered Questions Database from the Unanswered Questions Page and answer them. The database is accessed by API calls rather than directly by the skill function or the Questions Page, thus it maintains loose coupling feature.

C. ALEXA INTERACTION MODULE

The interaction model, which defines the terms by which the user interacts with the Alexa skill. First, the invocation is defined, which is the phrase that users say in order to start interacting with the skill. The invocation name of this project is simply "discussion board" in our project.

Intents define the actions that the skill will take at the user's direction. AWS has many default intents, such as YesIntent, NoIntent, and StopIntent. A custom intent is defined in order to allow the user to prompt Alexa to begin answering the question, which is passed through the intent as a slot. The only slot defined is called "QuestionText." Determining how to incorporate a custom intent into the sample utterances is difficult. Sample utterances are the words that a user speaks to activate the intent, and they must contain a clearly defined phrase, called the "carrier phrase" in addition to the slot. Sample utterances cannot be composed of only the slot value. In addition, QuestionText is designed to contain the full text of the question. If any words were omitted and given to the carrier phrase, such as the question words "who," "what," "when," etc., then the question would be stored as a truncated form, in case the user wants to store the question in the Unanswered Questions Database. Therefore, the sample utterances must contain simple carrier phrases such as "my question is" and "I want to know" immediately preceding the question text. This makes interaction with the skill somewhat inconvenient, as users cannot simply ask their questions but must precede them with the carrier phrase. However, this inconvenience is necessary for preserving the original question text, and it is acceptable for students participating in the course.

D. ALEXA SKILL FUNCTION

The Alexa skill function is a complex component because it makes not only the service calls to obtain candidate posts and the question text but also decides the course of the conversation by defining the response to every intent. Fig. 5 is an activity diagram showing what actions to take in response to certain events, assuming the user does not wish to ask another question. This is difficult because the desired action after an intent may differ according to the status of the conversation. If the user answers "yes" to the question of whether they were satisfied by the recited answer, then we expect a different behavior from the skill than that when they said "no" as being asked to send the question to the Unanswered Question Database. However, both events register as YesIntents and are mapped to the same header.

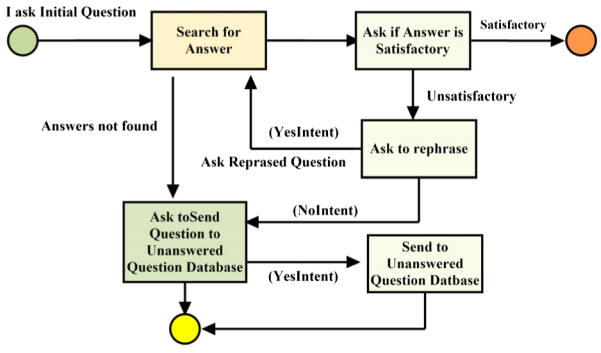

Fig. 5. Activity diagram between actions and events.

In order to achieve these conversational transitions, Alexa's session is used to attribute and define the state of the interaction. As shown in Table 1, the skill can be in one of the five different states: "Initial," "Answering," "Rephrasing," "SendAsking," and "Asking." Users can only ask a question when they are in the "Initial" state, and once that question is asked then they enter the "Answering" state. If the skill is in the "Answering" state and no answer is found from the discussion board data, then they will automatically transition to the "SendAsking" state rather than asking the user whether they were satisfied, which clearly would not be the case. If an answer is found, then the QuestionIntent handler would ask that question, and if the user says "yes," triggering a YesIntent, then Alexa will enter the "Asking" state and ask them if they would like to ask another question.

In the "Asking" state, replying "yes" will return them to the "Initial" state, while answering "no" will end the session. If the user says "no" to the question of whether they were satisfied, then they transition to the Rephrasing state, in which they will be asked if they would like to ask the question again in a different way. If they respond "yes" then they will return to the "Initial" state and try to ask the question again. If they respond "no," then they will enter the SendAsking state and be asked if they want to send the question to the Unanswered Page. Both responses to this question cause a transition to the Asking state, and the only difference is that saying "yes" to this question triggers an asynchronous call to the Database API to save the user's question to the database.

E. POST PROVIDER API

The POST Provider API is created for providing the Alexa Skill function with data scraped from ASU CSE445 course's discussion board. Because the code is separated, the only point of contact is an API call made during the answering process. There is no need to

TABLE 1 Alexa Skill State Transition on Reception of Intents

State	Description	Question Intent	Yes Intent	No Intent
Initial	Waiting for a question	Answering	n/a	n/a
Answering	Answering a question and asking for feedback	n/a	Asking	Rephrasing
Rephrasing	Asking if they would like to rephrase the question	n/a	Initial	Send asking
Send asking	Asking if they would like to send the question to the database	n/a	Asking	Asking
Asking	Asking if they would like to ask another question	n/a	Initial	End session

expose the lambda functions or databases that go into the functionality of the service, as shown in Fig. 5.

To use this service, the client using the Alexa Skill function sends a list of keywords in an array. The service then returns all discussion board posts, represented as JavaScript object notation (JSON) objects, which contain one or more occurrences of any one of the keywords in its post body. Alexa obtains candidate posts by passing the lemmatized form of the user's question as the keywords.

F. QUESTION ANSWERING MODEL

1) DATA PREPROCESSING. Data preprocessing is dependent on the lemmatize function, which takes a string as an input and returns an array of lemmatized tokens. Implementing this function was relatively straightforward because of the use of natural language toolkit (NLTK), which automatically supports tokenizing, and WordNet, which automatically supports lemmatizing. NLTK library and the associated Python libraries make developing this function easy and straightforward. Fig. 6 shows an example of processing a sentence through tokenization, stop word removal, and lemmatization. Lemmatization is the process of converting a word to its base form. The difference between stemming and lemmatization is, lemmatization considers the context and converts the word to its meaningful base form, whereas stemming just removes the last few characters, often leading to incorrect meanings and spelling errors.

2) SD-LSTM MODEL. An SD-LSTM model is proposed in this study and is illustrated in Fig. 7. Siamese networks perform well on similarity tasks and have been used for tasks like sentence semantic similarity, recognizing forged signatures, etc. Fig. 7 shows that Siamese networks are networks that have two or more identical subnetworks in them. It also shows an overview of the processing sequence in the model, which is composed of word embedded layer, LSTM layer, and output layer.

The embedded layer uses word2vec model to map the word number sequence to the word vector sequence as the input of LSTM layer. For example, there are two questions: "How to pass the exam?" and "How to avoid failure in the exam?" (people can use different ways to express the same meanings). We use simple sentences as examples here. In fact, there are more expressions that could have the same meaning. For example, "What should I do to avoid failure in this exam?" Using the model, the aforementioned two questions are tokenized and lowercased as simple words "how\ to\ pass\ the\ exam\n," "how\ to\avoid\failure\in\the\exam\n." Next, the stop words are removed, and the left words are "how\pass\exam \n" and "how\avoid\failure\exam\n," respectively. In the final step of the preprocessing, their main words are transformed into vectors by word2vec tools.

The Siamese LSTM layers are designed based on LSTM cell [24], [25]. LSTM makes improvement to solve the vanishing gradient problem, in which the backpropagated gradients become

(a) "Are notes allowed on the exam"
(b) ["are", "notes", "allowed", "on", "the", "exam"]
(c) ["notes", "allowed", "exam"]
(d) ["note", "allow", "exam"]

Fig. 6. (a) Sentence, (b) tokenization, (c) stop word removal, and (d) lemmatization.

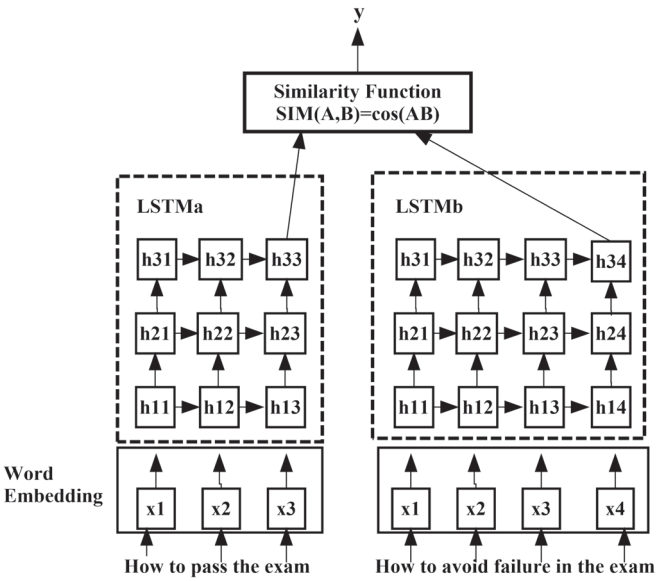

Fig. 7 The structure of the improved Siamese deep LSTM model.

vanishingly small over long sequences, and it makes the standard RNNs suffer. Like the standard RNN, the LSTM sequentially updates a hidden-state representation. The single unit of SD-LSTM is the basic LSTM unit, as shown in Fig. 8.

The components of the LSTM cell are explained as follows:

1) The forget gate

$$\Gamma_f^t = \sigma(W_f[a^{\langle t-1\rangle}, x^t] + b_f), \tag{1}$$

where W_f is the weight of the forget gate, concatenate $[a^{<t-1>}, x^t]$ and then multiply W_f to obtain a vector, whose value is between $0 \sim 1$. b_f is the bias. This forget gate vector will be multiplied by the previous unit's state C. If the result of the above formula is 0 (or near 0), the corresponding information should be deleted; if the value is 1, the information should be retained.

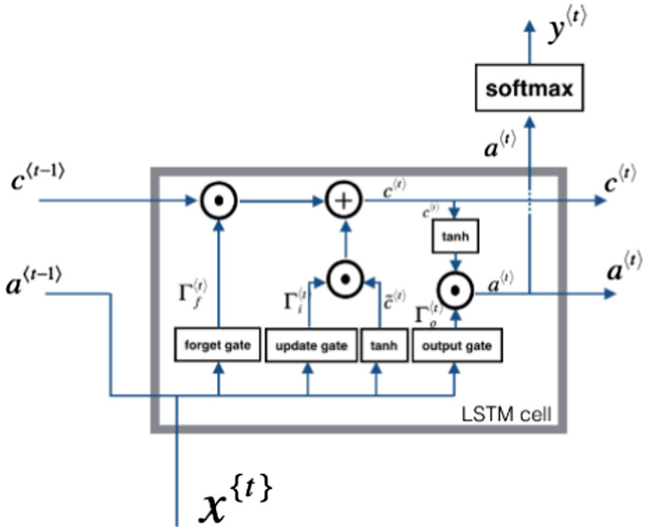

Fig. 8. Structure of an LSTM cell.

2) The update gate

$$\Gamma_i^t = \sigma(W_u[a^{\langle t-1\rangle}, x^t] + b_u). \tag{2}$$

It is similar to the forget gate, where W_u is the weight of the update gate and b_u is the bias.
To update, it needs to create a new vector as the following.

3) The output gate

$$\Gamma_o^{\langle t\rangle} = \sigma(W_o[a^{\langle t-1\rangle}, x^{\langle t\rangle}] + b_o \tag{3}$$

$$a^t = \Gamma_o^{\langle t\rangle} * \tanh(c^{\langle t\rangle}), \tag{4}$$

where W_o is the weight of the update gate and b_o is the bias.

The structure of LSTM neural network determines that it has a great advantage in processing serialized data. It can effectively maintain a long-term memory and can better acquire the semantic features of the whole sentence. LSTM neural network has been successfully applied in many natural language processing tasks, such as emotional analysis and machine translation. Therefore, LSTM is selected to calculate sentence similarity in the project.

The left LSTM and the right LSTM (LSTMa and LSTMb in Fig. 7) learn from the sequence data obtained from the previous part of processing and then use the cosine function to output the similarity value of two sentences:

$$\cos(A_1, B_1) = \frac{A_1 \cdot B_1}{\|A_1\|\|B_1\|}, \tag{5}$$

where, A_1 and B_1 are the projections of sentences 1 and 2 in the embedding space.

The instance loss function $L_W^i$is a contrastive loss function, consisting of terms for the dissimilar (y = 0) case (L_-) and the similar (y=1) case (L_+):

$$L_W^i = y^{(i)}L_+(A_1^{(i)}, B_1^{(i)}) + (1 - y^{(i)})L_-(A_1^{(i)}, B_1^{(i)}). \tag{6}$$

The total loss function over a dataset $X = \{\langle A_1^{(i)}, B_1^{(i)}, y^i\rangle\}$ is given by

$$L_w(X) = \sum_{i=1}^{N} L_W^{(i)}(A_1^{(i)}, B_1^{(i)}, y^{(i)}). \tag{7}$$

The loss functions for the dissimilar and similar cases are detailed as the following:

$$L_- = \begin{cases} E_W^2 & \text{if } Ew < m \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

$$L_+(A_1, B_1) = \frac{1}{4}(1 - E_w)^2, \tag{9}$$

where, $E_W(A_1, B_1) = \cos(A_1, B_1)$, and m is the loss value of the intersection which L_+ and L_- go through.

3) THE PSEUDO CODE OF A TRAINING STEP OF SIAMESE LSTM MODEL. The pseudo code of a training step of Siamese LSTM model is detailed in Fig. 9.

G. CORPUS MANAGEMENT

1) QUESTIONS DATABASE API. As mentioned before, the database is not accessed directly by the skill function or from the questions page. Instead, it is accessed through an API. This API offers a useful layer of abstraction. If, for some reason, it was decided to move from DynamoDB to some other database, then none of the clients' code needs to change, as long as the API remains the same. The available API methods include the following:

- Adding questions to the Questions table, as well as adding unanswered questions from the Alexa skill function.
- Selecting all questions from the Questions table and displaying the questions which cannot be answered on the unanswered questions page.
- Adding answers to the Answers table, as well as answering functionality of the unanswered questions page.
- Selecting all answers for a given question from the Answers table and displaying questions that cannot be answered in the unanswered questions page.

The unanswered questions page is a publicly accessible webpage implemented with React and hosted on AWS S3. On rendering the page, each member of the Questions table is given as an HTML collapsible component, as shown in Fig. 10. When these components are expanded, the answers to the question, if there are any, are loaded and displayed as shown in the lower part of Fig. 10. Users can submit answers to these questions using the text area and button. The submitted answers are then stored in the Answers table.

2) UNANSWERED QUESTIONS DATABASE. The unanswered questions database stores questions that the Alexa Skill is unable to give a satisfactory answer for into a table called "Questions," while answers for these questions submitted by users through the Unanswered Questions Page are stored in a table called "Answers." The "Questions" table stores the text of the question into "QuestionText" and the time the question was asked into "QuestionTime." The "Answers" table stores the "QuestionText" and "QuestionTime" of the question it is answering as well as columns

```
Input training set(x1_batch, x2_batch, y_batch),
initialized parameters dropout_keep_prob
define function train_step(x1_batch, x2_batch, y_batch)
if random() > 0.5:
    feed_dict = {
    siameseModel.input_x1: x1_batch,
    siameseModel.input_x2: x2_batch,
    siameseModel.input_y: y_batch,
    siameseModel.dropout_keep_prob: FLAGS.dropout_keep_prob, }
else
    feed_dict = {
    siameseModel.input_x1: x2_batch,
    siameseModel.input_x2: x1_batch,
    siameseModel.input_y: y_batch,
    siameseModel.dropout_keep_prob: FLAGS.dropout_keep_prob, }
step, loss, accuracy, dist = sess.run
([tr_op_set, global_step, loss, accuracy, distance], feed_dict)
time_str = datetime.datetime.now().isoformat()
d = np.copy(dist)
d[d >= 0.5] = 999.0
d[d < 0.5] = 1
d[d > 1.0] = 0
accuracy = np.mean(y_batch == d)
output loss, accuracy, dist, d
```

Fig. 9. Pseudo code of a train step of Siamese LSTM model.

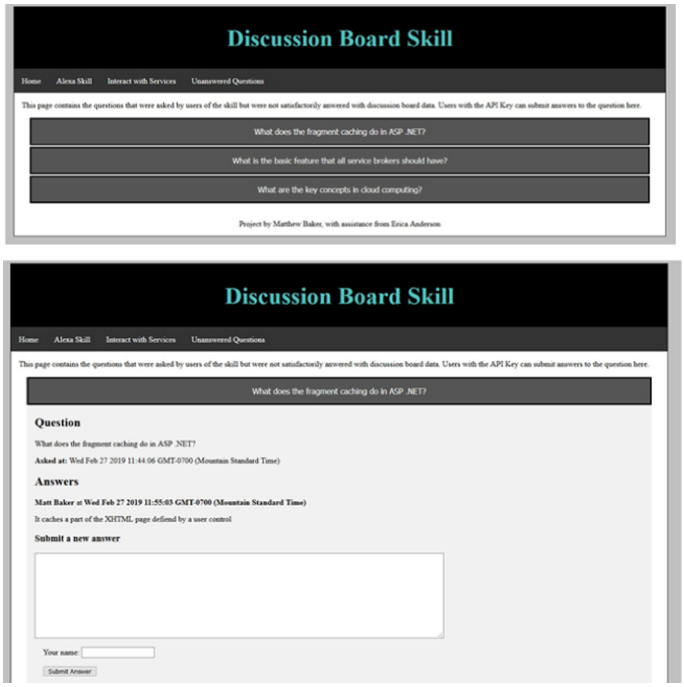

Fig. 10. Discussion board skill.

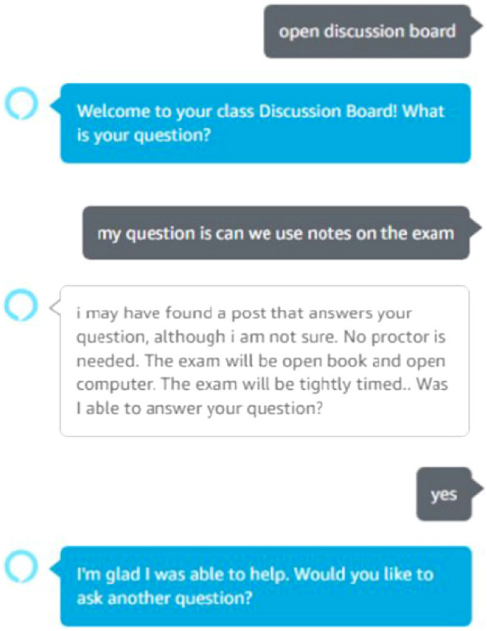

Fig. 11. Test sequence when an answer is found.

for the text of the answer, time of the answer, and answerer. The schema for the two tables is as follows:

Questions (QuestionText, QuestionTime)
Answers (AnswerText, AnswerTime, Answerer, QuestionText, QuestionTime)

All these column values are of type "string." DynamoDB is the most convenient and well-documented database solution for AWS applications. It is not a relational database, and in this case, it calls for a one-to-many relationship between questions and answers.

IV. EXPERIMENTS

The prototype we developed is up and running in AWS cloud and is equipped with the actual questions from ASU CSE445 class. In this section, we present the tests, data collection, and evaluation.

A. SAMPLE TESTS

First, we present the two scenarios of testing the developed prototype. Fig. 11 presents a question/answer sequence when a question is found in the system and the student is happy about the answer.

Fig. 12 presents a question/answer sequence when a question is not found in the system and the system asks the student if the student wants to send the question to the unanswered page, where the instructor can answer the question and add the question and answer into the corpus management system.

These test scenarios will be further analyzed and evaluated in the following sections.

B. DATA AUGMENTATION AND TRAINING

We conduct a set of experiments to test the proposed model's capabilities.

The data we collected from the school at this time are sufficient for the similarity test. We employed AskUbuntu corpus [26]. This repository contains a preprocessed collection of 167,765 questions taken from AskUbuntu.com. It contains manual annotations, marking pairs of questions as "similar" or "nonsimilar."

The training set for a Siamese network consists of triplets (A_1, B_1,y), where A_1 and B_1 are word sequences and $y \in \{0,1\}$. $y = 1$ indicates A_1 and B_1 are similar, while $y = 0$ means A_1 and B_1 are dissimilar. The aim of training is to minimize the distance in an embedding space between similar pair sentences and maximize the distance between dissimilar pair sentences.

Since the annotations have high precision and do not require additional manual annotations, it allows us to use a much larger training set. We randomly choose 30 questions from the corpus paired with each query question as negative pairs in each training epoch.

The first 2000 questions in the dev set and the test set were provided by Cicero dos Santos *et al.* [26]. For each of the above questions, we retrieved the top 10 similar candidates using Best Matching Algorithm (BM25) and manually annotated the resulting pairs as similar or dissimilar. We add questions from ASU Canvas course discussion board too.

The experiment results of the proposed model is shown in Fig. 13.

C. BASELINES AND EVALUATION METRICS

We compared our model with the following baseline models:

1) TF–IDF: Term frequency–inverse document frequency (TF–IDF) is a general method usually used for information retrieval. Words that appear in relatively fewer documents are given higher importance by multiplying every tf (term frequency) term by its respective idf (inverse document frequency) value.

$$\text{idf} = \log \frac{n}{x(t)+1}. \tag{10}$$

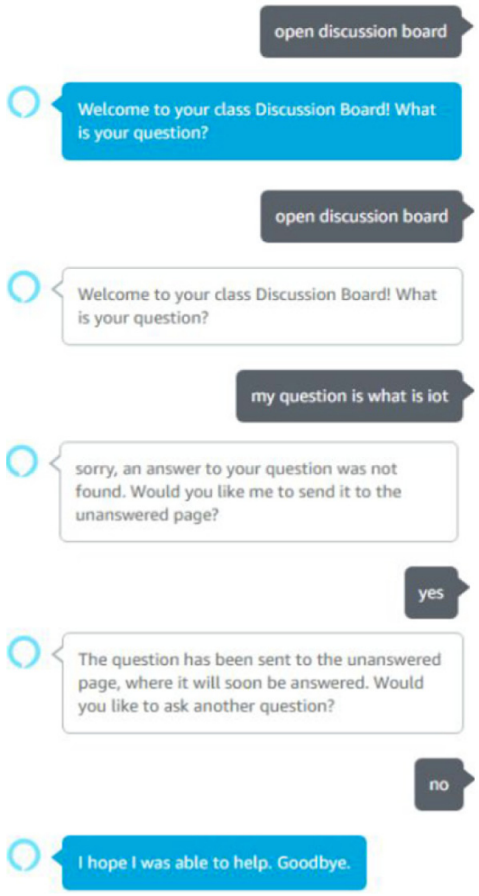

Fig. 12. Test sequence when an answer is not found.

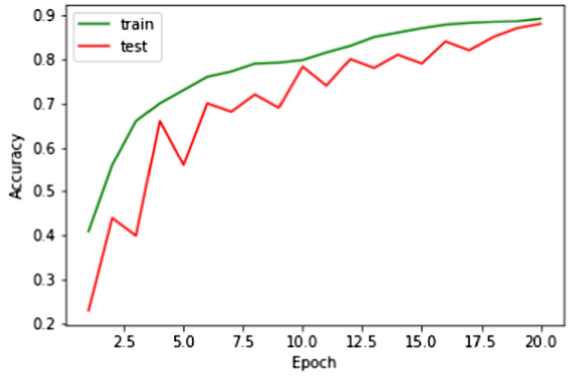

Fig. 13. Accuracy on the training and the testing data.

Once every tf value is multiplied by its idf value, we have the tf-idf matrix, and the vectors of the tf-idf matrix can be used in the cosine similarity formula given before to compute the similarity between two documents.

2) BM25: BM25 is an algorithm used for evaluating the correlation between search terms and documents. It is an algorithm based on probability retrieval model. It consists of word relevance in documents, word relevance in query keywords, and word weights.

TABLE 2 Experimental Results

	Dev		Test	
Method	**MAP**	**MRR**	**MAP**	**MRR**
TF-IDF	53.1	69.3	54.7	69.4
BM25	54.2	67.4	58.3	71.5
CNNs	63.5	74.6	65.2	75.8
SD-LSTM	87.5	89.1	88.4	90.7

3) CNNs: CNN with max-pooling is introduced to produce sentence embeddings, adaptable word embedding matrix preinitialized with 300D GloVe and a projection matrix applied to both sentences to project them to a common external similarity space.

The experimental results are shown in Table 2, where MAP is for mean average precision and MRR is for mean reciprocal rank.

The hyperparameters of SD-LSTM are set as the following: The embedding dimension is 300, the dropout rate is 0.6, the hidden units is 50, the batch size is 64, the number of epochs is 200, and the number of layers is 3. It can be seen from Table 2 that SD-LSTM outperformed TF-IDF, BM25, and CNNs model at both the dev dataset and the test dataset, whose MAP and MRR are 87.5/89.1 and 88.4/90.7, respectively.

D. PERFORMANCE EVALUATION

Different experiments have been conducted and different types of data have been collected. In this section, we present the experiments on the wait time between asking a question and receiving an answer. This can be due to a number of factors, such as the size of the lemmatization and similarity deployment packages or the amount of discussion board data to evaluate. However, an interesting trend with the duration of the service calls is that the first calls after long gaps tend to be the longest, which can be related cold-start effect. Fig. 14 shows the duration in milliseconds of the Alexa service lambda function. The experiments show a consistent pattern of abnormally high values after a period without activity. The wait time becomes less of an issue after more skills are used. Ways to mitigate the cold start issue include using a background process to continuously hit the lambda execution environment to keep it warm, or to utilize provisioned concurrency https://aws.amazon.com/blogs/compute/new-for-aws-lambda-predictable-start-up-times-with-provisioned-concurrency/.

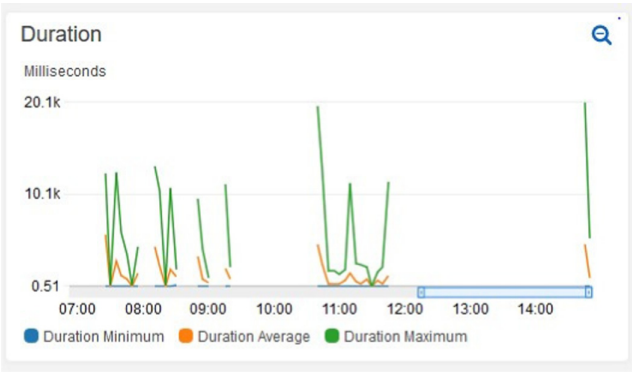

Fig. 14. Duration spikes after periods of inactivity.

Another option would be to not use a serverless architecture, but instead provision hosts using AWS EC2 or run the services in a container using AWS Fargate. Both would offer more robust response times, but at the cost of Lambda's "on-demand" pricing model.

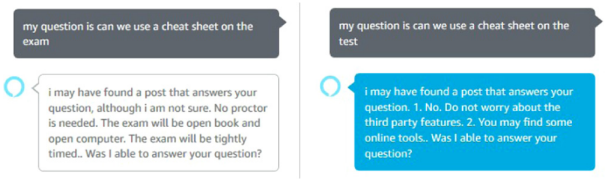

Fig. 15. Substituting one word with a synonym has a catastrophic result.

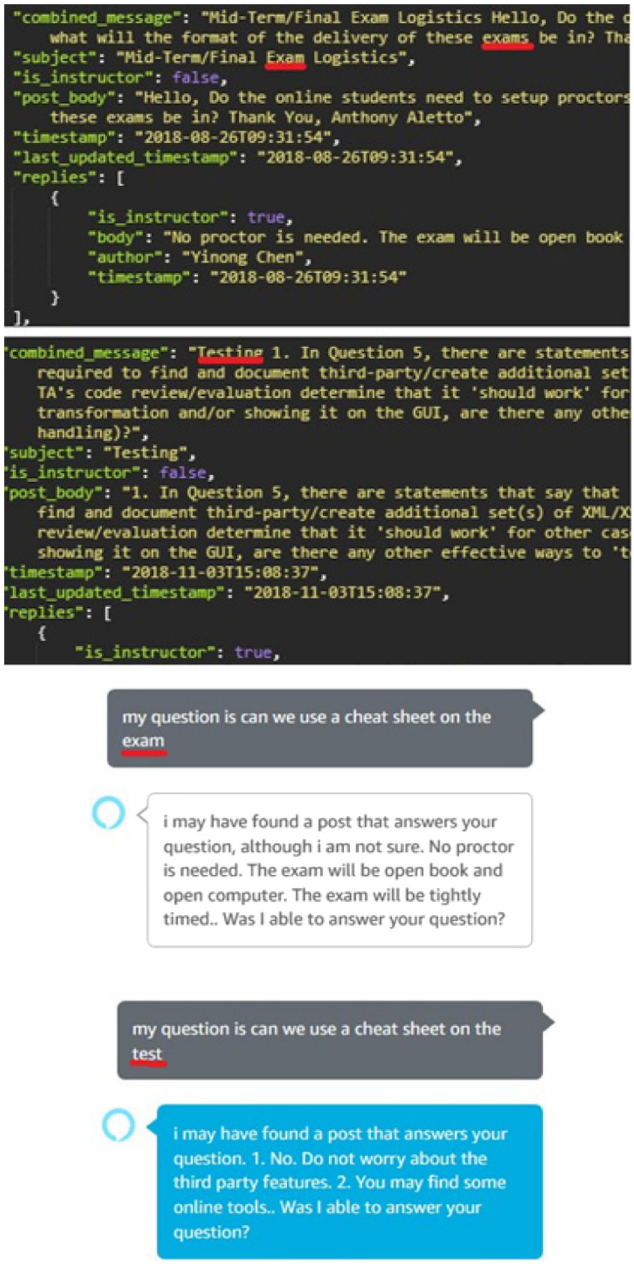

Fig. 16. TF-IDF similarity is based matching words, not meanings.

This behavior is well acceptable when a user asks a question that already exists on the discussion board. For example, a discussion board post exists in the dataset with the given subject.

When using TF-IDF method in the experiment, if a user asks a question, the skill matches with that post with a cosine similarity score of 0.7933. However, this tool does not perform well when it comes to understanding the semantic meaning of the question being asked. For example, if we asked the skill, "Can we use a cheat sheet on the exam," the skill will respond with: "No proctor is needed. The exam will be open book and open computer. The exam will be tightly timed." However, if we ask Alexa the same question but use "test" instead of "exam," we will obtain a different result, as pictured in Fig. 15.

This happens because TF-IDF is based on the overlap of *words*, not the **overlap** of *meaning*. The reason why the second wording of the question has a different answer is because the first answer originated from a post that featured the word "exam" but not the *exact* word "test." Meanwhile, the new answer originated from a post that featured the **word** "test" (or "testing," which is lemmatized as "test"), but not as a synonym for "exam." TF-IDF as used in this project does not take the words' context into account, nor is it capable of understanding the concept of synonyms. Fig. 16 shows this flaw in question processing.

The inability of TF-IDF and BM25 to find an answer for every single question posed is not a fundamental flaw. The data to answer that question may not be present in the discussion board data or in the unanswered database. However, when we used SD-LSTM the problem is overcome. The similarity reaches 0.9253 between the two sentences "My question is can we use a cheat sheet on the exam" and "My question is can we use a cheat sheet on the test."

V. CONCLUSIONS

The goal of this project was to create an IVIAS for answering students' questions, as a supplementary mechanism to the course discussion board data. IVIAS was designed with the similarity sentence matching algorithm instead of a more sophisticated question answering system. The skills were capable of matching questions to the posts that had nearly similar meaning or paraphrase. Compared with some baseline methods, it achieved better results. Future work will be done to improve the accuracy of the system based on the data for more questions and courses. The goal is to put the system into practical use for our classes.

ACKNOWLEDGMENT

The authors wish to thank their colleagues and students who were involved in this study and provided valuable implementation and technical support. The research is partly supported by general funding at IoT and Robotics Education Lab and FURI program at Arizona State University and is partly supported by China Scholarship Council, Guangdong Science and Technology Department, under Grant Number 2016A010101020, 2016A010101021, and 2016A010101022, and Guangzhou Science and Information Bureau under Grant Number 201802010033.

REFERENCES

[1] P. Rajpurkar, R. Jia, and P. Liang, "Know what you don't know: Unanswerable questions for SQuAD," in *Proc. 56th Annu. Meeting ACL*, Melbourne, Australia, vol. 2, 2018, pp. 784–789.

[2] J. Bao, N. Duan, M. Zhou, and T. Zhao, "Knowledge-based question answering as machine translation," in *Proc. 52nd Annu. Meeting ACL*, Baltimore, MD, USA, Jun. 23–25, 2014, pp. 967–976.
[3] Y. Chen, *Service-oriented Computing and System Integration: Software, IoT, Big Data, and AI as Services,* 7th ed., Kendall Hunt Publishing, Dubuque, IA , USA, 2020.
[4] W. T. Tsai, Q. Huang, J. Xu, Y. Chen, and R. Paul, "Ontology-based dynamic process collaboration in service-oriented architecture," in *IEEE Int. Conf. SOCA*, Newport Beach, CA, USA, 2007, pp. 39–46.
[5] Amazon AWS Alexa, https://developer.amazon.com/alexa, accessed on March 1, 2021.
[6] Apple Siri, https://www.apple.com/siri/, accessed on March 1, 2021.
[7] Google Assistant, https://assistant.google.com/, accessed on March 1, 2021.
[8] Microsoft Cortana, https://www.microsoft.com/en-us/cortana/, accessed on March 1, 2021.
[9] J. Zhu, J. Hu, M. Zhang, Y. Chen, and S. Bi, "A fog computing model for implementing motion guide to visually impaired," *Simul. Model. Pract. Th.,* vol. 101, May 2020.
[10] G. De Luca, Z. Li, S. Mian, and Y. Chen, "Visual programming language environment for different IoT and robotics platforms in computer science education," *CAAI Trans. Intell. Technol.,* vol. 3, no. 2, pp. 119–130, Jun. 2018.
[11] Y. Chen and H. Hu, "Internet of Intelligent Things and robot as a service," *Simul. Model. Pract. Th.*, vol. 34, pp. 159–171, May 2013.
[12] Y. Chen, "IoT, cloud, big data and AI in interdisciplinary domains," *Simul. Model. Pract. Th.*, vol. 102, p. 102070, 2020.
[13] G. De Luca and Y. Chen, "Explainable artificial intelligence for workflow verification in visual IoT/robotics programming language environment," *J. Artif. Intell. Technol.*, vol. 1, pp. 21–27, Jan. 2021.
[14] A. Miriyev and M. Kovac, "Skills for physical artificial intelligence," *Proc. IEEE*, vol. 2, pp. 658–660, Nov. 2020.
[15] S. Young, M. Gasic, B. Thomson, and J. D. Williams, "POMDP-based statistical spoken dialog systems: A review," in *Proc. IEEE*, vol. 101, no. 5, May 2013, pp. 1160–1179.
[16] A. Severyn and A. Moschitti, "Automatic feature engineering for answer selection and extraction," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, Seattle, Washington, USA, Oct. 18–21, 2013, pp. 458–467.
[17] I. Yu, J. Buys, and P. Blunsom, "Online segment to segment neural transduction," arXiv preprint arXiv:1609.08194, 2016.
[18] L. Dong, F. Wei, M. Zhou, and K. Xu, "Question answering over freebase with multi-column convolutional neural networks," in *ACL*, Beijing, China, vol. 1, 2015, pp. 260–269.
[19] M. Iyyer, J. L. Boyd-Graber, L. M. B. Claudino, R. Socher, and H. Daumé III, "A neural network for factoid question answering over paragraphs," in *EMNLP*, 2014, pp. 633–644.
[20] D. Wang and E. Nyberg, "A long short-term memory model for answer sentence selection in question answering," in *ACL-IJCNLP*, Beijing, China, 2015, pp. 707–712.
[21] M. Tan, C. dos Santos, B. Xiang, and B. Zhou, "LSTM-based deep learning models for non-factoid answer selection," arXiv.org, arXiv:1511.04108, 2015.
[22] Y. Yang, L. He, and C. C. Chen, "Question categorization of community question answering by combining Bi-LSTM and CNN with attention mechanism," *Comput. Syst. Appl.,* vol. 27, no. 9, pp. 157–162, 2018.
[23] J. Mueller and A. Thyagarajan, "Siamese recurrent architectures for learning sentence similarity," in *Proc. AAAI'16*, Atlanta, GA, USA, pp. 2786–2792.
[24] P. Neculoiu, M. Versteegh, and M. Rotaru, "Learning text similarity with Siamese recurrent networks," in *Proc. RepL4NLP*, Berlin, Germany, Aug. 2016, pp. W16–1617.
[25] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *ICML'13*, vol. 28, Atlanta, GA, USA, Jun. 16–21, 2013, pp. 1310–1318.
[26] C. dos Santos, L. Barbosa, D. Bogdanova, and B. Zadrozny, "Learning hybrid representations to retrieve semantically equivalent questions," in *ACL-IJCNLP*, vol. 2 (Short Papers), Beijing, China, Jul. 2015, pp. 694–699.