Journal of Scientific Research and Reports

Volume 29, Issue 12, Page 20-31, 2023; Article no.JSRR.110806
ISSN: 2320-0227

Analysis of Machine Reading Comprehension Problem Using Machine Learning Techniques

V. Kakulapati [a*], Gagganapalli Jithendhar Reddy [a], Koukuntla Kranthi Kumar Reddy [a], Putukapu Amarendhar Reddy [a] and Devender Reddy [a]

[a] *Sreenidhi Institute of Science and Technology, Yamnampet, Ghatkesar, Hyderabad, Telangana-501301, India.*

Authors' contributions

This work was carried out in collaboration among all authors. All authors read and approved the final manuscript.

Article Information

DOI: 10.9734/JSRR/2023/v29i121814

Open Peer Review History:
This journal follows the Advanced Open Peer Review policy. Identity of the Reviewers, Editor(s) and additional Reviewers, peer review comments, different versions of the manuscript, comments of the editors, etc are available here: https://www.sdiarticle5.com/review-history/110806

Original Research Article

Received: 14/10/2023
Accepted: 19/12/2023
Published: 20/12/2023

ABSTRACT

Machine reading Comprehension is a significant challenge in the field of natural language programming. In this problem, the objective is to read and grasp a given text passage before responding to questions that are dependent on the material. The most modern machine reading comprehension systems have accuracy levels that are superior to those of humans. On the other hand, when domains are switched, the majority of machine reading comprehension systems see a considerable drop in performance. However, certain machine reading comprehension systems have previously outperformed humans on a range of standard datasets, despite the evident and vast disparity among them. This is the case even though MRC models are not designed to read like humans. This demonstrates the need for enhancing the currently available datasets, assessment criteria, and models to progress the machine reading comprehension models toward "actual"

**Corresponding author: E-mail: vldms@yahoo.com;*

comprehension. In this work, the analysis of the machine reading comprehension problem performed by using logistic regression, K- nearest neighbor, and random forest. This strategy will include perspectives that are topic-oriented, concept-oriented, and time-oriented, and it will provide support for the summary of multilingual texts with the assistance of several machine reading comprehension models that are currently in development.

Keywords: Machine reading comprehension; model; Latent; machine learning; topic; concept; text; domain; multilingual.

1. INTRODUCTION

There are two main categories of machine reading comprehension, both of which aim to teach machines to infer answers from texts, namely extracting and non-extractive [1]. The goal of extractive machine reading comprehension is to train models to determine which parts of a reference book contain the answers to a given inquiry. Tasks like close-test [2] and span extraction [3] come to mind.

In low-resource languages, there is a lack of machine reading comprehension -based applications. By processing the language and transecting the response, however, the algorithm impressively simulates human talents. Since word embeddings (also known as word vectors) form the basis of many natural language processing techniques, deep learning is very helpful when working with them. When learning a representation for a given text, "word embedding" assigns the same representation to words that have the same meaning. This method involves representing each word as a real-valued vector in some kind of fixed vector space. Using a technique similar to a neural network, the numerical values of vectors representing individual words are learned. Because it solves so many issues with natural language processing, this phenomenon has been heralded as a major advance in deep learning [4].

As a consequence, NLP has been the subject of substantial study, and it is effective at several tasks, such as machine reading comprehension [5,6], automatic conversion [7,8], and linguistic interpretation [6]. In light of their latest grant, machine reading comprehension’s Arianna Dulizia was the associate editor in charge of organizing the evaluation of this manuscript and giving her stamp of approval before publication. Reading a complete text that pertains to a question and comprehending its context to deliver a response to that question is an activity that requires a substantial level of focus. This method is essential in many situations, including recommendation systems, question responding, and discourse, and it is comparable to the common human effort of reading comprehension. Therefore, reading comprehension robots aid humans inefficiently and conveniently obtaining knowledge. The Stanford Question Answering Dataset (SQuAD) [9], WikiQA [10], NewsQA [11], and TriviaQA [12] are only a few of the datasets utilized in recent works that suggest methods for using large-scale datasets for machine reading comprehension [13]. The basic building blocks of machine reading comprehension datasets are the context-query-response pairs. Most extant machine reading comprehension datasets have well-written contexts with enough evidence to answer the inquiry [8].

Specifically, we use a logistic regression model to solve this issue. The accuracy of the trained model is 29%, which means that out of all the papers, 29% had the blank properly filled in. A random estimate of a term for each document in the collection would have an accuracy of about 4%, given that most documents include approximately 25 entities. In other words, this model does a respectable job!

The results of the logistic regression model are satisfactory, but they are far from "human-like." The model is only 29% accurate. Now, the question is how we might improve our model's capacity for learning [14].

2. RELATED WORK

Research on MRC has been conducted mostly in English but also several other languages [15]. There are several sectors and uses for MRC methods.

The first machine reading comprehension systems appeared in the 1970s, with the most noteworthy being Lehnert's QUALM system [16]. Unfortunately, the magnitude and scope of this system prevented it from finding widespread use. In the 1980s and 1990s, machine reading comprehension research was largely disregarded. The authors' focus in yet another study was squarely on machine reading comprehension initiatives. An machine reading comprehension dataset with story content suited for third through sixth grade and five "wh" (what, where, when, why, and who) questions was made public in 1999 [17]. Formed a bag-of-words approach, whereby texts, including queries and environments, were depicted as a collection of phrases, and chosen terms occurring throughout the query, and background information were the response. Previously, machine reading comprehension issues were often handled using conventional procedures or algorithms. A rule-based machine reading comprehension method known as Quarc [18] is designed to answer "wh" questions of different issues via the use of morphological modeling, namely part-of-speech labeling, ontological category labeling, and object recognition. Bootstrapping, Markov logic, and autonomous training were among the ML methods utilized [19]. However, the following are some of the drawbacks of using such approaches: To begin, they rely heavily on rules or characteristics that have been painstakingly developed by humans. Second, such algorithms can't generalize; therefore, their effectiveness may suffer when confronted with massive datasets, including a wide variety of articles. Lastly, certain classic methods fail to extract contextual information and do not take into account long-range relationships.

The majority of machine reading comprehension jobs take the form of textual question responses, although there are many more possible configurations. According to Lucy Vanderwende [20], automated text comprehension is a kind of machine reading. "The capacity to correctly answer questions about a book has been used as a proxy for readers' comprehension of that work. The capacity to formulate relevant questions in response to a given material is one alternative means of gauging a person's level of comprehension. In reality, there are a plethora of benchmark datasets available specifically for this purpose. The conversational machine reading comprehension dataset ShARC [21] is one such example. Unlike with previous contemporary machine reading comprehension datasets, the machine addressing issues in the ShARC must infer additional information not available in the context to come up with the correct answer. When engaging in a ShARC conversation, the first inquiry usually does not provide enough context for a quick answer. So it's up to the computer to initiate the second inquiry; after it has amassed sufficient information, it may consider answering the first. Similarly, RecipeQA [22] is a collection of data for multi-modal analysis of demonstrated methods. In RecipeQA, the ordering task is one of four different types of challenges. A model's ability to properly organize a series of representative photos of a recipe is evaluated using an ordering challenge. The setting of this visual exercise is a set of recipe names and descriptions, similar to those of earlier challenges. To achieve this, the technique has to understand the chronological relationship between options, such as "boiling the water first, putting the spaghetti next," so that the ordered sequence of images matches the given pattern. Task delegation is another feature of MS Marco [23].

Adding further NLP knowledge to the machine reading comprehension study might be useful by constructing an attention-based NMT system [24] to address the problem of text comprehension in languages other than English. The NMT system interprets the non-English question pair into English. so that the English-extracting understanding of the texts system may answer; the system's attention-weighted scores are subsequently utilized to organize the replies in the target language. Machine reading comprehension activities may also benefit from the incorporation of new information. When completing the machine reading comprehension job, the developers relied on syntactic information to narrow their focus. By using syntactic dependence of interest (SDOI), they developed an SDOI-SAN that performed at the cutting edge of the field in the SQuAD 2.0 challenge. The results of more than 40 of the most popular deep learning text classification datasets were summarized, along with the results of more than 150 other methods. Several of the methods described here have already been used on existing machine reading comprehension initiatives.

3. METHODOLOGY

Algorithm 1 describes the whole two-step procedure. The first fine-tuning step involves supervised training on the labeled dataset Ds to further refine the pre-trained machine reading comprehension model. Model M0 serves as a foundation for further work. The training and labeling procedure is repeated numerous times throughout the self-training phase. In order to train M0. 0 on the unlabeled dataset, we first used the model Mi to produce pseudo-labels for the dataset Dt at each iteration i.

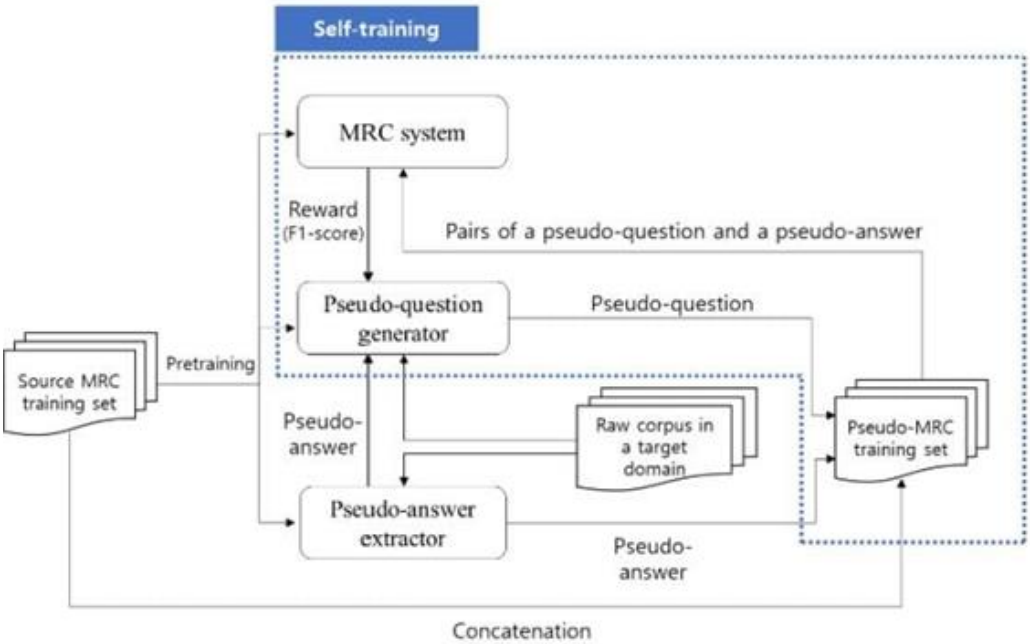

Chart 1. Framework for self-trained pseudo question generator

Here, one encounters several challenges:

- Initially, robots need to be taught how language works and what it means. Machines need to be taught how words work together in sentences before they can "understand" the meaning of a phrase the way a human can.

Secondly, the solution to a question may not be immediately apparent in the text. It is possible to spend a great deal of time poring through a piece of text without ever seeing the solution that is there. Since English is very flexible, a sequence of words that you are seeking could not show up word for word in the piece, making it much more difficult for robots.

The data set is collected from Kaggle which is freely available and after preprocessing generating the Question Generator class is defined, which is a transformer-based NLP system for generating reading comprehension-style questions from texts. It has several methods for generating questions, filtering low-quality questions, and evaluating the quality of generated QA pairs. The init method initializes the Question Generator object by loading a pre-trained question generator model and a QA evaluator model. It also sets some constants such as the answer token, context token, and sequence length.

The generate method takes an article (a string of text) and generates a set of question-and-answer pairs. It takes several optional arguments such as use_evaluator, num_questions, and answer_style. If use_evaluator is True, then QA pairs will be ranked and filtered based on their quality. The answer_style argument should be selected from ["all", "sentences", "multiple_choice"]. The generate_qg_inputs method takes a text as input and returns a list of model inputs and a list of corresponding answers. The data supplied format for the system is "answer_token answer text> context_token context text>," wherein the "answer" is a string of characters taken from the content, and the "context" is the remainder of the text. This method splits the text into segments, then splits each segment into sentences (if answer_style is "sentences" or "all"), and prepares the model inputs and answers for each sentence or segment. If answer_style is "multiple_choice" or "all", it prepares the model inputs and answers for multiple-choice questions.

The remaining methods are helper methods for generating and filtering questions. The _split_into_segments method splits a text into segments based on the number of sentences in each segment. The _split_text method splits a text into sentences. The _prepare_qg_inputs method prepares model inputs and answers for a sentence or segment of text. The

_prepare_qg_inputs_MC method prepares model inputs and answers for a multiple-choice question. The _get_all_qa_pairs method returns all generated QA pairs. The _get_ranked_qa_pairs method ranks generated QA pairs based on their quality and returns the top k pairs. generate_questions_from_inputs: Given a list of concatenated answers and contexts, generates a list of questions.
After generating questions, then

- _split_text: Splits the text into sentences and attempts to split or truncate long sentences.
- _split_into_segments: Splits a long text into segments short enough to be input into the transformer network. Segments are used as context for question generation.
- _prepare_qg_inputs: Uses sentences as answers and the text as context. Returns a tuple of (model inputs, answers). Model inputs are "answer_token <answer text> context_token <context text>".
- ._prepare_qg_inputs_MC: Extracts entities from text using NER (named entity recognition) and considers them while generating possible responses to MCQs. Sentences provide the background for questions, while entities provide the solutions. Obtains a tuple containing the model's inputs and outputs. Answer_token [answer text] context_token [context text] is the input to the model.
- ._get_MC_answers: Finds a set of alternative answers for a multiple-choice question. Will attempt to find alternatives of the same entity type as the correct answer if possible.
- _generate_question: Takes a qg_input, which is the concatenated answer and context, and uses it to generate a question sentence. The generated question is decoded and then returned.
- .generate_questions_from_inputs: Given a list of concatenated answers and contexts, generates a list of questions.
- _split_text: Splits the text into sentences and attempts to split or truncate long sentences.
- _split_into_segments: Splits a long text into segments short enough to be input into the transformer network. Segments are used as context for question generation.
- _prepare_qg_inputs: Uses sentences as answers and the text as context. Returns a tuple of (model inputs, answers). Model inputs are "answer_token <answer text> context_token <context text>".
- _prepare_qg_inputs_MC: Performs Named Entity Recognition (NER) on the text and uses extracted entities as candidate answers for multiple-choice questions. Sentences are used as context, and entities are used as answers. Returns a tuple of (model inputs, answers). Model inputs are "answer_token <answer text> context_token <context text>".
- _get_MC_answers: Finds a set of alternative answers for a multiple-choice question. Will attempt to find alternatives of the same entity type as the correct answer if possible.
- _generate_question: Takes a qg_input, which is the concatenated answer and context, and uses it to generate a question sentence. The generated question is decoded and then returned.

The first line imports the argparse module, which is used to parse command line arguments.

The next two lines import the QuestionGenerator class and the print_qa function from the pseudo_question_answer_generator module. The parse_args() function is defined to parse command line arguments. It creates an instance of the argparse. ArgumentParser class and adds several arguments to it using the add_argument () method. These arguments specify the desired type of answers, the model directory, the number of questions to generate, whether to show answers or not, the path to the input text file, and whether to use a QA evaluator. The function then returns a Namespace object that contains the parsed arguments. The if name == "main": block is the entry point of the script. It first calls the parse_args() function to parse the command line arguments, and then opens the input text file specified in the --text_file argument and reads its contents into a variable called text_file.

Run qa strings: This script generates questions and answers from a given text file using the QuestionGenerator class. Here's an explanation of how the code works:

- The argparse module is used to parse the command line arguments. The parse_args() function sets up an argument parser, adds the necessary arguments, and returns an object with the parsed arguments.
- In the if __name__ == "__main__": block, the script first parses the command line arguments using the parse_args() function.

- The script then reads the input text file specified in the arguments using the open() function and then with the statement.
- A QuestionGenerator object is initialized.
- The generate() method of the QuestionGenerator class is called to generate a list of question-answer pairs. The method takes in the input text, the desired number of questions to generate, the desired style of answers, and whether to use an evaluator to check the accuracy of the answers. The generated pairs are stored in the qa_list variable.
- The print_qa() function is called to print the question-answer pairs to the console. The function takes in the qa_list variable and whether to show the answers.
- The QuestionGenerator class and the print_qa() function are part of the pseudo_question_answer_generator module, which is used to generate question-answer pairs from text.

4. IMPLEMENTATION WORK

It is widely recognized that the size of the labeled dataset influences the performance of a pre-trained model used to fine-tune a subsequent task. The subsequent phase is to apply the method of training themselves to these kinds of models to check how the effectiveness of the learning strategy for the base model varies depending on the amount of the tagged dataset used for learning. To improve the self-training process, we doubled the size of the unlabeled dataset used. Fig. 4 shows that the basic model's evaluation effectiveness is consistently enhanced by the learning themselves technique, whereas the base model's fine-tuning is highly dependent on the quantity of domain-labeled data available. Nonetheless, the self-training strategy adds less and less to the overall effectiveness of the base model as the evaluation efficiency of the base mode l improves.

Fig. 1. Pseudo-answer strings

Table 1. Logistic Regression measures

	Precision	Recall	F1-Score	Support
0	0.89	0.86	0.88	102
1	0.86	0.89	0.87	98
Accuracy			0.88	200
Macro Avg	0.88	0.88	0.87	200
Weighted AVG	0.88	0.88	0.88	200

Fig. 2. Generating questions

Fig. 3. Question answer strings

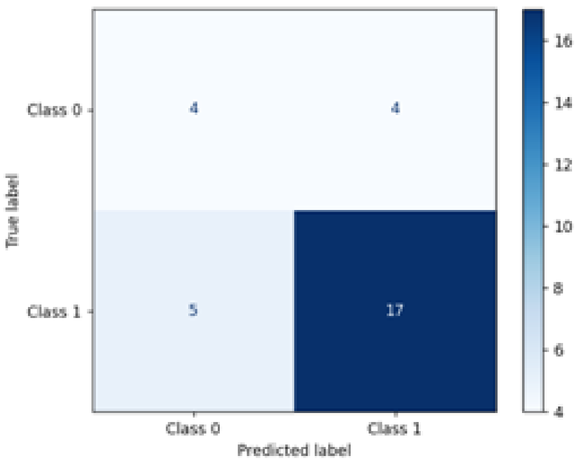

Fig. 4. Confusion Matrix of predicted questions

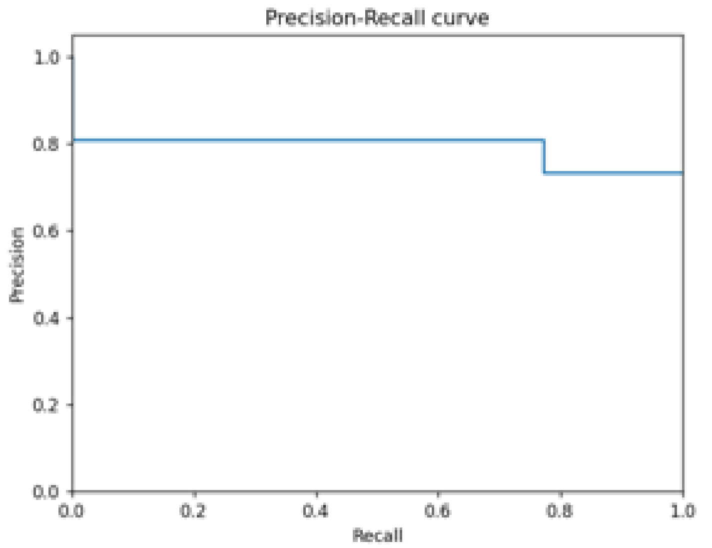

Fig. 5. Precision-Recall Curve

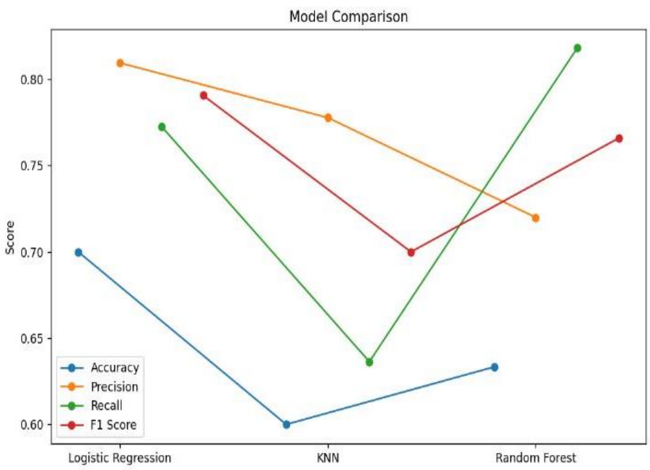

Fig. 6. A comparative analysis of different algorithms measures

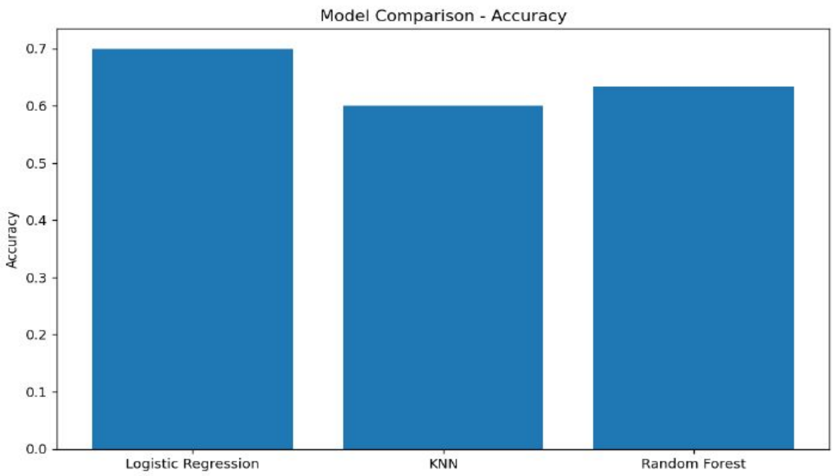

Fig. 7. Histogram comparative analysis of different algorithms measures

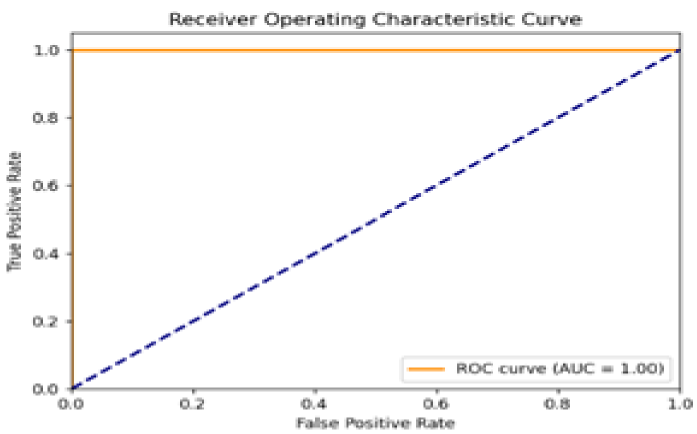

Fig. 8. Receiver Operating Characteristic Curve

5. DISCUSSION

The script then creates an instance of the QuestionGenerator class and assigns it to the variable qg. The generate () method of the qg object is called to create a list of question-answer pairs based on the input text. The method takes several arguments, including the input text, the number of questions to generate, the desired answer style, and whether to use a QA evaluator. The generated question-answer pairs are returned as a list and assigned to the variable qa_list. The print_qa() function is called to print the generated question-answer pairs. The function takes two arguments: the list of question-answer pairs and whether to show the answers or not, as specified by the --show_answers argument. That's a summary of what this script does. Overall, it's a useful tool for generating pseudo questions and answers from text, which could be helpful for tasks like building chatbots or generating study aids for students.

Humans rely heavily on their vast store of common sense and prior information to help them understand what they read. To achieve the same results with machine reading, a knowledge-based MRC is presented. The question of how to most efficiently integrate and use new information is continuing. One problem is that integrating text in context and questions with the structure of information held in knowledge bases is challenging. The success of knowledge-based MRC, on the other hand, depends heavily on the accuracy of the information it uses. Knowledge base construction is labor-intensive and time-consuming. Also, knowledge bases are often deficient, so it's not always easy to find relevant external information to back up response prediction and reasoning.

More research is needed on how to combine knowledge graphs with machine reading comprehension effectively.

6. CONCLUSION

MRC is an approach to learning that uses conversational techniques to enhance passage comprehension, such as the use of linked question and response procedures. The technology has shown promise in outperforming humans on benchmark datasets, but further advancement toward true understanding would need improved datasets, evaluation criteria, and models. Models developed using the MRC facilitate the summarization of texts in several languages from topical, conceptual, and temporal viewpoints. Using logistic regression, k-nearest neighbors, and random forest analysis This approach can aid in the summarization of multilingual texts with the use of numerous MRC models that are presently in development, and it will incorporate topic-oriented, concept-oriented, and time-oriented views.

7. FUTURE ENHANCEMENT

Through the use of self-training, we produce pseudo-labeled training data with which to train the model and boost its accuracy and generalization effectiveness. Nevertheless, training a model repeatedly using our self-training method takes a long time and is inefficient; therefore, we believe it might be improved upon. Unfortunately, our methodology is limited to only those three jobs and won't work for any other MRC problems. In the future, we'd like to investigate more advanced multimodal approaches for automated text reading.

COMPETING INTERESTS

Authors have declared that no competing interests exist.

REFERENCES

1. Zhang Z, Yang J, Zhao H. Retrospective Reader for Machine Reading Comprehension. In Proceedings of the AAAI Conference on Artificial Intelligence, Virtual, 2–9 February 2021; AAAI Press: Palo Alto, CA, USA. 2021;35:14506–14514.
2. Xie Q, Lai G, Dai Z, Hovy E. Large-scale cloze test dataset created by teachers. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, 31 October–4 November. 2018;2344–2356.
3. Inoue N, Stenetorp P, Inui K. R4C: A benchmark for evaluating RC systems to get the right answer for the right reason. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, Stroudsburg, PA, USA, 6–8 July 2020; Association for Computational Linguistics: Stroudsburg, PA, USA. 2020;6740–6750.
4. Alkhatnai Mubarak, Amjad Hamza Imam, Amjad Maaz, Gelbukh Alexander. Methods and Trends of Machine Reading Comprehension in the Arabic Language. Computación y Sistemas. 2020;24(4):1607-1615. Epub 11 de junio de 2021. Available:https://doi.org/10.13053/cys-24-4-3878.
5. Hu M, Peng Y, Huang Z, Qiu X, Wei F, Zhou M. ''Reinforced mnemonic reader for machine reading comprehension. In Proc. 27th Int. Joint Conf. Artif. Intell. Jul. 2018;4099–4106.
6. Brants T, Popat AC, Xu P, Och FJ, Dean J. Large language models in machine translation. In Proc. Joint Conf. Empirical Methods Natural Lang. Process. Comput. Natural Lang. Learn. 2007;858–867.
7. Liu X, He P, Chen W, Gao J. Multi-task deep neural networks for natural language understanding. In Proc. 57th Annu. Meeting Assoc. Comput. Linguistics. 2019;4487–4496.
8. Available:https://towardsdatascience.com/investigating-the-machine-reading-comprehension-problem-with-deep-learning-af850dbec4c0
9. Rajpurkar P, Zhang J, Lopyrev K, Liang P. ''SQuAD: 100,000+ questions for machine comprehension of text,'' in Proc. Conf. Empirical Methods Natural Lang. Process. 2016:2383–2392.
10. Yang Y, Yih W, Meek C. WikiQA: A challenge dataset for open-domain question answering,'' in Proc. Conf. Empirical Methods Natural Lang. Process. 2015;2013–2018. Trischler T, Wang X. Yuan J, Harris A, Sordoni P, Bachman, Suleman K. 'NewsQA: A machine comprehension dataset. In Proc. 2nd Workshop Represent. Learn. (NLP), Nov. 2016;191–200.
11. Joshi M, Choi E, Weld D, Zettlemoyer L. 'TriviaQA: A large scale distantly

supervised challenge dataset for reading comprehension,'' in Proc. 55th Annu. Meeting Assoc. for Comput. Linguistics. 2017;1601–1611.

12. Dunn M, Sagun L, Higgins M, Guney VU, Cirik V, Cho K. SearchQA: A new Q&A dataset augmented with context from a search engine.' Apr. 2017, arXiv:1704.05179. [Online]. Available: https://arxiv.org/abs/1704.05179.
13. Hu M, Wei F, Peng Y, Huang Z, Yang N, Li D. 'Read+verify Machine reading comprehension with unanswerable questions. in Proc. AAAI. 2019;33:6529–6537.
14. Yi Yang, Wen-tau Yih, Meek C. Wikiqa: A challenge dataset for open-domain question answering. Proceedings of the Conference on Empirical Methods in Natural Language Processing. 2015;2013–2018. DOI:10.18653/v1/D15-1237.
15. Lehnert WG. the process of question answering. Ph.D. Thesis, Yale University, New Haven, CT, USA; 1977.
16. Hirschman L, Light M, Breck E, Burger JD. Deep read: A reading comprehension system. In Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics on Computational Linguistics, College Park, MD, USA, 20–26 June 1999; Association for Computational Linguistics: Stroudsburg, PA, USA. 1999;325–332.
17. Riloff E. Thelen M. A rule-based question answering system for reading comprehension tests. In Proceedings of the 2000 ANLP/NAACL Workshop on Reading Comprehension Tests as Evaluation for Computer-based Language Understanding Sytems—Volume 6, Seattle, WA, USA, 4 May 2000; Association for Computational Linguistics: Stroudsburg, PA, USA. 2000;13–19.
18. Poon H, Christensen J, Domingos P, Etzioni O, Hoffmann R, Kiddon C, Lin T, Ling X, Ritter A, Schoenmackers S, et al. Machine Reading at the University of Washington. In Proceedings of the NAACL HLT 2010 First International Workshop on Formalisms and Methodology for Learning by Reading, Los Angeles, CA, USA, 6 June 2010; Association for Computational Linguistics: Stroudsburg, PA, USA. 2010;87–95.
19. Vanderwende L. Answering and Questioning for Machine Reading. AAAI Spring Symposium: Machine Reading. 2007;91.
20. Saeidi M, Bartolo M, Lewis P, Singh S, Rocktäschel T, Sheldon M, Bouchard G, Riedel S. Interpretation of natural language rules in conversational machine reading. Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing; Association for Computational Linguistics: Brussels, Belgium. 2018;2087–2097. DOI:10.18653/v1/D18-1233.
21. Yagcioglu S, Erdem A, Erdem E, Ikizler-Cinbis N. RecipeQA: A challenge dataset for multimodal comprehension of cooking recipes. Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing; Association for Computational Linguistics: Brussels, Belgium. 2018;1358–1368. DOI:10.18653/v1/D18-1166.
22. Asai A, Eriguchi A, Hashimoto K, Tsuruoka Y. Multilingual extractive reading comprehension by runtime machine translation; 2018.
23. Zhang Z, Wu Y, Zhou J, Duan S, Wang R. SG-Net: Syntax-guided machine reading comprehension. Proceedings of the Thirty-Fourth AAAI Conference on Artificial Intelligence (AAAI 2020); 2020.
24. Minaee S, Kalchbrenner N, Cambria E, Nikzad N, Chenaghlu M, Gao J. deep learning based text classification: A comprehensive review; 2020, [arXiv: cs.CL/2004.03705].

Peer-review history:
The peer review history for this paper can be accessed here:
https://www.sdiarticle5.com/review-history/110806