Learner question’s correctness assessment and a guided correction method: enhancing the user experience in an interactive online learning system

Saurabh Pal¹, Pijush Kanti Dutta Pramanik¹, Aranyak Maity² and Prasenjit Choudhury¹

¹ Department of Computer Science & Engineering, National Institute of Technology, Durgapur, West Bengal, India
² School of Computing, Informatics, and Decision Systems Engineering, Arizona State University, Tempe, AZ, USA

ABSTRACT

In an interactive online learning system (OLS), it is crucial for the learners to form the questions correctly in order to be provided or recommended appropriate learning materials. The incorrect question formation may lead the OLS to be confused, resulting in providing or recommending inappropriate study materials, which, in turn, affects the learning quality and experience and learner satisfaction. In this paper, we propose a novel method to assess the correctness of the learner’s question in terms of syntax and semantics. Assessing the learner’s query precisely will improve the performance of the recommendation. A tri-gram language model is built, and trained and tested on corpora of 2,533 and 634 questions on Java, respectively, collected from books, blogs, websites, and university exam papers. The proposed method has exhibited 92% accuracy in identifying a question as correct or incorrect. Furthermore, in case the learner’s input question is not correct, we propose an additional framework to guide the learner leading to a correct question that closely matches her intended question. For recommending correct questions, soft cosine based similarity is used. The proposed framework is tested on a group of learners’ real-time questions and observed to accomplish 85% accuracy.

INTRODUCTION

Online learning systems (OLSs) have brought great advantages to all kinds of formal and informal learning modes (Radović-Marković, 2010; Czerkawski, 2016; Pal et al., 2019). Over the years, OLSs have evolved from simple static information delivery systems to interactive, intelligent (Herder, Sosnovsky & Dimitrova, 2017; Huang et al., 2004), and context-aware learning systems (Wang & Wu, 2011), virtually incorporating real-life teaching and learning experience (Mukhopadhyay et al., 2020). In today’s OLSs, much of
the emphasis is given on designing and delivering learner-centric learning (Beckford & Mugisa, 2010) in terms of the learning style, learning approaches, and progress of a particular learner (Dey et al., 2020).

Like every learning process, one key aspect of an OLS is interaction, which makes learning more practical and dynamic (Donnelly, 2009; Pal, Pramanik & Choudhury, 2019). But, despite the advantages, due to high cost and complexity, contents developed for OLSs have limited or no interaction. The basic (or one way) interaction is incorporated in most of the OLSs through demonstration or illustration, which can be useful for very elementary learning options like remembering and comprehending. To achieve advanced learning skills like analyzing, evaluating, creating, and applying, a higher level of interactions like discussion, hands-on experiments, exchanging views with experts, etc., are required (Sun et al., 2008). The best possible way of interaction in an OLS is to devise real-time interaction between the learner and the expert/trainer (Woods & Baker, 2004; Wallace, 2003). In the absence of audio–video based interaction, the best option is to go for a question-answer based OLS (Nguyen, 2018; Srba et al., 2019) since questions are the most natural and implacable way a human enquires about information.

Interacting with a computer through natural language and to make it interpret the meaning of the communicated text has many implicit challenges associated with human-computer interaction. Existing applications like search engines, question-answering based systems (Allam & Haggag, 2012; Sneiders, 2009), chatbots (Adamopoulou & Moussiades, 2020), etc., work over user queries to deliver the required information. Fundamentally, these systems process the input query to determine its structure and semantics to understand the intention of the query. Therefore, the correctness of the semantics of the query determines the response given by these automated systems.

**Significance of the correctness of the input question in an interactive learning systems**

For efficient information retrieval, most of the recommendation systems focus on improving the efficiency of the recommendation engine. But, however efficient the recommendation engine is, if the query itself is incorrect, the search engine will not be able to retrieve the suitable information that was actually intended by the user.

Similarly, in an OLS, while interacting, if the learner inputs an incorrect question, due to the absence of the cognitive ability of the embedded search and recommendation engine, it will try to find the learning materials against the wrong input. This will lead to inappropriate learning material recommendations, which will, in effect, dissatisfy the learner, and the purpose of the OLS will not be fulfilled. Therefore, it is important that the OLS understands the learner’s actual intention when she inputs a question while interacting.

Hence, in an OLS, framing the right question in terms of grammar, word use, and semantics is an absolute requirement. But often, people frame questions incorrectly,
leading to ambiguous information retrieval, which misleads learners. Generally, the following are the two reasons for an incorrect question framing:

a) **Language incompetency**: The lack of expertise in communicative language may cause a learner to frame a question with incorrect grammatical structure, spelling mistakes, and the inability to use appropriate words. For instance, the non-native English-speaking people having poor knowledge of English most often find it difficult to compose questions in English. For example, a question given by such a user, “HTML in how Java”, demonstrates the incorrect framing of the question. What is being asked is not understandable. It could be the programming of HTML script through Java language, or it could be the application of Java program on an HTML page. The question lacks adequate articulation, due to which the desired meaning cannot be recognized. This makes correct parsing of the question impossible.

b) **Lack of domain knowledge**: Insufficient domain knowledge also leads to frame an incorrect question. For example, the question “how a parent class inherits a child class” is syntactically correct but semantically (or technically) incorrect. Exchanging the phrases “parent class” and “child class” would make the question correct. Ignorance or the lack of domain knowledge can reason these types of semantically incorrect framing of questions. In this case, the question might be parsed successfully, but the learner will get unintended results.

In both cases, users will not get the desired answer to their questions. Therefore, it is important to validate the correctness of the question in an interactive and question-answer based automated learning system.

**Research objective**

From the above discussion, we can put forward the following research objectives:

- How to assess if the learner’s question given as input to a query-based learning system is syntactically and semantically correct or not?
- If the question is not correct, then how to handle this to improve the recommendation?

**Existing solution techniques, their limitations, and research motivation**

In this section, we shall investigate whether the existing methods are capable of addressing the above-mentioned research objectives.

**Assessing the correctness of a question**

The problem of assessing the correctness of a question can be described in terms of sentence validation and meaning extraction. To address the problem of validation and semantics, the following existing techniques can be used.

**NLP**: The progress in natural language processing (NLP) has led to advanced techniques that allow knowing sentence structure, but comprehending its semantics still remains a challenge. NLP processing techniques for determining the intention or
semantics of a sentence include sentence parsing, phrase, and relevant words (or key terms) identification. The words thus identified are being related by the relationships like dependency, modifier, subject and object, action for inducing the meaning of the sentence. In determining the relationships among the words in a sentence, often rule-based approach is adopted. Defining the rules which understand words and the relationship and interdependency among them is a non-trivial task with limited application scope. Because it is not possible to rule in the usage and relationship of all words from the English language (Sun et al., 2007b; Soni & Thakur, 2018). And hence, understanding which word combination in framing a question is correct or incorrect is very difficult. The NLP techniques assume the placement and occurrence of words in question are implicitly correct. There is no knowledge for the words that are placed incorrectly or are missing. Because of this, NLP fails to identify if the question framing is correct or not (Cambria & White, 2014; Leacock et al., 2010).

**Pattern matching:** In another approach, pattern matching is used to assess the correctness of the sentence. Pattern matching, in contrast to NLP techniques, is a feasible solution that allows matching a sentence pattern from available sentence patterns to find whether the sentence is matching to existing patterns or not. This approach could suitably be applied to find whether the given question is correct or incorrect. Thus, escaping the intrinsic complexity of knowledge mapping, word by word relationship and missing word problem as found in NLP techniques. In regard to pattern matching, the application of machine learning is very successful in following up with patterns. But, its inherent limitation in learning by not considering word sequence in a sentence had put constraints in verifying a question’s rightness (Soni & Thakur, 2018). For machine learning algorithm, a question "can object be a data member of a class" and "can class be data member of an object" are same. The placement of words or the sequence of words in the sentence does not matter, but only their appearance matters. So, seemingly to a machine learning algorithm, both the sentences are the same (Kowsari et al., 2019). This raises issues where machine learning fails to interpret sentences that are meaningfully incorrect due to misplacements of words.

**Addressing the incorrect question**

Generally, in interactive systems such as recommendation systems and intelligent search engines, if the user enters an incorrect query, the system can autocorrect the wrong input query and searches information against the autocorrected query. Here, the user’s involvement is not required. But this approach suffers from the following issues (Wang et al., 2020):

- It is limited to structure and syntactic corrections of the sentence.
- Not able to correct the semantic errors.
- The intention of the query is not judged; hence, the correctness of the query may not be exact or appropriate.
**Motivation**

From the above discussions, it is obvious that the existing approaches have significant limitations in addressing our research goals. Moreover, none of the work has addressed the case of learner query in an OLS, especially the issues mentioned in the previous subsections “Assessing the correctness of a question” and “Addressing the incorrect question”. Furthermore, no work is found for checking the correctness of a learner’s question submitted to an OLS and to resolve the issue if the input question is incorrect.

**Proposed solution approach**

Considering the research gap, we propose the following two ways to address the two above mentioned research objectives:

- Using a tri-gram based pattern matching to check the sentential (by construction) and semantical (meaning) structure of the question.
- Instead of autocorrecting, guiding the learner to the intended correct question through one or more turns of question suggestions.

The abstract layout of the proposed approach is shown in Fig. 1.

**Authors’ contribution**

To attain the above-mentioned proposals, in this paper, we made the following contributions:

a) To assess the correctness of the learners’ questions:

- We built two sets of corpora comprising 2,533 (for training) and 634 (for testing) questions on core Java.
- We generated a tri-gram language model.
- We created a classifier to identify the correct and incorrect questions based on the tri-gram language model.
- The classification is evaluated on the test corpus data.
- The efficacy of the classifier was compared with other n-gram models as well as with other research works.

b) To address the issue of incorrect question:

- We proposed a framework for suggesting correct questions to the learner.
- We designed a web-based client/server model to implement the framework.
- The efficacy of the framework is assessed by a group of learners.
- The proposed similarity model used in the framework is compared with other existing similarity measures.
- The performance of the framework is assessed by Shannon’s diversity and equitability indices.
Paper organization

“Related Work” mentions related work discussing the different error-checking methods and their limitation. “Assessing the Correctness of the Learners Input Questions” presents the correctness assessment methodology of the learners’ questions. Guiding the learner to find the correct question is presented in “Guiding the Learner to the Probable Correct Question”. The experiments and the result analysis of both the proposed methods are discussed separately in their respective sections. “Conclusions and Further Scope” concludes the paper.

RELATED WORK

Identifying the correctness of a question is related to determining the errors in the sentential text. Sentential errors are not limited to the semantics of the text but to other different types of errors like the wrong usage of words, spelling mistakes, punctuation marks, grammatical errors, etc. Soni & Thakur (2018) categorized the errors in a sentence as:

- Sentence structure error: The error in a sentence generates due to different organizations of POS components in a sentence.
- Spelling error: The error which is generated due to the wrong spelling of words or meaningless strings in a sentence.
- Syntax error: The error in sentence due to wrong/violation of grammar. The syntax error is of the following types:
  - Subject-verb error
  - Article or determiner error
  - Noun number error
  - Verb tense or verb form error
  - Preposition error
- Punctuation error: The error in a sentence, which is generated due to misplacing or missing punctuation marks.
- Semantic error: The error that makes the sentence senseless or meaningless due to the wrong choice of words and their placing.

Among these five error types, detecting sentence structure error, syntax error, and semantic errors are the significant ones for finding the correctness of a question sentence used in a query-based interactive online recommendation system. Different approaches and strategies are found in the literature for detecting the different types of errors in a textual sentence. These different error detection approaches can be categorized as a rule-based approach, statistical approach, and hybrid approach (Soni & Thakur, 2018). These different error detection categories that are adopted in some notable research work that has been carried out for detecting the significant errors in a textual sentence are shown in Table 1.

It is seen that the rule-based approach has been quite effective in detecting sentence structure error, syntax error, and punctuation error. While, the statistical approach works well to find the structure errors, spelling errors, and semantic errors (word usage and placement error). Most of the research works for detecting an error in a textual sentence are limited to word ordering error, wrong usage of words, word collocation errors, and grammatical errors in a sentence.

The sentence structure errors due to the disarrangement of words (misplaced words) and incorrect organization of the sentence’s POS components have been mitigated differently. A rule-based approach was used by Malik et al. (Malik, Mandal & Bandyopadhyay, 2017) by applying POS identification and NLP production rule to check the grammatical error in the sentence. Chang et al. (2014) proposed a rule-based database approach to detect word error, word disorder error, and missing word error. Similarly, Lee et al. (2013a) manually created a list of 60 rules to detect sentence structure errors. In another approach, Tezcan, Hoste & Macken (2016) proposed a rule-based dependency parser that queries a treebank for detecting sentence structure error. In the statistical approach, n-gram based (Ganesh, Gupta & Sasikala, 2018) and machine learning based (Schmaltz et al., 2016) techniques are followed to determine the errors. Islam et al. (2018) proposed sequence to sequence learning model which uses encoder-decoder architecture for resolving missing word error and incorrect arrangement of words in the sentence. The decoder is a recurrent neural network (RNN) along with long and short-term memory (LSTM) for decoding the correct substitute for grammatical errors. Sun et al. (2007a) followed a hybrid approach to resolve the sentence structure error. They used an NLP-based POS tagging and parse tree to determine the features of an incorrect sentence and then classified for grammatical error using the classifiers like support vector machine (SVM) and Naïve Bayes (NB).

The syntax errors are due to wrong or inappropriate use of language grammar. Over the years, different approaches (e.g., rule-based, statistical, and hybrid) have been explored in research works. For syntax error detection, rule-based techniques like the NLP production rule (Malik, Mandal & Bandyopadhyay, 2017), rule-based database approach
Chang et al., 2014, and rule-based dependency parser (Tezcan, Hoste & Macken, 2016) have been applied. Othman, Al-Hagery & Hashemi (2020) proposed a model based on a set of Arabic grammatical rules and regular expressions. Among the different statistical techniques, the use of neural networks was found very effective in determining syntax error (Zhao et al., 2019). Different advanced variations of a neural network like bi-directional RNN with bidirectional LSTM (Rei & Yannakoudakis, 2017; Yannakoudakis et al., 2017), neural sequence to sequence model with encoder and decoder (Ge, Wei & Zhou, 2018), etc., are proposed for error detection in a sentence. Sonawane et al. (2020) introduced a multilayer convolution encoder-decoder model for detecting and correcting syntactical errors. Besides neural networks, another machine learning technique like SVM (Maghraby et al., 2020) is also found to be used for detecting syntax errors. The features that are considered for learning by various machine learning approaches are prefix, suffix, stem, and POS of each individual token (Wang et al., 2014). The error detection and correction are often carried out at the individual token level of each sentence (Felice & Briscoe, 2015).

| Table 1 Related work categorization based on error type and resolving approach. |
|---------------------------------|---------------------------------|---------------------------------|---------------------------------|
| Error detection approach         | Working method                  | Sentence structure error         | Syntax error                    | Semantic error                  |
| Rule-based approach              | The rule-based approach calls   | (Malik, Mandal & Bandyopadhyay, 2017; Chang et al., 2014; Tezcan, Hoste & Macken, 2016; Lee et al., 2013a) | (Malik, Mandal & Bandyopadhyay, 2017; Chang et al., 2014; Tezcan, Hoste & Macken, 2016; Othman, Al-Hagery & Hashemi, 2020) | (Chang et al., 2014) |
| Statistical approach             | The statistical approach uses   | (Ganesh, Gupta & Sasikala, 2018; Schmaltz et al., 2016; Islam et al., 2018; Xiang et al., 2015; Zheng et al., 2016; Yeh, Hsu & Yeh, 2016; Ferraro et al., 2014) | (Rei & Yannakoudakis, 2017; Ge, Wei & Zhou, 2018; Zhao et al., 2019; Yannakoudakis et al., 2017; Felice & Briscoe, 2015; Wang et al., 2014; Xiang et al., 2015; Zheng et al., 2016; Yeh, Hsu & Yeh, 2016; Ferraro et al., 2014; Sonawane et al., 2020; Zan et al., 2020; Agarwal, Wani & Bours, 2000; Maghraby et al., 2020) | (Yeh, Hsu & Yeh, 2016; Shiue, Huang & Chen, 2017; Yu & Chen, 2012; Cheng, Yu & Chen, 2014; Rei & Yannakoudakis, 2016; Rei & Yannakoudakis, 2017; Cheng, Fang & Ostendorf, 2017; Ferraro et al., 2014; Islam et al., 2018; Zheng et al., 2016; Xiang et al., 2015; Zan et al., 2015; Zan et al., 2020; Agarwal, Wani & Bours, 2000) |
| Hybrid approach                  | Each of the approaches has      | (Sun et al., 2007a)              | (Kao et al., 2019; Lee et al., 2014) |                                        |
Besides the rule and statistical-based approach, hybrid approaches are also followed for syntax error detection, thereby taking the advantages of both approaches. Kao et al. (2019) used NLP and statistical methods to detect collocation errors. Sentences were parsed to find the dependency and POS of every word in the sentence. Subsequently, the collocation was matched through a collocation database to find errors. Similarly, Lee et al. (2014) applied rule-based and n-gram based techniques for judging the correctness of a Chinese sentence. A total of 142 expert-made rules were used to check the potential rule violation in the sentence, while the n-gram method determines the correctness of the sentence.

The semantic error detection has largely carried out by statistical approach using techniques like n-gram methods or machine learning. The use of RNN is quite popular in semantic error detection (Cheng, Fang & Ostendorf, 2017). Zheng et al. (2016) and Yeh, Hsu & Yeh (2016) used an LSTM-based RNN to detect errors like redundant words, missing words, bad word selection, and disordered words. While, Cheng, Yu & Chen (2014) proposed conditional random fields (CRF) models to detect word ordering error (WOE) in textual segments. Zan et al. (2020) proposed syntactic and semantic error detection in the Chinese language by using BERT, BiLSTM, and CRF in sequence. Similarly, Agarwal, Wani & Bours (2000) applied LSTM neural network architecture to make an error detection classifier for detecting two types of error - syntax and semantic errors such as repeated word error, subject-verb agreement, word ordering, and missing verb. For detecting a grammatical error with a long sentence, Rei & Yannakoudakis (2016) proposed a neural sequence labeling framework. The authors found bi-directional LSTM outperforms other neural network architecture like convolution and bidirectional recurrent. Shiue, Huang & Chen (2017) claimed that among the other classifier, the decision tree yields better performance for morphological error and usage error. Yu & Chen (2012) proposed an SVM model for error detection like an adverb, verb, subject, object ordering and usage error, prepositional phase error, and pronoun and adjective ordering error. In (Xiang et al., 2015), it is found that supervised ensemble classifier – Random Feature space using POS tri-gram probability offers better performance for semantic error detection in comparison to other supervised classifiers. Ferraro et al. (2014) saw the different grammatical errors like sentence structure, syntax, and semantic errors as collocation errors. A collocation match in a corpus would able to detect collocation errors. Besides machine learning models, a statistical model based on sequential word pattern mining has been quite effective in detecting grammatical errors (Ganesh, Gupta & Sasikala, 2018). Statistical modeling and machine learning, though easy to implement, are sometimes outperformed by rule-based techniques. In (Lee et al., 2013b; Sun et al., 2007a), it is found that rule-based techniques for detecting grammatical errors yield a better result for the Chinese language.

The choice of error detection technique depends much upon the rules and science of the text language under consideration. Error detection using rule-based techniques demands human expertise in framing the rules. A language with a plethora of possibilities for sentence making leads to difficulty in framing rules to capture the different types of error. Moreover, this technique can be specific to a domain or application context and cannot be generalized.
Unlike rule-based techniques, error detection using machine learning demands a huge dataset, which may not be available for all types of application scenarios. Recently, it is found that most of the syntax and semantic error detection in the text is carried by LSTM, RNN, Sequence to Sequence modeling techniques. But these techniques require corpus with incorrect and their corresponding correct sentence data with appropriate annotation or labeling. The creation of such corpus is a non-trivial task. Moreover, the models do not generalize well. This means if a sentence in the corpus is not large enough, the source sentence for error detection may appear strange to the model. Even though a lot of work has been done in error detection in the Chinese language, but there is a huge lacking of work for semantic error detection for the English language.

Various works have been done for detecting the sentence structure, syntactical and semantic errors in a sentence, but none have been found for assessing the correctness of question framing. Questions are actually textual sentences, but the way they are interpreted in comparison to the other textual sentences requires a different approach for error checking. Comprehending a question generally requires knowing “what is being asked”, “which key concepts are involved”, and “how the key concepts are related in context to the question”. Thus, identifying the error in question framing involves issues like identifying specific ordering of the semantic words (key concepts) and identifying the verbs. The verbs and other grammatical words which relate to the key concepts orchestrate the meaning of the question. Detecting these two is important in interpreting the meaning of the question and subsequently assessing the error or wrong question framing. The characteristic features which differentiate the error checking strategy of questions from other textual sentences are given in Table 2.

Finding or detecting an error in question leads to two possibilities for correction—(a) automatic error correction and (b) recommending correct question. The automatic error correction techniques have not reached their maturity yet. It fails to correct sentences that are complex (logical or conceptual), and furthermore, it cannot align with the intent of the learner. Mostly the automatic error correction fails to correct semantic errors.

The other possibility is recommending the correct question, i.e., suggesting the probable correct questions to the learner against the incorrect input question. This facilitates the learner to navigate through the suggested question to choose the correct question which matches her intended question.

Most of the works on question recommendation are limited to Community Question Answer (CQA), which basically recommends the unanswered question to the user to be

| Table 2 | Differentiating characteristic feature of question in relation to textual sentence. |
|---------|--------------------------------------------------------------------------------|
| **Question** | **Other textual sentence** |
| Subject domains involved is important. | The subject domain is not important. |
| Presence and specific ordering of keywords (semantic words) is significant. | No significance is given to particular words and their ordering and placement. |
| Verb/grammatical words which relates the semantic words carries the entire meaning of the question. | The verb and other grammatical word play important role to the whole sentence instead limiting to specific words. |
answered correctly (Szpektor, Maarek & Pelleg, 2013). The question recommendation is made based on the learner’s dynamic interest (Wang et al., 2017), previous interest (Qu et al., 2009), expertise (Wang et al., 2017; Yang, Adamson & Rosé, 2014), load (Yang, Adamson & Rosé, 2014), user model. Besides the CQA system, the question recommendation is commonly used in a frequently asked question (FAQ) based system, where questions similar or related to user questions are retrieved and recommended from the base. For finding similar questions, cosine similarity (Cai et al., 2017), syntactic similarity (Fang et al., 2017), concept similarity (Fang et al., 2017), and TFIDF, knowledge-based, Latent Dirichlet Allocation (LDA) (Li & Manandhar, 2011), recurrent and convolution model (Lei et al., 2016) are commonly used. Despite our best effort, we did not find work on the correct question recommendation for a given incorrect question.

The only work which is close to our framework is the work done by Giffels et al. (2014). It is a question answering system developed with much focus given on the completeness of the user input question. Mostly factoid-based questions like “wh” questions and true or false questions are accepted in the system. Every time a user inputs a question, it is lexically and syntactically analyzed to find the named entities—what is being asked and what is the subject of the question. The input question strength is calculated as a score based on its completeness. If the score is high, suitable answers are recommended from the base. When the score is less than a threshold, the user is given feedback on restructuring the question, and the entire process cycle is repeated until the input score is high than the threshold. The system has the following two big shortcomings:

- It does not check whether the input question is correct or not. It considers only the question is complete or not.
- Based on the question score, the system gives the feedback. This puts forward a big issue. If the learner lacks knowledge and language skills, she will not be able to frame logical or conceptual questions completely or correctly. This leads to different answers which the learner may not agree with.

To address the issue of checking a question’s correctness, we have proposed a methodology that is more precise and practical. Further, an automatic navigation system is proposed that allows the learner to select the correct question nearly matching to her intent.

**ASSESSING THE CORRECTNESS OF THE LEARNERS’ INPUT QUESTIONS**

In this section, we present the proposed work for assessing whether the learner’s input questions to the query-based learning system are correct or not.

**Theoretical background**

The fundamental concepts that we adopted to assess the correctness of a question are the n-gram and sequential pattern mining. The basics of these concepts are briefed below.
**N-gram**

The n-gram is a sequence of n items adjacent to each other in a string of tokens (text). The items in the string could be letters, syllables, or words. The size of n can be 1 (uni-gram), 2 (bi-gram), 3 (tri-gram), and so on. For example, in the string “the world is a beautiful place”, the possible bigrams are “the world”, “world is”, “is a”, “a beautiful”, and “beautiful place”. Similarly, for the sentence “a document consists of many sentences”, the word-based tri-grams will be “a document consists”, “of many sentences”. The tri-grams can also be overlapping like “a document consists”, “document consists of”, “consists of many”, and “of many sentences”. The same applies to the other higher-level n-grams.

**Sequential pattern mining**

The sequential pattern is a set of items that occur in a specific order (Joshi, Jadon & Jain, 2012; Slimani & Lazzez, 2013). Sequential data patterns reflect the nature and situation of data generation activity over time. The existence of frequent subsequence totally or partially ordered is very useful to get insight knowledge. These patterns are common and natural, for example, genome sequence, computer network, and characters in a text string (Mooney & Roddick, 2013).

Sequential pattern mining (SPM) is the process of extracting items of a certain sequential pattern from a base or repository (Joshi, Jadon & Jain, 2012). Additionally, it helps to find the sequence of events that have occurred and the relationship between them, and the specific order of occurrences. Formally, the problem of subsequence in SPM is described as, for a sequence is an ordered list of events, denoted as \( \alpha_1 \alpha_2 \ldots \alpha_n \). Given two sequences \( P = \langle x_1, x_2, \ldots, x_n \rangle \) and \( Q = \langle y_1, y_2, \ldots, y_m \rangle \), then \( P \) is called a subsequence of \( Q \), denoted as \( P \subseteq Q \), if there exist integers \( 1 \leq j_1 < j_2 < \ldots < j_n \leq m \) such that \( x_1 \subseteq y_{j_1}, x_2 \subseteq y_{j_2}, \ldots, \) and \( x_n \subseteq y_{j_n} \) (Slimani & Lazzez, 2013; Zhao & Bhowmick, 2003).

**Need for using tri-gram based pattern matching**

In this section, we justified the application of n-gram pattern matching and specifically the tri-gram for assessing the correctness of a learner question.

**N-gram based pattern matching for question’s correctness assessment**

Typically, the faults in an ill-framed user question lie in the sentence structure (missing subject or verb/phrase error), syntactic structure (grammatical error like subject-verb agreement, error related to the article, plural, verb form, preposition), and semantic errors (incorrect usage and placement of word).

Domain-specific questions are interrogative sentences that specify entities, concepts, and relations (between themselves) in a particular sequence. The sequential pattern focuses on how the concepts and entities are related and what interrogative meaning can be inferred from them (the question intention). Word collocation, like words around the entities, concepts, relations together, makes word clusters. The link between the different word clusters in sentence subsequences would enable us to get insight into the structural and semantic aspects of a question. In this direction, pattern match for finding the correct word clusters and their sequences could be a prospective approach in the assessment of a question.
The \( n \)-gram language model allows for pattern matching and probability estimation of \( n \)-words in a sentence. The high probability of \( n \)-gram pattern similarity match could lead us to assume that \( n \)-word cluster for a subsequence in a sentence is correct for their syntactic structure and semantic composition. If the entire sentence is split into an ordered sequence of \( n \)-gram subsequences, the aggregated probability estimation of correctness for each \( n \)-gram could lead us to assume the correctness of the entire question. Hypothetically, if we consider the probability estimation of the correctness is a cumulative assessment of individual \( n \)-gram sequences in the question, then which \( n \)-gram should be chosen for the optimum result? We shall try to find the answer to this in the next subsection.

**Tri-gram: the preferred choice for language modeling**

In \( n \)-gram, increasing the \( n \) value would result in clustering an increased number of words as a sequence and thus decreasing the total number of subsequences in a sentence. This leads to an increase in biasness toward similarity pattern matching and thereby decreases the similarity matching probability of diverse sequence patterns. Whereas decreasing \( n \) increases the number of subsequences in a sentence, thereby increasing the probability of similarity match at smaller sentences, but fails to find cohesion among word clusters and hence decreases the probability of accuracy for the larger sentences.

A tri-gram is a perfect capture for the desired features of the sentences and, at the same time, maintaining the optimum complexity factor of the program. While resoluting the sense from a group of words in sequence, it is observed that tri-gram (given one word on either side of the word) is more effective than two words on either side (5-gram). It is also found that increasing or reducing the word on either side of a given word does not significantly make it better or worse in \( n \)-gram sequencing (Islam, Milios & Kešelj, 2012).

**Question’s correctness assessment using tri-gram approach**

In this section, we present the proposed approach for assessing the correctness of the learner question using tri-gram. The method includes building a tri-gram language model that is trained to assess the correctness of a question on Java, and devising a classification method to separate correctly and incorrectly framed questions. The details are described in the following subsections.
**Tri-gram language model generation**

The specific procedures for generating the tri-gram based language model are explained in the following. The process flow of the language model generation is shown in Fig. 2.

**Data collection and corpus preparation**

The language model is designed, trained, and tested on a corpus of sentences. To build the needed corpus, we collected a total number of 2,533 questions on the various topics of Java from books (available as hardcopy and softcopy), blogs, websites, and university exam papers. We adopted both manual and automatic approaches to extract and collect the questions. A group of four experts in the Java language was involved in the manual collection of questions. For automatic extraction, we used a web crawler with a question parser. The crawler, an HTML parsing program designed in Python language, reads the webpage and spawns across other inbound webpages. Using the appropriate regular expression, the expected question sentences were extracted from the parsed pages. The returned texts were then manually verified and corrected, if required, to obtain meaningful questions.

To test the efficiency of the proposed method in rightly identifying a correct and incorrect question, we needed a set of wrong questions as well. A number of incorrectly framed questions were collected from learners' interaction with the online learning portals and institutional online learning system and questions asked by the students in the class. The incorrect questions contain grammatical errors (sentence structure and syntactic errors) and semantic errors.

The details of the question datasets are as following:

- Number of questions in training dataset: 2,533 (all correct)
- Number of questions in testing dataset: 634
- Number of correct questions in testing dataset: 334
- Number of incorrect questions in testing dataset: 300

**Data preprocessing for language model generation**

As the collected questions consisted of many redundancies and anomalies, we preprocessed them to develop a suitable language model for questions. Text preprocessing typically includes steps like stopword removal, lemmatization, etc. Stopwords are frequently used words like “I”, “the”, “are”, “is”, “and”, etc., which provide no useful information. Removing these from a question optimizes the text for further analysis. However, sometimes certain domain-specific keywords coincide with the stopwords, removal of which may result in a loss of information from the questions. Therefore, we modified the list of stopwords by removing the domain-specific keywords from the Natural Language Toolkit (NLTK https://www.nltk.org/) stopword list to avert eliminating the required stopwords. The modified NLTK stopword list is used to remove stopwords from the questions, excluding those which are meant for the Java language.

Each question is broken down in the form of tokens using the regular expression tokenizer, which is present in the NLTK library. Each of these tokens is converted into
their stem (root word) form using the Wordnet Lemmatizer to reduce any inflectional form of words. The steps for preprocessing an input question are shown in Fig. 3.

**Language modeling**

The preprocessed questions are broken down into sets of distinct uni-, bi-, and tri-gram sequences. The uni-gram set is built on individual tokens in the questions. Whereas the bi- and tri-grams are formed using overlapping two- and three-token sequences, respectively, as shown in Fig. 4.

The respective count of each n-gram occurrence is obtained from the question corpus. Along with the count, based on the relative occurrences in the corpus, the unconditional log probabilities of each uni-gram, as represented by Eq. (1), and conditional log probabilities of each bi- and tri-gram, as represented by Eqs. (2) and (3), respectively, are calculated.

\[
P(w_1) = \log \left( \frac{C(w_1)}{C(w_n)} \right)
\]  

(1)

Where \(w_n\) represents the words in the corpus and \(c(w_n)\) returns the count of the total number of words in the corpus.
The log probabilities in Eqs. (1) and (2) allow transforming higher fractional probability values to lower ones, which are easy to be used in the computation. A sample representation of the language model is shown in Table 3. The entire language model derived from the question corpus is saved in ARPA (http://www.speech.sri.com/projects/srilm/manpages/ngram-format.5.html) format.

**Classifying correct and incorrect questions**

The correctness of a question is estimated based on its syntactical and semantic aspects and accordingly is classified as correct or incorrect. The complete process of identifying correct and incorrect questions is pictorially shown in Fig. 5.

**Preprocessing the learners’ input questions**

The input questions from the learner are preprocessed to remove the stopwords and the irrelevant words. Also, lemmatization is carried over the input question.
After preprocessing, the question is broken down into overlapping tri-gram sequences. Each tri-gram sequence is estimated for probability by maximum likelihood estimation (MLE) from the language model. If a tri-gram sequence of the question is not present in the language model, it will lead to zero estimation. However, though the entire tri-gram sequence may not occur in the language model, a partial word sequence, a lower-order n-gram (bi-gram) of it, could be valid. The Backoff approach (Jurafsky & Martin, 2013; Brants et al., 2007) is considered for tri-grams to take into account of sequence which counts to zero. The tri-gram sequences which estimate to zero are further estimated for their bigrams.

The probability of each tri-gram ranges from 0 <= P <= 1. A higher probability refers to more correctness and higher occurrence. The entire probability of syntactic correctness of the sentence can be obtained as the addition of probability of each tri-gram in the question in Eq. (5), where k is the number of tri-grams in the question and P_i is the probability of the i^{th} tri-gram sequence in the sentence.

\[
E_{sy} = \frac{1}{k} \sum_{i=1}^{k} P_i
\]

Probability estimation for question correctness based on semantic aspect
The correctness of question semantic is assessed by estimating the validity of individual overlapping tri-gram sequences of the sentence. The validity of the tri-gram is assessed by the probability estimation of each tri-gram sequence in question found matches in the language model, as shown in Eq. (6). The semantic correctness of a question is estimated on the full similarity match of each tri-gram sequence. More the number of subsequences of the question sentence matches the language model, more is the chance of the question being semantically correct. The overlapping tri-gram sequences reflect the cohesion among words in the sentence subsequences. Thus, increasing the number of matching of the tri-gram sequences establishes a higher probability of semantic accuracy of the question. The semantic correctness of the question is calculated as the summative average of probabilities of each tri-gram sequence in the sentence is shown in Eq. (7).

\[
P(w_3|w_1, w_2) = \begin{cases} 
1, & \text{if } P(w_3|w_1, w_2) > 0 \\
0, & \text{if } P(w_3|w_1, w_2) = 0
\end{cases}
\]

\[
E_{sm} = \frac{1}{k} \sum_{i=1}^{k} P_i
\]
Classification

The correctness of a question is calculated by Eq. (8), where $E_{sy}$ and $E_{sm}$ are the probability estimates of syntactical and semantic correctness of the sentence, respectively. A syntactically correct question has $E_{sy} = 1$, and $E_{sm} = 1$ for semantically correct. Hence, the standard score for a correct question is $1 + 1 = 2$. Thus the degree of correctness ($C_d$) of the question with respect to the complete correctness (i.e., 2) is assessed by adding the calculated probability estimates $E_{sy}$ and $E_{sm}$ and subtracting from 2. We considered the question is correctly structured, if $C_d \leq 20$; otherwise, the framing of the question is not correct.

$$C_d = (2 - (E_{sy} + E_{sm}) \times 50$$

Experiment and performance evaluation for question’s correctness assessment

The evaluation of the performance measure of the proposed approach for assessing the correctness of the learner question is done on a corpus of 634 annotated questions, where 52% of questions are correctly framed. The performance of the tri-gram approach for classifying questions as correct or incorrect is measured based on the metrics: true positive, true negative, false negative, and false positive, and the performance measures: Accuracy, Precision, Recall, F1-Score, as shown in Table 4.

In the experiment, we attempted to distinguish between correct and incorrect questions based on the probabilistic calculation proposed by our approach. The experimental results show that our method fails to classify 50 of these questions correctly. Out of these 50 questions, 32 were correct questions but are identified as incorrect. Further analysis of these false-negative questions reveals that after preprocessing and stopword removal, the length of most of the questions is reduced to less than three. These questions fail to generate any tri-grams to perform the probabilistic calculation. So, these questions by convention get marked as incorrect. Some of these false-negative questions even belong to domains that are not present in the training dataset. As a result, the proposed method fails to identify these questions correctly. The other set of incorrectly classified questions comprises incorrect questions which are marked as correct. The false-positive questions primarily have misplaced punctuation marks which results in the structure of the incorrect question identical to the correct questions in the training set. They form tri-grams or bi-grams, which perfectly match the tri-grams or bi-grams from the language model and

| Performance metric | Value | Performance measure | Value |
|--------------------|-------|----------------------|-------|
| True positive      | 282   | Accuracy             | 0.9211|
| False positive     | 18    | Precision            | 0.9400|
| True negative      | 302   | Recall               | 0.8980|
| False negative     | 32    | F1-Score             | 0.9188|

Table 4 Performance measures of the proposed approach.
render a high probabilistic score for the question. A margin of 8% error shows the efficiency of the proposed approach.

The efficacy of the tri-gram model approach was compared with other n-grams. The models were trained over the same question dataset to keep the experiment bias-free. Figure 6 shows a comparison of the accuracy measures obtained for each n-gram approach over the same statistical calculation. It is evidently seen that the accuracy of tri-gram is far better than other n-grams. The accuracy decreases with the increasing value of n in n-gram. It leads to biased higher-order word sequence pattern search and fewer options for pattern assessment at lower orders. This causes restricted pattern search and a decrease in accuracy. Similarly, decreasing n leads to word sequence pattern search at lower order, which restricts the probability of correctness of the word sequences at higher orders. This typically reduces the accuracy. The comparative experiment thus concludes that the use of the tri-gram model for question assessment leads to better assessment results.

The result of the proposed approach is compared with the results of another similar work by Ganesh, Gupta & Sasikala (2018), in which the authors applied a tri-gram based approach to detect an error in English language sentences. Table 5 shows the result comparison in terms of four assessment metrics. From the table, it is evident that the accuracy of our proposed approach is much better. However, the precision of both approaches is the same. This establishes the true positive and true negative identification cases are better in our approach for detecting the errors and thus the correctness or incorrectness of the question sentences.

![Figure 6](https://DOI: 10.7717/peerj-cs.532/fig-6)

| Table 5 | Comparative results of the proposed approach and the solution given in (Ganesh, Gupta & Sasikala, 2018). |
|---------|--------------------------------------------------------------------------------------------------|
| Proposed approach (%) | Result of (Ganesh, Gupta & Sasikala, 2018) (%) |
| Accuracy | 92.11 | 83.33 |
| Precision | 94.00 | 94.11 |
| Recall | 89.80 | 80.00 |
| F1-Score | 91.88 | 86.48 |
GUIDING THE LEARNER TO THE PROBABLE CORRECT QUESTION

In the previous section (“Assessing the Correctness of the Learners’ Input Questions”), we checked if the question given as input by the learner to the query-based learning system is syntactically and semantically correct or not. If the question is not correct, we guide the learner to the probable correct question that she actually intended to ask through one or multiple steps of question suggestions. The detailed methodology and framework of the proposed work are discussed in the following subsections.

Similarity-based recommendation for mitigating incorrect learner question

Computationally auto-correcting the incorrectly framed question is one of the acclaimed ways followed in literature. But the success is limited and restricted to correcting only a few types of errors or mistakes. The typical mistakes a learner commits while articulating a question are shown in Fig. 7. For instance, inappropriate word selection may not reflect the exact intention of the learner. Similarly, insufficient keywords may not express the intended concept.

In regard to these, except for grammatical and sequential ordering errors, auto-correction for other types of errors is not possible. The other way around, the problem is suggesting correct questions to the learner which are near to what she intended to ask. Suggesting correct questions which are similar to information and morphological structure to the given question could lead to having a chance that learner may found the right question which she intends to ask. Considering the information like the concepts and functional words which are used in compiling the question is best of her knowledge in the current information seeking situation, the learner could be recommended appropriate questions which are aligned to/with the information they are seeking for. Thus, suggesting correct questions in contrast to the incorrect question imposed by the learner is through similarity-based recommendation is an effective way to overcome the incorrect question problem.

Issues in similarity-based recommendation of questions

Cosine and Jaccard similarity techniques are the two text-based similarity approach which has been widely incorporated for finding similar text (Sohangir & Wang, 2017; Amer & Abdalla, 2020). But these approaches, when applied to question-based corpus for
identifying similar question text, lead to the recommendation issues, as discussed in the following subsections.

**Information overload**

Text similarity based on word match searches for similarity for every occurring word in the source sentence—incorrect question text for an exact match in the questions present in the question corpus. The needful comparison based on matching word occurrence among the sentences returns similar text. Since the question framing is incorrect, taking a part of the entire sentence which seemingly founds to be correct and conveys the learner’s intent, could lead to a better similarity match. However, the prevailing constraint and limitations of NLP fail to analyze and identify the parts of the source sentence, which are correct as per learner intention. Failing to determine this leads to ambiguity in identifying the parts of a sentence that are to be taken correctly for similarity match. Without this knowledge, the similarity search is done for each occurring word (assuming they are correct as per the learner intent) in the question against the questions in the corpus lead to a huge set of information. For example, a learner questions on Java with incorrect word ordering and missing words like “What different are interface implement”, when runs for similarity match like Jaccard similarity on a question corpus returns a lot of information, as shown in Table 6. With this amount of information, the learner may get confused and lost.

**Diverse information**

A learner, when composing a question, intends to seek information limited to a particular topic(s). Text similarity based on word match searches for similarity for every occurring word in the source sentence for an exact match into the question corpus. For similarity measurement, weightage is given to word occurrence frequency rather than on their subject domain relevancy. No consideration is given to individual tokens belonging to a topic of a domain. Since a question is made up of functional words (noun or verb) along with concepts (domain keywords), the word match found for every functional word in the corpus leads to different questions having different topics which the learner does not intends to seek. This results in questions that are beyond the search topic boundary, leading to diversification of information. For example, the similarity search for an incomplete question like “access modifier in Java” using Jaccard similarity returns questions of different topics, as shown in Table 7. Figure 8 shows the share of the number of questions belonging to different topics for the given similarity recommendation. A large number of questions are on a different topic than that of the input question. This may put the learner in jeopardy and confusion. Conclusively, the similarity match on functional words of the source question in the corpus may result in diversification instead of convergence.

**Biased to exact word match**

While framing a question, keywords and functional words are integrated and sequenced in an appropriate manner to make meaning out of the question. The use of these words by
the learner is the natural outcome of the learner’s knowledge and communication skill. And as a reason, lack of a learner’s expertise does not assure the correctness of question framing. The similarity assessment technique performs an exact word match. This will return only those questions, the words of which are exactly matched (word-by-word) with the learner’s input question. This results in obscuring many other similar questions, which are having different words but similar or near to similar meanings. And thus, many of the questions having similar meanings but having different word construction are ignored, resulting in poor efficiency.

| Returned similar question                                      | Returned similar question                                      |
|----------------------------------------------------------------|----------------------------------------------------------------|
| 1 What is the need for an interface?                          | 31 What do you mean by interface?                             |
| 2 What are the properties of an interface?                    | 32 How interface is different from abstract class?            |
| 3 What is interface?                                          | 33 What are the different types of applet?                    |
| 4 What are the methods under action interface?                | 34 Which methods of serializable interface should I implement? |
| 5 What are the methods under window listener interface?       | 35 What is an externalizable interface?                        |
| 6 What is Java interface?                                     | 36 What is vector? how is it different from an array?         |
| 7 What are the advantages of interfaces                       | 37 What are the methods under action interface?               |
| 8 What interfaces is needed                                    | 38 What are the methods under window listener interface?      |
| 9 What is an interface?                                       | 39 What is Java interface?                                    |
| 10 What are interfaces?                                       | 40 What are the advantages of interfaces?                     |
| 11 What are constructors? how are they different from methods? | 41 What are constructors? how are they different from methods? |
| 12 How interface is different from class                       | 42 Is it necessary to implement all methods in an interface?  |
| 13 What is an interface? how is it implemented?                | 43 If you do not implement all the methods of an interface what specifier should you use for the class? |
| 14 What are different modifiers?                              | 44 What is difference between interface and class?            |
| 15 Is it necessary to implement all methods in an interface?  | 45 What is difference between package and interface?          |
| 16 How interface is different from abstract class?            | 46 What do you mean by interface?                             |
| 17 What are different comments?                               | 47 What do you mean by interface?                             |
| 18 What are different modifiers?                              | 48 What is the nature of methods in interface?                |
| 19 Is it necessary to implement all methods in an interface?  | 49 What do you know about the file name filter interface?    |
| 20 How interface is different from abstract class?            | 50 What is a nested interface?                                |
| 21 If you do not implement all the methods of an interface while implementing what specifier should you use for the class? | 51 Which classes implements set interface?                     |
| 22 What must a class do to implement an interface?             | 52 What is the interface of legacy?                           |
| 23 What interface must an object implement before it can be written to a stream as an object? | 53 What is different between iterator and listiterator?       |
| 24 What is applet stub interface?                              | 54 What are different collection views provided by map interface? |
| 25 How interface is different from a class.                   | 55 What is comparable and comparator interface?               |
| 26 What is an interface?                                      | 56 What will happen if one of the members in the class doesn’t implement serializable interface? |
| 29 What is interface?                                         | 57 What is serializable interface in Java?                    |
| 30 How interface is different from class?                     | 58 What is externalizable interface?                          |

Table 6 Similar questions returned by Jaccard similarity for the learner question “what different are interface implement”.

Pal et al. (2021), PeerJ Comput. Sci., DOI 10.7717/peerj-cs.532
| Recommended Question                                                                 | Topic               | Recommended Question                                                                 | Topic               |
|------------------------------------------------------------------------------------|---------------------|--------------------------------------------------------------------------------------|---------------------|
| 1. What are the features of java language?                                        | Basics              | 52. Briefly discuss the features of java.                                             | Basics              |
| 2. What is the need for java language?                                             | Basics              | 53. What is jvm? Explain how java works on a typical computer?                        | Basics              |
| 3. How java supports platform independency?                                        | Basics              | 54. List out at least 10 difference between java & c++.                               | Basics              |
| 4. Why java is important to internet?                                              | Basics              | 55. Explain, why java is the language of choice among network programmers.             | Basics              |
| 5. What are the types of programs java can handle?                                 | Basics              | 56. Write a java program to accept two strings and check whether string1 is a sub string of string2 or not. | String              |
| 6. What are the advantages of java language?                                       | Basics              | 57. Explain the relevance of static variable and static methods in java programming with an example. | Class & Object      |
| 7. Give the contents of java environment (jdk).                                    | Basics              | 58. Describe the syntax of single inheritance in java.                                | Inheritance         |
| 8. Give any 4 differences between c and java.                                      | Basics              | 59. Name at least 10 java api class you have used while programming.                  | Package             |
| 9. Give any 4 differences between c++ and java.                                    | Basics              | 60. Significance of interface in java?                                                | Interface           |
| 10. What are the different types of comment symbols in java?                       | Basics              | 61. Do class declaration include both abstract and final modifiers?                   | Inheritance         |
| 11. What are the data types supported in java?                                     | Data type & variable| 62. Do java supports operator overloading?                                            | Operator            |
| 12. How is a constant defined in java?                                             | Data type & variable| 63. Does java support multithreaded programming?                                      | Thread              |
| 13. What are the different types of operators used in java?                        | Operator            | 64. Do java has a keyword called finally?                                             | Exception handling  |
| 14. What are the types of variables java handles?                                  | Data type & variable| 65. Java does not provide destructors?                                                | Class & Object      |
| 15. How is object destruction done in java?                                        | Class & Object      | 66. Do the vector class is contained in java.util package?                            | Package             |
| 16. What is a string in java?                                                      | String              | 67. Does private modifier can be invoked only by code in a subclass?                  | Inheritance         |
| 17. What are the different access specifiers available in java?                    | Package             | 68. Does all files are included in the java.io package?                               | Package             |
| 18. What is the default access specifier in java?                                  | Package             | 69. Java supports multiple inheritance?                                               | Interface           |
| 19. What is a package in java?                                                     | Package             | 70. Do java.applet is used for creating and implementing applets?                    | Applet              |
| 20. Name some java api packages                                                    | Package             | 71. How applets are programs that executes within a java enabled web browser?         | Applet              |
| 21. Explain the features of java language.                                        | Basics              | 72. Is java a high-level language?                                                    | Basics              |
| 22. Compare and contrast java with c.                                              | Basics              | 73. What are byte codes and java virtual machine?                                     | Basics              |
| 23. Compare and contrast java with c++.                                            | Basics              | 74. Explain about java variables.                                                     | Data type & variable|
| 24. Discuss in detail the access specifiers available in java.                    | Package             | 75. Differentiate between java applications and java applets.                         | Applet              |
| 25. Explain the different methods in java.util. arrays class with example.        | Array               | 76. What is a thread in java?                                                         | Thread              |
| 26. How multiple inheritance is achieved in java?                                 | Package             | 77. Explain the meaning of public static and void modifiers for the main() method in a java program. | Class & Object      |
| 27. How does java handle integer overflows and underflows?                        | Data type & variable| 78. Explain about inheritance in java.                                                | Inheritance         |
| Recommended Question | Topic | Recommended Question | Topic |
|-----------------------|-------|-----------------------|-------|
| 28 How java handle overflows and underflows? | Data type & variable | 79 Explain about polymorphism in java. | Inheritance |
| 29 What are the threads will start when you start the java program? | Thread | 80 Explain the structure of a java program. | Basics |
| 30 What is java math class? List 10 method with syntax. | Package | 81 What are the steps for implementing a java program? | Basics |
| 31 Explain java data types? | Data type & variable | 82 Explain java data types. | Data type & variable |
| 32 What is java array? | Array | 83 Write the different operators in java. | Operator |
| 33 Write short notes on java method with syntax and example. | Class & Object | 84 What are the control statements available in java? | Control structure |
| 34 What is java variable? Explain the different types of variable. | Data type & variable | 85 What are the looping statements available in java? | Control structure |
| 35 Explain garbage collection in java. | Class & Object | 86 What are the different string methods available in java? | String |
| 36 There is no destructor in java, Justify. | Class & Object | 87 What are the different string buffer methods available in java? | String |
| 37 What are java classes? | Class & Object | 88 What is the use of this keyword in java? | Class & Object |
| 38 How we can create java classes. | Class & Object | 89 What is the use of super keyword in java? | Inheritance |
| 39 How we can create java objects? | Class & Object | 90 What is the use of finally keywords in java? | Exception Handling |
| 40 What is java string? | String | 91 Explain about different class modifiers. | Package |
| 41 How we can initialize and create java string explains with 10 methods? | String | 92 Explain about different constructor modifiers. | Class & Object |
| 42 Explain java character class with suitable example and methods | Package | 93 Explain the use of method modifiers. | Inheritance |
| 43 What is inheritance in java? Explain all its type with example. | Inheritance | 94 Write short notes on different java api packages. | Package |
| 44 Explain interface in java. How do interfaces support polymorphism? | Interface | 95 Write short notes on different exception types available in java. | Exception handling |
| 45 Explain package in java. List out all packages with short description. | Package | 96 Explain about catch, throw and try statement in java. | Exception handling |
| 46 What is java interface. | Interface | 97 Why does java not support destructors and how does the finalize method will help in garbage collections? | Class & Object |
| 47 Explain exception handling in java. | Exception handling | 98 Write short notes on access specifiers and modifiers in java. | Package |
| 48 What led to the creation of java? | Basics | 99 Discuss the working and meaning of the “static” modifier with suitable examples. | Class & Object |
| 49 What are the steps to be followed for executing a java program? | Basics | 100 Explain in detail as how inheritance is supported in java with necessary example. | Inheritance |
| 50 Explain the data types available in java | Data type & variable | 101 Explain in detail as how polymorphism is supported in java with necessary example | Inheritance |
| 51 What are the different types of operators in java? | Operator | 102 What are the java apis used for package? | Package |
Proposed framework for correct question suggestion to the learner

Considering the above-mentioned three problems, we have adopted the soft cosine technique to find similar sentences. The similarity matching is augmented by question selection and iteration pass. We propose a similarity assessment framework for suggesting the correct question for a given incorrect question on a particular domain. The framework consists of three phases of working, as discussed below. The framework is shown in Fig. 9, while the process flow is shown in Fig. 10.

Selecting questions with similar concepts

The selection of questions with similar concepts limits the search boundary, and hence the diverse information issue can be addressed. Learners impose questions using the best of their knowledge. This makes them use concepts that are more aligned with the information they are trying to seek. Though not all the concepts which are articulated in the question are rightly chosen, the probability of having the required concept in the question also persists. And thus, claiming all questions from the corpus having the same concept(s) as present in the source question could increase the likelihood of finding the right intended question. This also reduces the probability of recommending questions that are completely on a different topic(s) or concept(s) not relating to the concept(s) present in the source question. As a reason, the concept-wise selection of questions will reduce the diversification of information recommendation.
Similarity assessment and correct question recommendation

A learner may compose an incorrect question due to the following three reasons:

a) There are insufficient keywords to express the question.
b) Insufficient number of words used to express the question.
c) The selection of words and their usage may be incorrect.

In all the cases, we need to find the alternative questions closest to the learner’s intended question. For estimating the similarity, we suggested looking for the questions that have the same or similar word features as the learner’s question. A hard similarity (word to word) match for word features between the incorrect and alternative question reduces the chances of getting a more accurate alternative. Moreover, conducting a hard similarity search in the word feature space of the correct question, the source question’s inappropriate words would be of no use. Rather a soft similarity (synonym or close related words) match would give a high probability of finding the questions that are meaningfully aligned to the learner’s intent. To address the similarity match problem and to find the correct question, we applied soft cosine measures. Soft cosine allows finding the questions that are significantly similar in terms of the semantic matching, irrespective of the exact word match.

The similarity measure \( \text{sim}(f_i, f_j) \) in soft cosine calculates the similarity for synonym or relatedness between the features \( f_i \) and \( f_j \) of the vectors under consideration. Here, the vector is a question, and the words of the question represent its features. A dictionary
approach like WordNet::Similarity is being used to calculate the similarity (or relatedness) among the features (Sidorov et al., 2014).

From the n-dimensional vector space model’s perspective, the soft cosine measures the semantic comparability between two vectors. It captures the orientation (the angle) between the two vectors. But unlike cosine similarity, the features are projected in an n-dimensional space so that similar features are close by with very less angle difference. This causes the meaningfully similar words (features) of vectors (questions) to have minimal angle differences (Hasan et al., 2019), as shown in Fig. 11. The equation for soft cosine is given in Eq. (9).

\[
\text{Soft cosine}(p, q) = \frac{\sum_{i,j} S_{ij}p_{i}q_{j}}{\sqrt{\sum_{i,j} S_{ij}p_{i}p_{j}} \sqrt{\sum_{i,j} S_{ij}q_{i}q_{j}}}
\]  \hspace{1cm} (9)

Where, \( S_{ij} \) is the similarity between the features \( i \) and \( j \), and \( p \) and \( q \) are the input question and the correct question, respectively.

**Iteration and question selection**

To overcome the issue of information overload, ten questions whose similarities are found more than 50% in relation to the source question text are enlisted to choose by the learner. This allows the learner to focus much on what he is actually seeking rather than getting overwhelmed by the huge information which would have been recommended otherwise. Since the approach is probabilistic, chances are there that no right question which is close to learner intention is found in the list. In such a case, selecting a question from the recommended list nearer to the question which learner intends to seek would allow the system to have better-informed data. The learner selected questions that, in turn, act as a seed for further similarity search. Considering the selected question (seed question) as new input for further similarity search would actually converge the search boundary and increase the homogeneity of information. This will reduce diversification. With every recommendation pass, the degree of concept-wise similarity increases, which, in turn, increases the range of similar questions. This makes the question suggestion to shift closer to the learner’s intention. The complete process is presented in Algorithm 1.
Algorithm 1 Finding the correct question as per learner intent.

Input: Incorrect question Wq, Corpus crp

Output: The intended question

Label 1: concepts[] = get_concept(Wq)
Selected_question[] = search_question(crp, concepts)
Similar_correct_question[] = soft_cosine_similarity(Selected_question, Wq)

for q in similar_correct_question
    similarity = score_similarity(q)
    if similarity > 0.50 then
        print q
    end if
end for

print "input the question and abort/search"
input q, status

if status == "Abort" then
    print q, "is the intended question"
else if
    Wq = q
    goto Label 1
end if

Experiment for correct question suggestion

Experimental procedure

For experimentation and performance analysis, the proposed methodology for similarity assessment and recommendation of correct question is implemented as a web-based client/server model, as shown in Fig. 12.

The server contains the web application (WebApp) with the requisite HTML and Python file, Flask (https://flask.palletsprojects.com/en/1.1.x/) framework, and Python (version 3.8). Flask is a web application microframework that allows to delegate web pages over the network and handle learner’s input requests. The framework is glued as a layer to Python for executing the processes. The model is implemented in Python and is deployed in WebApp as a Python file. Further, the learner’s different interactions with
the system are stored as the experimental data in the SQLite database, which comes default with Python.

The web server is connected to the client devices over the internet or LAN to exchange HTTP Requests and HTTP Responses. And, the learner (client) interacts with the model through the webpage, as shown in Fig. 13. The reason behind choosing this web model for the experiment is as follows:

- Python programs allow for text-based interaction, which disinterests learners, making them less attentive. This causes lacking full involvement of the learner. In contrast, a web-based model gives a graphical interface for interaction and thus better involvement of the learner.

- Since the experiment involves many learners, a web-based model allows them to participate in the experimentation from anywhere and anytime. This gave the learner more freedom to choose the place and time of their own to take part in the experimentation. Moreover, this web model allows multiple candidates to simultaneously participate in the experiment from different client devices while the experimentation result is getting stored centrally.

Selection of the questions based on the concept, and followed by similarity assessment, is carried out in the server. Three similarity assessment techniques—soft cosine, Jaccard, and cosine similarity are used to find the intended correct questions from the corpus. These three techniques are followed in parallel for assessing their performance for the given incorrect input questions. For this experiment, we used the complete training corpus (i.e., 2,533 questions).

To select the probable correct question from the recommend similarity list, a threshold of 0.5 is considered as the minimum similarity score for soft cosine, while 0.2 is considered for Jaccard and cosine. It was found that Jaccard and cosine similarity techniques returned either no or very few (one or two) similar questions, which were not suitable for carrying out the experiment. Further, in some cases, while searching for similar
questions to the given incorrect question, the same question is iteratively returned for each consecutive pass. As a reason, in the cases of Jaccard and cosine, the threshold for similarity score is reduced to a lower value of 0.2. This gave some outputs needed to carry out the experiment and compared to the result of soft cosine.

**Learner verification**

The performance of the framework for similarity-based recommendation to find the intended question was verified by manual assessment. The assessment was carried by a group of learners. A total of 34 students of the CSE department at Bengal Institute of Technology, studying Java in their 6th semester of the B.Tech degree program, were selected. The students chosen were low scorers in the subject. The rationale behind choosing these students was that we wanted to select learners who are aware of the Java language and its terminology but are neither expert nor good in the subject. This made them suitable candidates as they were susceptible to compose incorrect questions.

Each student was instructed to inputs approximately three incorrect questions, totaling 100. Corresponding to each question, three recommendations are made using the soft cosine, Jaccard, and cosine similarity techniques, as shown in Fig. 13. If the student found the correct intended question, the iteration was stopped for the respective similarity technique. If the intended question was not found in the recommended list, the student chose a question from the list as a seed question that was close to the intended question, and another iteration or pass was followed. If the intended question was not found within three passes, the recommendation process for the respective individual similarity technique was stopped. The purpose of using three similarity techniques is to make a comparison and find the best performance among the three.

**Result and analysis**

**Accuracy**

The learner input and feedback on a total of 100 incorrect questions are shown in Table 8. The learner acceptance result of finding the intended correct question against the incorrect input question is summarized and shown in Fig. 14. The summarization is made on the basis of whether the learner finds the intended question or not for each of the three similarities-based recommendations.

Based on learner input and the system feedback, the framework is evaluated for the accuracy metric. The accuracy is an intuitive performance measure, a ratio of correct observation made to the total observation made. The accuracy is defined in percentage by Eq. (10).

\[
\text{Accuracy} = \frac{A \times 100}{B}
\]  

Where,

- A is the number of observations made where the learner finds the correct intended question.
- B is the total number of questions taken for observation.
Table 8  Similarity recommendation against learner questions.

| User Input                                                                 | Error type | Intended question                                                                 | Soft Cosine No. of Pass (Score) | Cosine No. of Pass (Score) | Jaccard No. of Pass (Score) |
|---------------------------------------------------------------------------|------------|----------------------------------------------------------------------------------|-------------------------------|---------------------------|----------------------------|
| what difference interface                                                | IS         | What is the difference between abstract class and interface?                     | 1(0.63)                      | 1(0.51)                   | NF                         |
| define method in subclass with same name                                  | IS         | It is not possible to define a method in the subclass that has the same name     | 1(0.66)                      | 1(0.5)                    | NF                         |
| java not have destroy and how garbage collect                            | EG         | Why does Java not support destructors and how does the finalize method will help | 2(0.61)                      | NF                        | NF                         |
| how to overload                                                           | IS         | What is method overloading? Explain with example.                                | 2(0.6)                       | NF                        | NF                         |
| why main public                                                           | IS         | Why is main method assigned as static?                                           | 2(0.75)                      | 2(0.6)                    |                            |
| object stored reach                                                       | EG         | When an object is stored are all of the objects that are reachable from that     | 1(0.91)                      | NF                        | NF                         |
| what mechanism used for a single thread at a time                        | EG         | What is the mechanism defined by java for the resources to be used by only one   | 1(0.59)                      | 2(0.41)                   | 1(0.36)                    |
| applets talk on web page                                                  | IS         | How can I arrange for different applets on a web page to communicate with each   | 1(0.57)                      | NF                        | 1(0.42)                    |
| show try catch throw                                                      | IS         | Write a Java program which illustrates the try catch throw and throws and finally | 1(0.64)                      | NF                        | NF                         |
| why thread synchronization needed                                         | IS         | Describe the need of thread synchronization. How is it achieved in Java           | 2(0.51)                      | 2(0.33)                   | 2(0.37)                    |
| access modifiers in java                                                  | IS         | Explain access modifiers and access controls at class and package level in Java. | 1(0.58)                      | NF                        | 1(0.25)                    |
| difference between exceptions                                            | IS         | What is difference between user defined exceptions and system exceptions?        | NF                           | 1(0.37)                   | NF                         |
| inbuilt exceptions in class                                              | IS         | Explain with example any three inbuilt exceptions and any three inbuilt methods  | 1(0.51)                      | NF                        | NF                         |
| class extends another class                                               | IS         | If my class already extends from some other class then what should I do, if I      | 1(0.72)                      | NF                        | NF                         |
| how to handle exception                                                   | IS         | want an instance of my class to be thrown as an exception object?                |                              |                           |                            |
| if we do not initialize variables                                         | IS         | What happens if you do not initialize an instance variable of any of the          | NF                           | NF                        | NF                         |
| Inheritance hierarchy in AWT.                                             | EG         | Draw the inheritance hierarchy for the frame and component classes in AWT.        | 1(0.53)                      | 1(0.37)                   | NF                         |
| which specifier to use while all not interface implement                  | EG         | If you do not implement all the methods of an interface while implementing what   | 1(0.75)                      | NF                        | 1(0.5)                     |
| first value of array elements                                             | IS         | specifier should you use for the class?                                          |                              |                           |                            |
| difference between two types programming language                        | IS         | What will be the default values of all the elements of an array that are defined  | 1(0.56)                      | NF                        | 1(0.33)                    |
| we change throws when override                                            | EG         | What is the difference between an object-oriented programming language and        | 1(0.68)                      | 1(0.41)                   | NF                         |
| name of object with own lifecycle                                        | IS         | object-based programming language?                                              |                              |                           |                            |
| boolean value operators                                                   | IS         | Can we modify the throws clause of the superclass method while overriding it in    | 2(0.55)                      | NF                        | NF                         |
| From main call and check string palindrome or not                         | IS         | the subclass?                                                                    |                              |                           |                            |
| methods String available under Name class some. Buffer                    | EG         | What is it called where object has its own lifecycle and child object cannot       | 1(0.52)                      | NF                        | NF                         |
| name of object with own lifecycle                                        | IS         | belong to another parent object?                                                 |                              |                           |                            |
| boolean value operators                                                   | IS         | Which of the operators can operate on a Boolean variable?                        | 3(0.57)                      | NF                        | 1(0.4)                     |
| From main call and check string palindrome or not                         | IS         | Write a method that checks if a string is a palindrome. Call your method from     | 1(0.57)                      | NF                        | 1(0.5)                     |
| methods String available under Name class some. Buffer                    | EG         | the main method.                                                                 |                              |                           |                            |

(Continued)
| User Input                                         | Error type | Intended question                                                                 | Soft Cosine No. of Pass (Score) | Cosine No. of Pass (Score) | Jaccard No. of Pass (Score) |
|--------------------------------------------------|------------|-----------------------------------------------------------------------------------|-------------------------------|---------------------------|-----------------------------|
| high power file copy                             | I₈         | Which streams are advised to use to have maximum performance in file copying?     | NF                            | NF                        | NF                          |
| compare different controls for visibility         | I₈         | Explain the different visibility controls and also compare with each of them.    | 1(0.54)                       | 1(0.51)                   | 1(0.66)                     |
| use reflection to build array                     | I₈         | How to create arrays dynamically using reflection package.                       | 1(0.59)                       | NF                        | NF                          |
| voice message with playMessage method             | E₉         | Develop a message abstract class which contains playMessage abstract method.      | NF                            | NF                        | NF                          |
| all methods of object class                       | I₈         | Explain the different methods supported in Object class with example.             | 2(0.53)                       | 2(0.33)                   | 2(0.42)                     |
| special style of text example                    | E₉         | How do achieve special fonts for your text? Give example.                        | 1(0.64)                       | 1(0.54)                   | 1(0.42)                     |
| keep integer overflow                             | I₈         | How does Java handle integer overflows and underflows?                          | 1(0.65)                       | 1(0.47)                   | NF                          |
| thread start initial                              | I₈         | When a thread is created and started what is its initial state?                 | 1(0.56)                       | 1(0.47)                   | NF                          |
| shift operation in short circuit                  | E₉         | Explain short circuited operators and shift operators                            | 1(0.59)                       | 1(0.31)                   | NF                          |
| what are different interface implement            | I₈         | Describe different forms of interface implementation with their syntax declaration. | NF                            | NF                        | NF                          |
| all ways to call method                           | I₈         | What are the different ways of calling a static method from a program?           | 1(0.52)                       | NF                        | NF                          |
| java program to create person from class          | I₈         | Consider a class person with attributes firstname and lastname. Write a Java program to create and clone instances of the Person class. | 1(0.72)                       | NF                        | 1(0.4)                      |
| vector difference show                            | E₉         | How vector is different from array? Illustrate with programming Example.         | 2(0.68)                       | 1(0.33)                   | NF                          |
| break statement how different                     | E₉         | Write the difference between break and continue statements in Java.             | 1(0.56)                       | 1(0.4)                    | 2(0.67)                     |
| more than one inheritance support                 | E₉         | What is inheritance? Is multiple inheritance supported by Java?                  | 1(0.52)                       | 1(0.31)                   | 2(0.33)                     |
| applet application program                        | I₈         | What is an applet? How do applets differ from an application program?            | 1(0.82)                       | NF                        | 1(0.5)                      |
| import class in program                           | E₉         | How can class be imported from a package to a program?                           | 1(0.76)                       | NF                        | 1(0.4)                      |
| can interface be used in class                    | I₈         | Is it possible to use few methods of an interface in a class? If so, how?        | 1(0.66)                       | NF                        | NF                          |
| monitor procedure for many package import auto    | I₈         | What is the procedure to own the monitor by many threads?                       | 1(0.64)                       | 1(0.44)                   | 1(0.33)                     |
| what is architecture independence                 | I₈         | Explain architecture neutral & platform independent.                            | 2(0.72)                       | 1(0.47)                   | NF                          |
| important classpath variable                      | E₉         | Write an importance of classpath variable.                                       | 2(0.51)                       | NF                        | 2(0.33)                     |
| need to import lang package                       | E₉         | Do I need to import Java lang package any time? Why?                             | 1(0.73)                       | NF                        | 1(0.4)                      |
| what is serial                                    | E₉         | Explain serialization?                                                          | 1(0.81)                       | NF                        | 1(0.66)                     |
| what is locale class                              | I₈         | What is the significance of Locale class?                                       | 1(0.68)                       | NF                        | NF                          |
| what are the alternatives to inheritance          | E₉         | Mention some alternatives to inheritance.                                       | 1(0.86)                       | NF                        | NF                          |
| is the method what finalize? of use               | E₈         | Explain the use of finalize method                                              | 1(0.86)                       | 1(0.57)                   | 1(0.6)                      |
| each of control for what the is use structure     | E₈         | What is the use of each control structure?                                      | 1(0.86)                       | 1(0.5)                    | 1(0.6)                      |
| any give 4 ++ C and Java differences. between     | E₉         | Give any 4 differences between Java and C++.                                     | 1(0.6)                        | NF                        | 1(0.25)                     |

Pal et al. (2021), PeerJ Comput. Sci., DOI 10.7717/peerj-cs.532
| User Input                                                                 | Error type | Intended question                                                                 | Soft Cosine No. of Pass (Score) | Cosine No. of Pass (Score) | Jaccard No. of Pass (Score) |
|---------------------------------------------------------------------------|------------|-----------------------------------------------------------------------------------|-------------------------------|---------------------------|-----------------------------|
| Of what handle can the are Java programs types                           | E₈         | What are the different types of program Java can handle?                          | NF                            | NF                        | 1(0.42)                     |
| What platform is independence                                            | E₆         | What is platform independency?                                                    | 1(0.81)                       | 1(0.57)                   | 1(0.5)                      |
| in of symbols comment types different are java What the                  | E₆         | What are the different types of comment symbols in Java                           | 1(0.57)                       | NF                        | 1(0.5)                      |
| How is a constant in Java defined                                        | E₆         | How is a constant defined in Java?                                                | 1(0.81)                       | 1(0.47)                   | 1(0.4)                      |
| use keyword is the what of final                                        | E₆         | What is the use of final keyword?                                                 | 1(0.86)                       | NF                        | 1(0.6)                      |
| the of is control structure use what each for                            | E₆         | What is the use of each control structure?                                       | 1(0.86)                       | 1(0.5)                    | 1(0.6)                      |
| constants constants static and compare final                             | E₆         | Compare static constants and final constants                                      | 1(0.53)                       | NF                        | 1(0.5)                      |
| need methods for is the what static                                      | E₆         | What is the need for static method?                                               | 1(0.72)                       | 1(0.37)                   | 1(0.6)                      |
| platform supports how java independency                                  | E₆         | How Java supports platform independency?                                          | 1(0.72)                       | 1(0.51)                   | 1(0.5)                      |
| to important java why is internet                                       | E₆         | Why Java is important to the internet?                                            | 1(0.54)                       | NF                        | 1(0.16)                     |
| is internet to java important why                                        | E₆         | Why Java is important to the internet?                                            | 1(0.57)                       | NF                        | 1(0.16)                     |
| Application and Applet Compare                                           | E₆         | Compare applet and application                                                    | NF                            | NF                        | NF                          |
| difference copy with clone pros and cons of static nested class          | E₆         | Differentiate cloning and copying.                                               | 2(0.59)                       | 1(0.35)                   | NF                          |
| what fields method                                                       | E₆         | Explain about Final class Fields Methods.                                         | NF                            | NF                        | NF                          |
| package access specifier priority in garbage collector                   | I₇         | What do you understand by package access specifier?                               | 1(0.63)                       | 1(0.51)                   | 1(0.33)                     |
| circle filled when right click                                           | I₇         | Garbage collector thread belongs to which priority?                               | 1(0.73)                       | 1(0.67)                   | 1(0.6)                      |
| explain assertion use                                                    | I₈         | Develop Java program that changes the color of a filled circle when you make a    | 1(0.64)                       | 1(0.56)                   | 1(0.2)                      |
| array fill                                                               | I₈         | When is an assertion? What is its use in programming?                             | NF                            | NF                        | NF                          |
| method to demon thread                                                   | I₈         | Give the syntax for array fill operation.                                         | 1(0.63)                       | NF                        | 1(0.4)                      |
| what class on read side byte stream                                      | E₆         | Which method is used to create the demon thread?                                  | 1(0.51)                       | NF                        | NF                          |
| what is the use of input stream                                          | E₆         | Name the filter stream classes on reading side of byte stream?                    | 1(0.71)                       | NF                        | NF                          |
| see if file is hidden or not                                             | E₆         | What is the functionality of sequence input stream?                               | 1(0.63)                       | NF                        | NF                          |
| when was the file last modified                                         | E₆         | How to check if a file is hidden?                                                 | 1(0.81)                       | 1(0.43)                   | 1(0.5)                      |
| use of encapsulation                                                    | E₆         | How to get file last modified time?                                               | 1(0.7)                        | 1(0.5)                    | 1(0.42)                     |
| (Continued)                                                              |            |                                                                                   |                               |                           |                             |
### Table 8 (continued)

| User Input                   | Error type | Intended question                                                                 | Soft Cosine No. of Pass (Score) | Cosine No. of Pass (Score) | Jaccard No. of Pass (Score) |
|-----------------------------|------------|----------------------------------------------------------------------------------|---------------------------------|----------------------------|-----------------------------|
| most used algorithm in collection | IS         | What are common algorithms implemented in Collections Framework?                  | 1(0.57)                         | 1(0.36)                    | NF                          |
| how is iterator designed    | EG         | What is the design pattern that iterator uses?                                   | NF                              | NF                         | NF                          |
| component size preferred    | IS         | What is the preferred size of a component?                                       | 1(0.86)                         | 1(0.86)                    | 1(0.75)                     |
| read each line of file      | EG         | How to read file content line by line in Java?                                   | 1(0.81)                         | 1(0.63)                    | NF                          |
| delete a file that is temporary | EG        | How to delete temporary file in Java?                                            | 1(0.86)                         | 1(0.61)                    | NF                          |
| calculate factorial of a number | IS        | Java program to find factorial of a number using loops                           | 1(0.51)                         | 1(0.33)                    | 1(0.25)                     |
| thread priorities           | IS         | Discuss about thread groups and thread priorities.                               | 1(0.75)                         | NF                         | NF                          |
| java programming            | IS         | Explain Java programming environment.                                            | 1(0.5)                          | NF                         | NF                          |
| what are lexical issues     | EG         | Discuss the lexical issues of Java.                                              | NF                              | NF                         | NF                          |
| data type used for arithmetic operators | EG      | Which data type can be operands of arithmetic operators?                        | NF                              | 1(0.36)                    | NF                          |
| compound assign             | IS         | What are compound assignment operators?                                          | 1(0.56)                         | 1(0.4)                     | 1(0.25)                     |
| use bitwise in boolean      | EG         | Can bitwise operators be used in Boolean operations?                             | 1(0.66)                         | 1(0.44)                    | NF                          |
| order for call awt          | EG         | What is the sequence for calling the methods by AWT for applets?                 | NF                              | NF                         | NF                          |
| cast object explain         | EG         | What do you understand by casting an object? Explain with the example            | 1(0.72)                         | NF                         | NF                          |
| find size of object         | IS         | Does Java provide any construct to find out the size of an object?              | 1(0.7)                          | NF                         | NF                          |
| required for try to follow catch | EG       | Is it necessary that each try block be followed by a catch block?               | 1(0.52)                         | 1(0.42)                    | 1(0.33)                     |
| what is precedence rule     | IS         | Explain precedence rules and associativity concept                               | 1(0.58)                         | NF                         | NF                          |
| type upgrade in method      | EG         | What type promotion has to do with method overloading?                           | NF                              | NF                         | NF                          |
| overloading                 |            |                                                                                  |                                 |                            |                             |
| explain two types of        | IS         | What is run time polymorphism and compile time polymorphism?                    | 1(0.57)                         | NF                         | NF                          |
| polymorphism                |            |                                                                                  |                                 |                            |                             |
| explain blockingqueue       | EG         | What do you understand by BlockingQueue?                                         | 1(0.64)                         | NF                         | NF                          |
| java and internet           | IS         | Why java is important to the Internet                                            | 1(0.81)                         | 1(0.66)                    | 1(0.66)                     |

**Note:**
- IS, insufficient information; EG, grammatical error; ES, sequential error; NF, not found.

The overall accuracy result of the framework corresponding to the soft cosine, Jaccard, and cosine similarity techniques is shown in Fig. 15.

The accuracy results for learners accepting the recommended question show that soft cosine similarity outperforms the cosine and Jaccard similarities. In the given experimental
data set, the soft cosine based recommendation returns the correct result in two or more passes for 12 input questions. While, for the other 73 input questions, it returns the result in one pass. Therefore, it can be concluded that though the soft cosine similarity-based recommendation returns the intended question in one pass for the maximum number of questions, recommending results in two or more passes is unavoidable. It is observed that input questions lacking sufficient information cause the recommendation system to iterate multiple passes of learner’s interaction to reach the intended question. The hefty size of the corpus might be another reason for the increased number of passes.

The results also show that for 15 input questions, the soft cosine similarity-based recommendation fails to find the correct question matching to learner’s intent. It is observed that in very few cases where the words in the input question are highly scrambled or out of sequence, it may cause the soft cosine to fail to find the correct questions. In this case, the Jaccard similarity outperforms the soft cosine. The other reason which contributes to soft cosine failing is the string length of the input question. If the string length is reduced to one or two words after stopword removal in question preprocessing, the soft cosine based recommendation is unable to find the exact intended question from the huge number of questions within a limited number (three passes) of learner’s interaction. Perhaps a greater number of interactions were needed. Besides these two structural issues on input questions, the soft cosine has some inherent limitation which causes the recommendation set to fail in retrieving the appropriate questions near to learner intention. Even though it is claimed that soft cosine works well on word similarity, actually, it does not do well for multiple synonyms while matching for similarity. The other inherent issue is that the soft cosine fails to infer the common-sense meaning from a sequence of words or phrases to find semantical similarity.

**Diversity and evenness**

Soft cosine technique with every iteration converges the search for questions on a particular topic. This causes the recommended questions to be very much focused on the
intent of the input question. To assess the effectiveness of soft cosine in each pass, the iteration result of the recommended question list, obtained by the three similarity assessment techniques, is analyzed for diversity and evenness. The diversity specifies how the questions in the recommended list are diverse in terms of topic. Where the evenness specifies how evenly the topic information (concepts) are spread (distribution) in the recommended list. The diversity and the evenness of information in the recommended list of questions in each pass are calculated by Shannon’s diversity index ($H$) and Shannon’s equitability ($E_H$), respectively, as given by Eqs. (11) and (12).

$$H = -\sum_{i=0}^{n} P_i \ln P_i$$  \hspace{1cm} (11)

Where, $n$ is the number of topic category and $P_i$ is the proportion of the number of $i^{th}$ topic relative to the total count of individual topics for all questions in the recommended list.

$$E_H = \frac{H}{\ln S}$$  \hspace{1cm} (12)

Where, $S$ is the total count of individual topics for all questions in the recommended list. The evenness value assumes between 0 and 1, where 1 denoting completely even. In an ideal situation, $H \approx 0$ specifies that topic in recommendation question list is not diverse and all recommended question focuses on one topic. Similarly, $E_H \approx 0$ specifies zero dispersion of topics in the recommended question list.

The changes in diversity and equitability indices along to each pass for a given incorrect question “java not have destroy and how garbage collect” are discussed below.

a) Each keyword in the source question denotes a concept which in turn relates to a topic. The keywords in the question are used to select and group questions from the corpus belonging to the same topic domains. The incorrect question is matched with the grouped question using the soft cosine measure. The set of suggested questions returned by the soft cosine similarity measure in the first pass is shown in Table 9. Each keyword in the recommended similar question list reflects a concept which accounts for a count of the respective topics. Based on which the $H$ and $E_H$ are calculated for the list as given in Table 10.

| Pass   | Suggested similar question                                                                 |
|--------|-------------------------------------------------------------------------------------------|
| Pass 1 | • Explain garbage collection                                                                |
|        | • How we can create java classes                                                           |
|        | • How we can create java objects                                                           |
|        | • Explain garbage collection in java programming                                           |
|        | • What is garbage collection                                                               |
|        | • How to create a file in Java                                                             |
|        | • How to read a file in Java                                                               |
b) The learner chooses the question “explain garbage collection in java programming” from the recommended list of questions which is closest to her intent as the seed question for further searching.

c) In the second pass, again, based on the keywords from the source question, the questions on the same topic are selected and grouped from the corpus. The set of suggested questions returned by the soft cosine similarity for the selected question against the selected source question is shown in Table 11.

Based on the individual topic count and the total topic count, the H and E\textsubscript{H} are calculated for the list, as given in Table 12. It is evident that the diversity index $H = 1.02$ in...
pass 1 is reduced to $H = 0.85$ in pass 2. This implies that the diversity of topic information found in the recommended list decreases along with the passes. This signifies the search information space converges, which give learner to be focused and better options to select the question from the list. Further, the evenness $E_{H1} = 0.985$ in pass 1 is reduced to $E_{H2} = 0.781$ in pass 2. This implies that the unevenness of topic distribution among the questions increases. This signifies that the distribution of the intended topic among the question increases which give a high probability of finding the right question.

The keyword-based selection and grouping of questions from corpus eliminates the otherwise irrelevant questions and thereby restricts it to a reduced topic search space. Further, soft cosine measure based similarity concretely shrinking the search to more meaningful questions close to the learner’s intent and thereby decreasing the diversity.

From the results, a sample of nine questions that passed two iterations, applying the soft cosine similarity, was considered. Table 13 shows the diversity and evenness calculated on the topic information for the recommended question list obtained after each pass corresponding to the three similarity assessment techniques for a given question. Here, diversity and evenness equating to 0 indicate that the suggested question list belongs to the same topic. Some question searches using the similarity-based technique led the learner to find the intended question in the first pass. This made the second pass for the question search a not applicable (NA) case. From the table, it is quite clear that with every pass, the diversity in the recommended list of the question, obtained by soft cosine in comparison to other, decreases. This made us conclude that with the progression of search iteration, the search space becomes narrower; in other words, the search converges.

### Table 13 Diversity index and equitability on recommended questions.

| Question Description                                      | Soft cosine | Cosine | Jaccard |
|-----------------------------------------------------------|------------|--------|---------|
| Java not have destroy and how garbage collect             | Pass 1     | 1.02   | 0.85    | 1.72    | 0.41    |
|                                                          | Pass 2     | 0.85   | 0.78    | 0.96    | 0.37    |
| how to overload                                            | Pass 1     | 0.67   | 0.97    | 0.50    | 0.72    | 0.45    |
|                                                          | Pass 2     | 0.50   | 0.72    | 0.32    | 0.46    | 0.32    |
| why thread synchronization needed                          | Pass 1     | 0.45   | 0.41    | 0.22    | 0.46    | 0.60    |
|                                                          | Pass 2     | 0.22   | 0.46    | 0.32    | 0.46    | 0.54    |
| we change throws when override                            | Pass 1     | 0.67   | 0.97    | 0.50    | 0.72    | 0.94    |
|                                                          | Pass 2     | 0.50   | 0.72    | 0.32    | 0.46    | 0.94    |
| all methods of object class                               | Pass 1     | 1.19   | 0.86    | 1.08    | 0.78    | 1.60    |
|                                                          | Pass 2     | 1.08   | 0.78    | 0.84    | 0.54    | 0.75    |
| vector difference show                                     | Pass 1     | 0.56   | 0.51    | 0.63    | 0.91    | NA      |
|                                                          | Pass 2     | 0.63   | 0.91    | 0.54    | 0.88    | NA      |
| package import auto                                        | Pass 1     | 0.41   | 0.59    | 0.00    | NA      | 0.50    |
|                                                          | Pass 2     | 0.00   | 0.00    | NA      | 0.00    | 0.90    |
| what is architecture independence                          | Pass 1     | 1.27   | 0.71    | 0.60    | 0.89    | 1.60    |
|                                                          | Pass 2     | 0.60   | 0.89    | 0.60    | 0.89    | 0.85    |
| Why main public                                            | Pass 1     | 0.00   | 0.00    | 0.50    | 0.72    | 0.94    |
|                                                          | Pass 2     | 0.50   | 0.72    | 0.32    | 0.46    | 0.54    |

---

Pal et al. (2021), *PeerJ Comput. Sci.*, DOI 10.7717/peerj-cs.532
This ensures the search result to be focused on the intended topic, which helps the learner in reaching the intended question quickly.

CONCLUSIONS AND FURTHER SCOPE

A lot of emphases are given to developing and structuring the contents so that it can be attractive and motivating to learners. Due to the high-cost factor and difficulty in managing peer-to-peer support, learner-expert based interaction is being less encouraged in online systems. Questions are one of the key forms of natural language interaction with computers which gives the learner an upper hand in interacting with computers more broadly. Composing correct questions is essential from this perspective. A rightly composed question allows a clear understanding of what the learner wants to know. An incorrectly composed question raises ambiguity and diversions, which results in incorrect information. This often misleads the learner. For determining the intent and objective and hence the semantics of the question, it is important to know whether the question is composed correctly to its semantics. Determining whether the input question is incorrectly or rightly composed would increase the accuracy of information retrieval. This put the absolute requirement for verifying whether the question framing is and by semantics is correct or not before it can be used for information retrieval.

This paper proposes an approach for assessing the validity of framing the question and its semantics. A tri-gram based language model is used for assessing the question’s correctness in terms of syntax and semantics. The model outperforms the other n-gram approaches and establishes the fact that tri-gram optimally performs well in assessing the questions. The tri-gram language model exhibits an accuracy of 92%, which is way higher than the accuracy shown by 2-gram, 4-gram, and 5-gram over the same test data assessment.

The work also proposes an interactive framework for correct question recommendation. The framework uses a soft cosine based similarity technique for recommending the correct question to the learner. The proposed framework is assessed by learner questions and compared with other similarity assessment techniques, viz. cosine and Jaccard. The soft cosine similarity technique recommends the correct question way better than the other two, achieving an accuracy of 85%. In the case of multi-pass interaction, as the number of passes increased, the information diversity is reduced, and the search is converged to the intended question quickly.

In conclusion, incorporating the presented work in an interactive OLS will not only improve the performance of the system significantly but will also enhance the learner satisfaction and learning focus, leading to a boosted quality of learning. The proposed approach can be used in precise personalized learning recommendations and mitigating the associated cold start problem.

However, this work has a couple of limitations which opens up further research scopes. Since we used a tri-gram based approach, it cannot assess the correctness of a question that has less than three words. Also, it fails to assess the informal questions that typically comprise compound and multiple sentences. Techniques like graphs (semantic network), machine learning (LSTM), etc., can be explored to solve these issues.
ADDITIONAL INFORMATION AND DECLARATIONS

Funding
The authors received no funding for this work.

Competing Interests
The authors declare that they have no competing interests.

Author Contributions
- Saurabh Pal conceived and designed the experiments, performed the experiments, analyzed the data, performed the computation work, prepared figures and/or tables, authored or reviewed drafts of the paper, and approved the final draft.
- Pijush Kanti Dutta Pramanik conceived and designed the experiments, analyzed the data, prepared figures and/or tables, authored or reviewed drafts of the paper, and approved the final draft.
- Aranyak Maity conceived and designed the experiments, performed the experiments, analyzed the data, performed the computation work, authored or reviewed drafts of the paper, and approved the final draft.
- Prasenjit Choudhury analyzed the data, authored or reviewed drafts of the paper, and approved the final draft.

Data Availability
The following information was supplied regarding data availability:

The corpus used in the experiment and the raw codes are available in the Supplemental Files.

Supplemental Information
Supplemental information for this article can be found online at http://dx.doi.org/10.7717/peerj-cs.532#supplemental-information.

REFERENCES
Adamopoulou E, Moussiades L. 2020. An overview of chatbot technology. In: Maglogiannis I, Iliadis L, Pimenidis E, eds. Artificial Intelligence Applications and Innovations. AIAI 2020. IFIP Advances in Information and Communication Technology. Vol. 584. Cham: Springer.
Agarwal N, Wani MA, Bours P. 2000. Lex-pos feature-based grammar error detection system for the English language. Electronics IX(10):1–17.
Allam AMN, Haggag MH. 2012. The question answering systems: a survey. International Journal of Research and Reviews in Information Sciences 2(3):1–13.
Amer AA, Abdalla HI. 2020. A set theory based similarity measure for text clustering and classification. Journal of Big Data 7(74):1–43 DOI 10.1186/s40537-019-0278-0.
Beckford C, Mugisa E. 2010. A tool for learner-centric course delivery in e-Learning. In: E-Learn: World Conference on E-Learning in Corporate, Government. Orlando, Florida: Healthcare, and Higher Education.
Brants T, Popat AC, Xu P, Och Fj, Dean J. 2007. Large language models in machine translation. In: Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL). Prague, Czech Republic.

Cai H, Yan C, Yin A, Zhao X. 2017. Question recommendation in medical community-based question answering. In: Neural Information Processing. Lecture Notes in Computer Science. Guangzhou, China.

Cambria E, White B. 2014. Jumping NLP curves: a review of natural language processing research. IEEE Computational Intelligence Magazine 9(2):48–57.

Chang T-H, Sung Y-T, Hong J-F, Chang J-I. 2014. KNQED: a tool for grammatical error diagnosis of Chinese sentences. In: International Conference on Computers in Education Japan.

Cheng H, Fang H, Ostendorf M. 2017. Open-domain name error detection using a multi-task RNN. In: Empirical Methods in Natural Language Processing. Lisbon.

Cheng S-M, Yu C-H, Chen H-H. 2014. Chinese word ordering errors detection and correction for non—native chinese language learner. In: International Conference on Computational Linguistics: Technical Papers. Dublin, Ireland.

Czerkawski BC. 2016. Blending formal and informal learning networks for online learning. International Review of Research in Open and Distributed Learning 17(3):138–156 DOI 10.19173/irrodl.v17i3.2344.

Dey AK, Poddar B, Pramanik PKD, Debnath NC, Aljahdali S, Choudhury P. 2020. Real-time learner classification using cognitive score. In: EPiC Series in Computing (Proceedings of 35th International Conference on Computers and Their Applications). 69:264–276.

Donnelly RC. 2009. Embedding interaction within a blend of learner centric pedagogy and technology. World Journal on Educational Technology 1:6–29.

Fang L, Tuan LA, Hui SC, Wu L. 2017. Personalized question recommendation for English grammar learning. Expert System 35(2):1–15.

Felice M, Briscoe T. 2015. Towards a standard evaluation method for grammatical error detection and correction. In: Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. Denver, Colorado.

Ferraro G, Nazar R, Ramos MA, Wanner L. 2014. Towards advanced collocation error correction in Spanish learner corpora. Language Resources and Evaluation 48:45–64.

Ganesh BR, Gupta D, Sasikala T. 2018. Grammar error detection tool for medical transcription using stop words parts-of-speech tags ngram based model. In: International Conference on Computational Intelligence and Informatics, Advances in Intelligent Systems and Computing.

Ge T, Wei F, Zhou M. 2018. Reaching human-level performance in automatic grammatical error correction: an empirical study. arXiv preprint. Available at https://arxiv.org/abs/1807.01270.

Giffels KM, Karle CJ, O’Keeffe WG, Patel KT, Taieb DD, Yee S. 2014. Displaying quality of question being asked a question answering system. USA Patent US. Available at https://patents.google.com/patent/US9633309B2/en.

Hasan MZ, Hossain S, Rizvee MA, Rana MS. 2019. Content based document classification using soft cosine measure. International Journal of Advanced Computer Science and Applications 10(4):522–528.

Herder E, Sosnovsky S, Dimitrova V. 2017. Adaptive intelligent learning environments. Cham: Springer.

Huang C-J, Liu M-C, Chu S-S, Cheng C-L. 2004. Application of machine learning techniques to Web-based intelligent learning diagnosis system. In: International Conference on Hybrid Intelligent Systems. Kitakyushu, Japan.
Islam A, Milios E, Kešelj V. 2012. Text similarity using google tri-grams. In: Kosseim L, Inkpen D, eds. Advances in Artificial Intelligence. Heidelberg, Berlin: Springer, 312–317.

Islam S, Sarkar MF, Hussain T, Hasan MM, Farid DM, Shatabda S. 2018. Bangla sentence correction using deep neural network based sequence to sequence learning. In: International Conference on Computer and Information Technology. Dhaka.

Joshi S, Jadon RS, Jain RC. 2012. Sequential pattern mining using formal language tools. International Journal of Computer Science Issues IX(5):2016–2325.

Jurafsky D, Martin JH. 2013. Speech and language processing. Second Edition. Chennai: Pearson Education India.

Kao CC, Chang JW, Wang TI, Huang YM, Chiu PS. 2019. Design and development of the sentence—based collocation recommendation with error detection for academic writing. Journal of Internet Technology 20(1):229–236.

Kowsari K, Meimandi KJ, Heidarysafa M, Mendu S, Barnes L, Brown D. 2019. Text classification algorithms: a survey. Information: An International Interdisciplinary Journal 10(4):1–68.

Leacock C, Chodorow M, Gamon M, Tetreault J. 2010. Automated grammatical error detection for language learners. USA: Morgan & Claypool.

Lee L-H, Chang L-P, Lee K-C, Tseng Y-H, Chen H-H. 2013a. Linguistic rules based Chinese error detection for second language learning. In: International Conference on Computers in Education. Denpasar Bali, Indonesia.

Lee LH, Chang L, Lee KC, Tseng YH, Chen HH. 2013b. Linguistic rules based chinese error detection for second language learning. In: Proceedings of the 21st International Conference on Computers in Education (ICCE’13). Bali, Indonesia.

Lee L-H, Yu L-C, Lee K-C, Tseng Y-H, Chang L-P, Chen H-H. 2014. A sentence judgment system for grammatical error detection. In: International Conference on Computational Linguistics: System Demonstrations. Dublin.

Lei T, Joshi H, Barzilay R, Jaakkola T, Tymoshenko K, Moschitti A, Marquez L. 2016. Semi-supervised question retrieval with gated convolutions. In: Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. San Diego, California.

Li S, Manandhar S. 2011. Improving question recommendation by exploiting information need. In: Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies. Portland, Oregon, USA.

Maghraby AO, Khan NN, Ahmed HA, Brohi GN, Assouli HF, Melibari JS. 2020. Tibyan automated Arabic correction using machine-learning in detecting syntactical mistakes. International Journal of Cognitive and Language Sciences 14(6):402–406.

Malik S, Mandal M, Bandyopadhyay SK. 2017. Grammatical error checking and polarity determination of a simple sentence. Global Research and Development Journal for Engineering II(7):142–146.

Mooney CH, Roddick JF. 2013. Sequential pattern mining—approaches and algorithms. ACM Computing Surveys 45(2):1–39 DOI 10.1145/2431211.2431218.

Mukhopadhyay M, Pal S, Nayyar A, Pramanik PKD, Dasgupta N, Choudhury P. 2020. Facial emotion detection to assess Learner’s State of mind in an online learning system. In: 5th International Conference on Intelligent Information Technology (ICIIT 2020). Hanoi, Vietnam.

Nguyen VA. 2018. An online question and answering system for support teacher-student interaction in the blended learning course. International Journal of Learning and Teaching 4(3):178–184.
Othman MTB, Al-Hagery MA, Hashemi YMEl. 2020. Arabic text processing model: verbs roots and conjugation automation. *IEEE Access* **VIII**:103913–103923.

Pal S, Pramanik PKD, Choudhury P. 2019. A step towards smart learning: designing an interactive video-based M-learning system for educational institutes. *International Journal of Web-Based Learning and Teaching Technologies* **14**(4):26–48 DOI 10.4018/IJWLTT.2019100102.

Pal S, Pramanik PKD, Majumdar T, Choudhury P. 2019. A semi-automatic metadata extraction model and method for video-based e-learning contents. *Education and Information Technologies* **24**(6):3243–3268 DOI 10.1007/s10639-019-09926-y.

Qu M, Qiu G, He X, Zhang C, Wu H, Bu J, Chen C. 2009. Probabilistic question recommendation for question answering communities. In: *International Conference on World Wide Web*. Madrid, Spain.

Radović-Marković M. 2010. Advantages and disadvantages of e-learning in comparison to traditional forms of learning. *Annals of the University of Petroșani, Economics* **10**(2):289–298.

Rei M, Yannakoudakis H. 2016. Compositional sequence labeling models for error detection in learner writing. In: *Annual Meeting of the Association for Computational Linguistics*. Berlin.

Rei M, Yannakoudakis H. 2017. Auxiliary objectives for neural error detection models. In: *Workshop on Innovative Use of NLP for Building Educational Applications*. Copenhagen.

Schmaltz A, Kim Y, Rush AM, Shieber SM. 2016. Sentence-level grammatical error identification as sequence-to-sequence correction. In: *Workshop on Innovative Use of NLP for Building Educational Applications*. San Diego.

Shiue Y-T, Huang H-H, Chen H-H. 2017. Detection of ChineseWord usage errors for non-native Chinese learners with bidirectional LSTM. In: *Annual Meeting of the Association for Computational Linguistics (Short Papers)*. Vancouver, Canada.

Sidorov G, Gelbukh A, Gómez-Adorno H, Pinto D. 2014. Soft similarity and soft cosine measure: similarity of features in vector space model. *Computacion y Sistemas* **18**(3):1–14.

Slimani T, Lazzez A. 2013. Sequential mining: patterns and algorithms analysis. *International Journal of Computer & Electronics Research* **2**:639–664.

Sneiders E. 2009. Automated FAQ answering with question-specific knowledge representation for web self-service. In: *Conference on Human System Interactions*. Catania, Italy.

Sohangir S, Wang D. 2017. Improved sqrt-cosine similarity measurement. *Journal of Big Data* **4**(25):1–13 DOI 10.1186/s40537-017-0083-6.

Sonawane A, Vishwakarma SK, Srivastava B, Singh AK. 2020. Generating inflectional errors for grammatical error correction in Hindi. In: *Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the International Joint Conference on Natural Language Processing: Student Research Workshop*. Suzhou, China.

Soni M, Thakur JS. 2018. A systematic review of automated grammar checking in English language. arXiv. Available at [http://arxiv.org/abs/1804.00540](http://arxiv.org/abs/1804.00540).

Srba I, Savic M, Bielikova M, Ivanovic M, Pautasso C. 2019. Employing community question answering for online discussions in university courses: students’ perspective. *Computers & Education* **135**:75–90.

Sun G, Cong G, Liu X, Lin C-Y, Zhou M. 2007a. Mining sequential patterns and tree patterns to detect erroneous sentences. In: *Association for the Advancement of Artificial Intelligence*.

Sun G, Liu X, Cong G, Zhou M, Xiong Z, Lee J, Lin C-Y. 2007b. Detecting erroneous sentences using automatically mined sequential patterns. In: *Annual Meeting of the Association of Computational Linguistics*. Prague, Czech Republic.
Sun P-C, Tsai RJ, Finger G, Chen Y-Y, Yeh D. 2008. What drives a successful e-learning? An empirical investigation of the critical factors influencing learner satisfaction. *Computers & Education* 50(4):1183–1202 DOI 10.1016/j.compedu.2006.11.007.

Szpektor I, Maarek Y, Pelleg D. 2013. When relevance is not enough: promoting diversity and freshness in personalized question recommendation. In: *International Conference on World Wide Web*. Rio de Janeiro, Brazil.

Tezcan A, Hoste V, Macken L. 2016. Detecting grammatical errors in machine translation output using dependency parsing and treebank querying. In: *Conference of the European Association for Machine Translation*. Riga, Latvia.

Wallace RM. 2003. Online learning in higher education: a review of research on interactions among teachers and students. *Education, Communication & Information* 3(2):241–280 DOI 10.1080/14636310303143.

Wang J, Su L, Chen J, Jiang D. 2017. Question recommendation based on user model in CQA. In: Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering. Guangzhou, China.

Wang Y, Wang Y, Liu J, Liu Z. 2020. A comprehensive survey of grammar error correction. *arXiv*. 1–35. Available at http://arxiv.org/abs/2005.06600.

Wang Y, Wang L, Wong DF, Chao IS, Zeng X, Lu Y. 2014. Factored statistical machine translation for grammatical error correction. In: *Conference on Computational Natural Language Learning: Shared Task*. Baltimore.

Wang S-L, Wu C-Y. 2011. Application of context-aware and personalized recommendation to implement an adaptive ubiquitous learning system. *Expert Systems with Applications* 38(9):10831–10838 DOI 10.1016/j.eswa.2011.02.083.

Woods RH, Baker JD. 2004. Interaction and immediacy in online learning. *International Review of Research in Open and Distance Learning* 5(2):1–13 DOI 10.19173/irrodl.v5i2.186.

Xiang Y, Wang X, Han W, Hong Q. 2015. Chinese grammatical error diagnosis using ensemble learning. In: *Workshop on Natural Language Processing Techniques for Educational Applications*. Beijing.

Yang D, Adamson D, Rosé CP. 2014. Question recommendation with constraints for massive open online courses. In: *ACM Conference on Recommender systems*. New York: ACM.

Yannakoudakis H, Rei M, Andersen Ø.E, Yuan Z. 2017. Neural sequence-labelling models for grammatical error correction. In: *Conference on Empirical Methods in Natural Language Processing*. Copenhagen.

Yeh J-F, Hsu T-W, Yeh C-K. 2016. Grammatical error detection based on machine learning for mandarin as second language learning. In: *Workshop on Natural Language Processing Techniques for Educational Applications*. Osaka, Japan.

Yu C-H, Chen H-H. 2012. Detecting word ordering errors in Chinese sentences for learning Chinese as a Foreign language. In: *COLING 2012: Technical Papers*. Mumbai, India.

Zan H, Han Y, Huang H, Yan Y, Wang Y, Han Y. 2020. Chinese grammatical errors diagnosis system based on BERT at NLPTEA-2020 CGED shared task. In: *Workshop on Natural Language Processing Techniques for Educational Applications*. Suzhou, China.

Zhao Q, Bhowmick SS. 2003. *Sequential pattern mining: a survey*. Singapore: CAIS, Nanyang Technological University.
Zhao W, Wang L, Shen K, Jia R, Liu J. 2019. Improving grammatical error correction via pre-training a copy-augmented architecture with unlabeled data. In: Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. Minneapolis.

Zheng B, Che W, Guo J, Liu T. 2016. Chinese grammatical error diagnosis with long short-term memory networks. In: Workshop on Natural Language Processing Techniques for Educational Applications. Osaka.