Sensors and Materials, Vol. 35, No. 12 (2023) 4325–4336
MYU Tokyo

S & M 3483

MCQA: A Responsive Question-answering System for Online Education

Yi Wang,[1*†] Jinsheng Deng,[1†] Xi Yang,[1] Jianyu Yi,[2] and Zhaohui Ye[1]

[1]National University of Defense Technology, 109 Deya Rd, Changsha, 410073, China
[2]Fudan University, 220 Handan Rd, Shanghai, 200433, China

(Received April 28, 2023; accepted December 8, 2023)

Keywords: QA system, MOOCs, question classification, similarity retrieval, similarity computation, chitchat generation

Massive Open Online Courses (MOOCs) are now considered as representatives of online education. In addition to watching course videos and taking tests, online question & answering (Q&A) also plays an important role during MOOC learning. In this paper, we introduce a question-answering (QA) system called MCQA for MOOCs. The system comprises several modules, including question classification, similarity retrieval, similarity computation, and chitchat generation. On a real MOOC platform, MCQA demonstrates exceptional performance, with experimental results showing a precision rate exceeding 90% and an average duration of a Q&A session of less than 100 ms. Compared with other Chinese-based QA systems, MCQA provides superior open-ended QA capabilities, excelling in performance and covering numerous learning scenarios.

1. Introduction

With the continuous development of online education, traditional education is transitioning towards personalized education.(1) The change from traditional education to personalized education shows that education has more emphasis on learner's subjective initiative. As a typical model in online education, MOOCs make education available to large demographics.(2) One of these challenges is that, since courses are taken online, it has become more difficult to interact with instructors owing to the lack of face-to-face interaction.(3) It has become easy for learners to experience isolated learning, which may result in low completion rates for courses or even dropping out.(4) Particularly when they have problems in online learning but cannot get answers immediately, they may struggle to attain the desired learning outcomes.

However, instructors cannot always be online. The emergence of a QA system has proven to be a great solution to this challenge. In Fig. 1, a QA system typically enables learners to type a question, and it will automatically retrieve the most relevant answer from the knowledge base. The fundamental process of a QA system begins with learners posing their questions, which are then transmitted to the QA system via a communication link. The QA system exchanges data

*Corresponding author: e-mail: wangyi17d@nudt.edu.cn
†These authors contributed equally to this work.
https://doi.org/10.18494/SAM4483

ISSN 0914-4935
https://myukk.org/

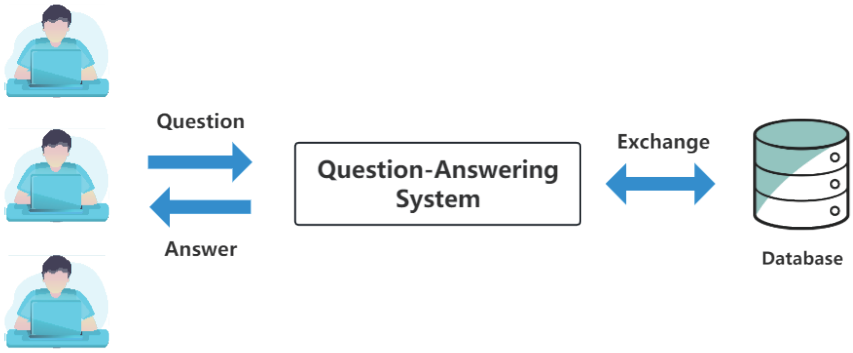

Fig. 1. (Color online) Fundamental process of QA system.

with the target database and performs appropriate data processing. Finally, the ideal answer is obtained and fed back to the learners. In this way, learners can not only obtain answers quickly, but it can also alleviate the workload of instructors and improve the students' learning efficiency. Research on the implementation of the QA system has been ongoing for years, and various technologies and solutions make the QA system widely used in many domain areas.

Currently, there are numerous cutting-edge studies focusing on QA systems. In general applications, these approaches have shown promising performance. However, in the realm of online education, there are limited studies and cases available. These factors such as the complexity of real-world application scenarios and the need for a system structure that is easy to maintain and cost-effective contribute to the challenges associated with developing a QA system for online education platforms.

With the development of MOOC platforms, we have observed the emergence of new characteristics in the application scenarios of QA systems, as follows:

1. Not all the questions are directly related to MOOCs. Learners' questions can be categorized into chitchat and specific questions about MOOCs.
2. The majority of questions pertain to the knowledge points covered in the course, and the answers generally consist of the original content of the course.
3. The questions asked by learners may vary in wording, but they often convey similar meanings, allowing for the reuse of answers.
4. The duration of the Q&A session affects the learners' learning experience.
5. For chitchat questions, providing natural and diverse responses can also affect the learners' learning experience.

Previous studies have focused on the correct and answer rates of question answering. However, these clearly cannot cover all of the above scenarios. Hence, we present a QA system called MCQA and our contributions are as follows:

1. We designed a multimodule QA system that efficiently responds to both chitchat and professional questions related to MOOCs. The system boasts a response time of less than 100 ms from the time a question is asked to the delivery of an answer.
2. We proposed an algorithm for collecting data from MOOC platforms, aimed at facilitating training and evaluation processes.

3. Compared with state-of-the-art methods, we have achieved optimal performance during the design of each module.

The main sections of this paper are as follows: In Sect. 2, we review the related and previous studies that have been conducted in the QA domain. In Sect. 3, we introduce the architecture and modules of MCQA. In Sect. 4, we present the workflow and evaluation of MCQA. In Sect. 5, we provide the conclusion and discuss future work.

2. Related Work

As technology continues to evolve, the QA system has made considerable progress in recent years. It combines many research domains such as natural language processing (NLP) and information retrieval (IR). In different domain areas, the framework of the QA system may be different. Through our investigation, the types of QA system are open- and close-domain systems.[5] Open-domain systems can answer the question whether it is professional or not, and they usually have a large knowledge database to retrieve the right answer. On the other hand, close-domain systems can answer only domain-specific questions.

In terms of function, the QA system can be divided into many types of approach. For instance, the Frequently Asked Question & Answer (FAQs)[6] is a simple approach that takes inputs to compare with the questions in the database directly. The information retrieval (IR) approach[7] is an approach that retrieves correct answers from a knowledge base.

In general, the QA system needs to solve problems such as question analysis. These approaches, called question classification, can fall into three categories: NLP approach, machine learning approach, and hybrid approach.[8]

A wide range of QA systems have emerged over 20 years. Seena *et al.*[9] presented a close-domain QA system in Malayalam. The system can output correct answers for the question based on domain-specific documents. It can also analyze the keywords in the question and find the closest sentence in the context as the returned answer.

Ryu *et al.*[10] used a Wikipedia-based knowledge model to create an open-domain QA system. With the huge data source of Wikipedia, the system has multiple answer matching modules. The question analysis works to define the type of question and its semantics. The corresponding modules process questions to obtain correct answers.

Wu *et al.*[11] conducted a survey of open-domain Question Answering over Knowledge Base (KBQA). They showed that KBQA is generally based on the knowledge base that considers fact retrieval matching and reasoning, which generally consists of modules such as Question Analysis and Disambiguation. The two categories of methods for KBQA are Semantic Parsing and IR.

Cabrio *et al.*[12] proposed an open-domain QA system called QAKiS based on relational patterns. The system can address the problem of question interpretation as a relation-based match. It consists of two modules, namely, Query Generator to generate the queries from the retrieved patterns and Pattern Matcher to retrieve the patterns matching typed questions with the highest similarity.

In addition to the research of the QA system, other research studies in the fields of machine learning and natural language processing (NLP) are also worthy of reference. The system developed by Rodrigues *et al.*[13] for real-time Twitter spam detection and sentiment analysis showed that a single data source can obtain valuable analysis from multiple perspectives, and the architecture of Live Tweet analysis was worth learning. Dimensionality reduction techniques allow us to understand that not all the attributes of raw data are useful, and effective data collection can extract worthy information from raw data.

In Chinese-based MOOCs, Xiao-Shih, the first intelligent question answering bot on Chinese-based MOOCs, can be deeply involved in online learning and can have a high correct answer rate.[14] The architecture of Xiao-Shih has three modules: data collection, question retrieval, and answer selection. They make Xiao-Shih predict the best answer to learners. However, the situation of chitchat should also be discussed, but it is not mentioned in this study.

3. System Design and Implementation

3.1 System design and architecture

Figure 2 shows the architecture of MCQA. In the question classification module, we proposed trained models to estimate whether a question is professional or not. In the similarity retrieval module, we implemented the problem of finding semantically similar questions based on the vector indexing algorithm. We also contrasted some state-of-the-art methods for encoding sentence vectors. Furthermore, MCQA would select the best answer in the similarity computation module and make some chitchat in the chitchat generation module when learners want to chat.

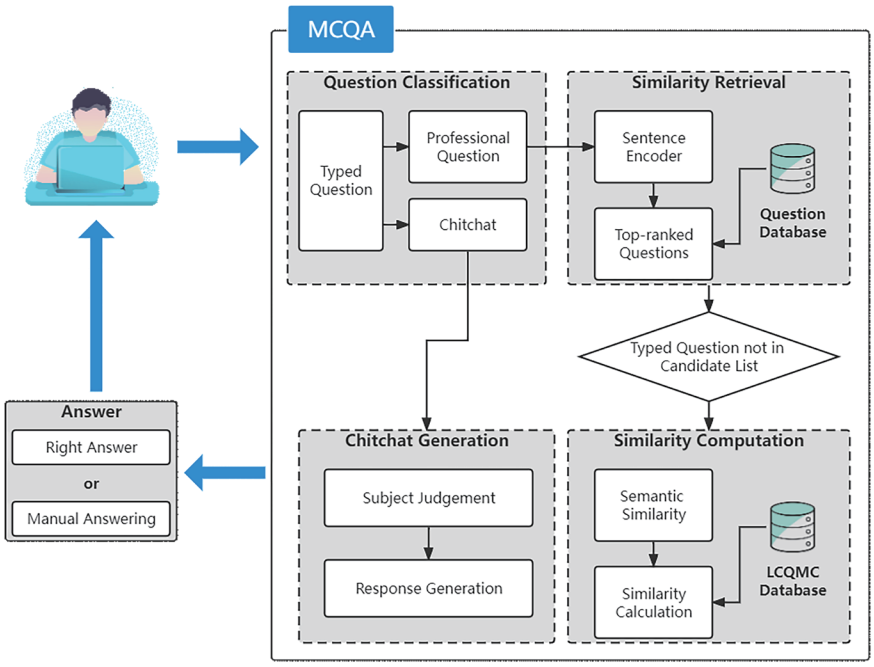

Fig. 2. (Color online) Architecture of MCQA.

3.2 Data collection

The method of data collection was similar to that of Jenders *et al.*[15] The data collected from the MOOC platform for MCQA are mainly derived from questions designed by instructors and posed by learners in a discussion forum. During data transmission and storage, the presence of special symbols may potentially result in escape problems, leading to increased data uncertainty. The first crucial step in addressing this issue is data cleansing, which involves retrieving the raw data from the database and removing any extraneous labels, such as ' '. This process essentially guarantees the absence of special symbols within the data, resulting in improved readability and standardization.

A question in our database is composed of various properties, including title, content, and answer. However, it is important to note that the relationship between the title and the content is not the only correspondence. Upon data retrieval, it has been observed that a single title may correspond to multiple similar content cases. This indicates the presence of irrelevant or low-quality data in the answers, which cannot be accurately matched to the questions. It is evident that the current raw data do not meet the required standards, thus necessitating data cleansing and data enhancement processes. Algorithm 1 (Fig. 3) further illustrates how to improve data quality.

Algorithm 1 Algorithm to improve data quality.

Input: *raw data D*
Output: *training data T*

function CLEANHTML(p : *raw data*)
  $Data[1...m]\{title, content, answer\} \Leftarrow$ CLEARHTMLTAGS(p)
  return CREATELIST($Data$)
end function
function CREATELIST($Data$)
  $List[1...2*m] \Leftarrow$ create a list with the title and content of each row of Data corresponding to answer respectively
  $UniqueList \Leftarrow$ remove duplicates and trash data such as '?' from UniqueList
  return UniqueList
end function

$List \Leftarrow$ CLEANHTML(D)
for $i = 0 \rightarrow Lengthof List - 1$ **do**
  for $j = i + 1 \rightarrow Lengthof List - 1$ **do**
    if GETSIMILARITY($List[i], List[j]$) > 0.7 **then**
      remove List[j] item from the List
    end if
  end for
end for
$output \Leftarrow List$

Fig. 3. Algorithm of how to improve data quality.

In this algorithm, two functions, which correspond to data cleansing and data enhancement have been determined. In the main process, the raw data go through the function CLEANHTML to complete data cleansing; then, the function CREATELIST works for data enhancement, which involves the addition of data and the deletion of meaningless contents. The QA pairs are created by loop structures for data similarity comparison. Finally, we have obtained more than 480 thousand pairs for training.

3.3 Question classification

In the question classification module, we choose FastText, TextCNN, and Bert as the methods of testing and optimization. FastText[16] is a toolkit developed by Facebook for the effective learning of word representations. TextCNN[17] is the application of the convolutional neural network (CNN) to the text classification task. BERT[18] is a pretrained model proposed by Google AI, which is widely used in NLP. There are more than 940 thousand pairs as the classification database, which consists of the professional QA pairs we prepared in Sect. 3.1 and the chitchat data collected from the Internet.

The metrics chosen for evaluating performance are precision, recall, and $F1$ score. $F1$ score is a metric calculated as the harmonic mean of precision and recall. Precision is the fraction of the predicted type, which is present in the ground truth, whereas Recall is the fraction of the ground truth, which is present in the prediction.

$$F1 = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{1}$$

Table 1 shows the experimental results for question classification. Through training, all these methods can obtain a precision of more than 90%. The results show that these methods can fit the application considerably. For reasoning performance, there is a certain difference between the three methods. Bert takes the longest time at 19.53 ms, TextCNN takes 1.31 ms, whereas FastText only takes 0.42 ms.

3.4 Similarity retrieval

In the similarity retrieval module, similar questions are automatically matched and found in the knowledge base automatically by the model we proposed, which is based on the Facebook Artificial Intelligence Similarity Search (FAISS)[19] framework with the Hierarchical Navigable Small World (HNSW)[20] graph as an index structure.

Table 1
Experimental results for question classification.

Method	Precision (%)	Recall (%)	$F1$ score (%)
FastText	99.15	99.15	99.15
TextCNN	98.47	99.12	99.12
Bert	99.30	99.60	99.50

HNSW is a type of K-Approximate Nearest Neighbors Search (K-ANNS) algorithm, which is similar to the previously studied Navigable Small World (NSW). NSW solves the problem of the divergent search of neighbor graphs by designing navigable graphs, but its search complexity is still very high, reaching the level of multiple logarithms, and the overall performance is easily affected by the size of the graphs. However, HNSW can solve these problems well, and the proposed HNSW is based on SkipLists. Figure 4 shows the illustration of HNSW.[21]

In the realization of the HNSW vector index, all the questions that we prepared need to be vectorized. The training is based on the Wikipedia dataset, which has more than 12 million words including professional words to train word vectors. This ensures that out-of-vocabulary words are avoided as much as possible when creating sentence vectors. Such optimization makes the generated sentence vectors retain the information of keywords in the original sentence as much as possible so that the retrieval task can work well.

Other state-of-the-art methods for encoding sentence vectors, such as Word2Vec[22] and Embeddings from Language Models (ELMo),[23] are also being compared. Word2Vec is a classical word vector model, which mainly uses skip-gram and Continuous Bag of Words (CBOW) to realize context prediction. ELMo is a new deep contextualized lexical expression, which uses the biLM language task to predict current words. On the basis of encoding sentence vectors, the questions are recalled by calculating cosine similarity.

The performance results are presented in Table 2 along with the corresponding values of retrieval time. Through the comparison, FAISS substantially outperforms the other models.

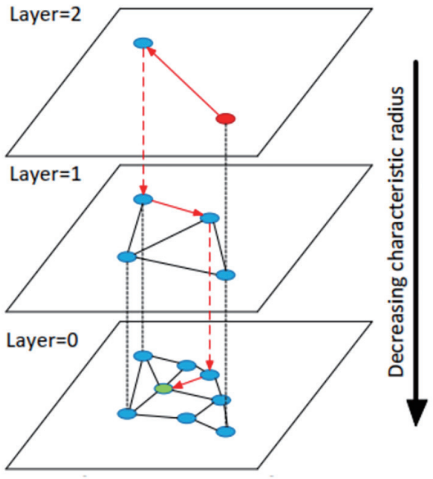

Fig. 4. (Color online) Illustration of HNSW.

Table 2
Performance of similarity retrieval.

Method	Minimum time (ms)	Maximum time (ms)	Mean time (ms)
FAISS	22.948	45.157	36.273
Word2Vec	183.251	680.995	457.010
ELMo	529.626	896.795	471.943

3.5 Similarity computation

The semantic similarity calculation method aims to calculate the semantic similarity of retrieval questions and inputs, and returns the most similar question and answer.

Sentence-BERT (SBERT) is a modification of the BERT using siamese and triplet network structures. We use SBERT to compute the cosine similarity between two sentences in the retrieval results, as shown in Fig. 5.[24]

Compared with BERT and RoBERTa, SBERT delivers superior performance in similarity computation. Particularly, when handling diverse sentence pairs, SBERT efficiently establishes the embedding vector and calculates distances in a very short period of time. In experiments, the computation time ranges between 42 and 71 ms.

3.6 Chitchat generation

The chitchat generation module's objective is to provide answers to questions classified as chitchat, as submitted by learners. The generated responses should be diverse and engaging, ensuring that learners have lively and interesting conversations and interactions with the intelligent question-answering system.

GPT-2 is a language model using the Transformer decoder structure, which is proposed by OpenAI.[25] As the successor of GPT, GPT-2 uses a larger scale of tokens and increases sequence length. The dataset with more than 1 billion words has been used for the training of GPT-2. Therefore, it is widely used in various language generation tasks and has good context capturing ability. Figure 6 shows the illustration of GPT-2.

We have over half a million Chinese chattering corpus to finish training tasks. On a modern V100 GPU, the training takes five days to complete about 40 epochs. Finally, the loss was around 2.0 in the end, and the effect of generating chitchat answers in predicting inference also reached the expectation.

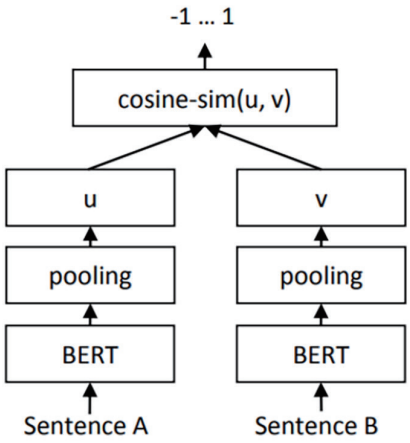

Fig. 5. Illustration of SBERT.

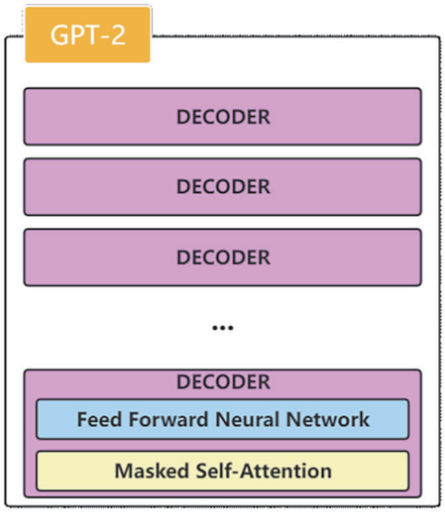

Fig. 6. (Color online) Illustration of GPT-2.

4. Experimental Results and Evaluation

The workflow of MCQA is shown in Fig. 7, and the detailed implementations are as follows.

First, questions are classified as either professional or chitchat. We utilized the FastText model with a threshold of 0.6. If the data exceeded this threshold, it would be assigned as a professional question; otherwise, it would be classified as chitchat. In the case of a chitchat classification, the model based on GPT-2 engages in conversation with the learners. For professional questions, the next step is to find semantically similar questions from the archived questions in the knowledge base. We used a model based on FAISS to retrieve the top-k questions, where k is defined as 10. Once these questions are obtained, we compute their similarities using cosine similarity in the SBERT-based model. Furthermore, we set the threshold for cosine similarity at 0.8 to determine if there is a suitable answer available.

An evaluation was conducted to predict accepted answers, involving the participation of several learners. Over a period of seven days, the system provided answers to their questions during the learning process on the platform. Afterwards, the learners were asked to indicate whether the answers successfully resolved their respective problems.

Table 3 shows the results for each question type and total. As can be seen, the system would give an answer accepted by learners with up to 90% accuracy. However, the results can be further analyzed because of people's subjective judgments and complicated questions. In such cases, there were still some correct answers not found in the database, which means that database expansion can help improve prediction ability.

After a number of experiments, the average duration of a Q&A session was 96 ms, and performance was measured on a server with Intel Xeon Platinum 8260 CPU @ 2.40 GHz, Nvidia Tesla V100 GPU, and CUDA 9.2.

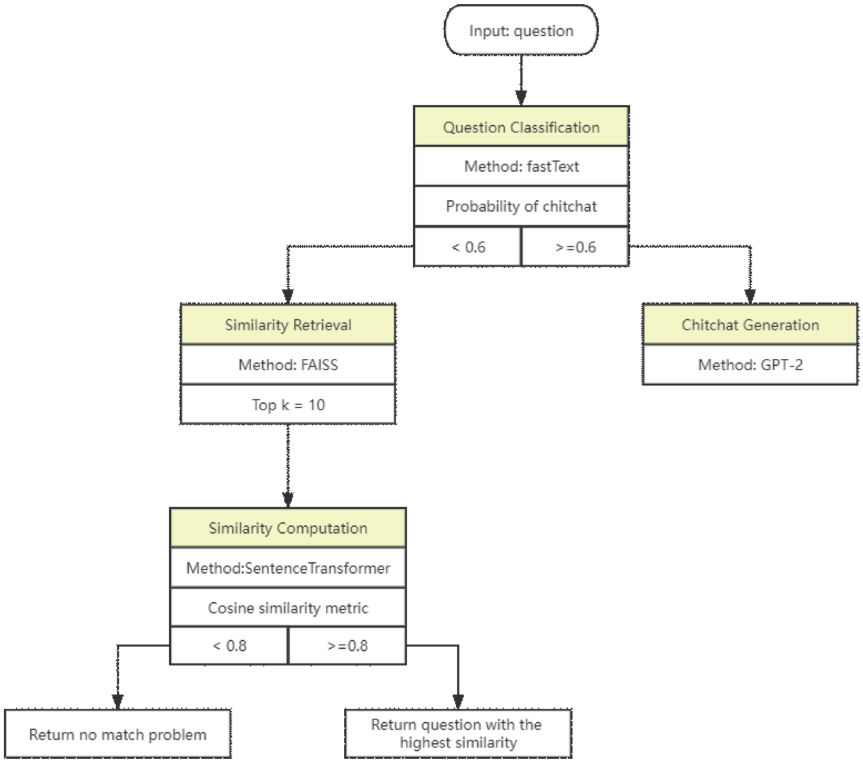

Fig. 7. (Color online) Workflow of MCQA.

Table 3
Evaluation on accepted answer prediction.

Question type	Correct	Incorrect
Professional question	219	21
Chitchat	93	7
Total	312	28

Correct indicates the answers accepted by learners; incorrect indicates no answer or dissatisfaction.

5. Conclusion and Future Work

In this paper, we proposed an open-domain QA system named MCQA on Chinese-based MOOCs to automatically respond to learners' questions. The system consists of question classification, similarity retrieval, similarity computation, and chitchat generation. Experimental results showed that it can achieve 91.76% precision in real cases. The average time for each response is less than 100 ms, which means high performance in QA systems. Compared with other Chinese-based QA systems, it can provide open-ended QA with excellent performance. Our research has contributed to an improved online learning experience for learners on the MOOC platform and has addressed the issue of instructors being unable to provide assistance to learners at all times.

In the experiment, the system exhibited certain limitations, such as its oversensitivity to domain-specific terminology, resulting in classification errors. Our future objective is to enhance the system's semantic analysis capability and establish a self-training model mechanism that enables the model to autonomously learn from new data without human intervention. These improvements will help MOOC platforms achieve stronger learning support capabilities at a lower cost.

Acknowledgments

The authors did not receive specific funding.

Conflicts of Interest

The authors declare that they have no competing interest.

Data Availability Statement

The data used to support the findings of this study are included within the article.

References

1 S. Maghsudi, A. Lan, J. Xu, and M. van der Schaar: IEEE Signal Process. Mag. **38** (2021) 37. https://doi.org/10.1109/MSP.2021.3055032
2 C. R. Glass, M. S. Shiokawa-Baklan, and A. J. Saltarelli: New Directions Inst. Res. **167** (2015) 41. https://doi.org/10.1002/ir.20153
3 D. Bouhnik and T. Marcus: J. Am. Soc. Inf. Sci. Technol. **57** (2006) 299. https://doi.org/10.1002/asi.20277
4 R. Kim, L. Olfman, T. Ryan, and E. Eryilmaz: Comput. Educ. **70** (2014) 150. https://doi-org-s.libyc.nudt.edu.cn:443/10.1016/j.compedu.2013.08.006
5 S. K. Dwivedi and V. Singh: Procedia Technol. **10** (2013) 417. https://doi.org/10.1016/j.protcy.2013.12.378
6 R. D. Burke, K. J. Hammond, V. Kulyukin, S. L. Lytinen, N. Tomuro, and S. Schoenberg: AI mag. **18** (1997) 57. https://doi.org/10.1609/aimag.v18i2.1294
7 Y. Chen and F. Zulkernine: Proc. 2021 IEEE Int. Conf. Big Data (IEEE, 2021) 3503–3510.
8 S. Yilmaz and S. Toklu: Neural Comput. Appl. **32** (2020) 2909. https://doi.org/10.1007/s00521-020-04725-w
9 I. T. Seena, G. M. Sini, and R. Binu: Procedia Technol. **24** (2016) 1388. https://doi.org/10.1016/j.protcy.2016.05.155
10 P. M. Ryu, M. G. Jang, and H. K. Kim: Inf. Process. Manage. 50.**5** (2014) 683. https://doi.org/10.1016/j.ipm.2014.04.007
11 P. Wu, X. Zhang, and Z. Feng: Proc. 2019 Knowledge Graph and Semantic Computing: Knowledge Computing and Language Understanding: 4th China Conference. (Springer Singapore, 2019) 86–97.
12 E. Cabrio, J. Cojan, A. P. Aprosio, B. Magnini, A. Lavelli, and F. Gandon: 2012 Int. Semantic Web Conference.
13 A. P. Rodrigues, R. Fernandes, A. Shetty, K. Lakshmanna, and R. M. Shafi: 2022 Computational Intelligence and Neuroscience.
14 H. H. Hsu, N. F. Huang: IEEE Transactions on Learning Technologies **15** (2022) 223–237. https://doi.org/10.1109/tlt.2022.3162572
15 M. Jenders, R. Krestel, and F. Naumann: 2016 Proc. 25th Int. Conf. Companion on World Wide Web. (2016) 679–684.
16 B. Kuyumcu, C. Aksakalli, and S. Delil: Proc. 2019 3rd Int. Conf. Natural Language Processing and Information Retrieval (2019) 1–4.
17 B. Guo, C. Zhang, J. Liu, and X. Ma: Neurocomput. **363** (2019) 366. https://doi.org/10.1016/j.neucom.2019.07.052
18 A. Adhikari, A. Ram, R. Tang, and J. Lin: arXiv preprint (2019) 1904.08398.

19 C. Mu, B. Yang, and Z. Yan: arXiv preprint (2019) 1906.10095.
20 K. R. Bashyam, and S. Vadhiyar: Proc. 2020 IEEE Int. Conf. Cluster Computing (IEEE, 2020) 294–302.
21 Y. A. Malkov and D. A. Yashunin: IEEE Trans. Pattern Anal. Mach. Intell. **42** (2018) 824.
22 X. Rong: arXiv preprint (2014) 1411.2738. https://arxiv.org/abs/1411.2738
23 K. Ethayarajh: arXiv preprint (2019) 1909.00512. https://arxiv.org/abs/1909.00512
24 N. Reimers and I. Gurevych: arXiv preprint (2019) 1908.10084. https://arxiv.org/abs/1908.10084
25 Y. Qu, P. Liu, W. Song, L. Liu, and M. Cheng: Proc. 2020 IEEE 10th Int. Conf. Electronics Information and Emergency Communication. (IEEE 2020) 323–326.