A Question-Answering System that Can Count

Abbas Saliimi Lokman, Mohamed Ariff Ameedeen, and Ngahzaifa Ab. Ghani

Abstract This paper proposes a conceptual architectural design of Question-Answering (QA) system that can solve "counting" problem. Counting problem is the inability of QA system to produce numerical answer based on retrieved rationale (in text passage) containing list of items. For example, consider "How many items are on sale?" as question and "Currently shampoo, soap and conditioner are on sale" as retrieved rationale from text passage. Normally, system will produce "shampoo, soap and conditioner" as an answer while the ground truth answer is "three". In other words, system is simply unable to perform the counting process needed in order to correctly answer such questions. To solve this problem, QA system architecture with following components is proposed: (1) A classifier to determine if given question requires a counting answer, (2) A classifier to determine if current system's answer is not numeric, and (3) A counting method to produce numerical answer based on given rationale. Despite looking like a whole system, the proposed architecture is actually a modular system whereby each component can operate independently (allowing each component to be separately implemented by other systems). In essence, this paper intents to demonstrate a general idea of how the defined problem can be solved using a modular system, that hopefully also opens up more flexible enhancements in the future.

Keywords QA system · Natural language processing · Machine learning

This work was supported in part by Department of Higher Education, Ministry of Education Malaysia under the Fundamental Research Grant Scheme (FRGS), through Universiti Malaysia Pahang (Ref: FRGS/1/2018/ICT02/UMP/02/12).

A. S. Lokman (✉) · M. A. Ameedeen · N. Ab. Ghani
Faculty of Computing, College of Computing and Applied Sciences, Universiti Malaysia Pahang, Lebuhraya Tun Razak, Gambang, Kuantan, Pahang 26300, Malaysia
e-mail: abbas@ump.edu.my
URL: http://www.ump.edu.my

R. Alfred et al. (eds.), *Computational Science and Technology*, Lecture Notes in Electrical Engineering 724, https://doi.org/10.1007/978-981-33-4069-5_6

1 Introduction

Within the software engineering field [16, 17], Question-Answering (QA) system is a computerized algorithm that produces output (answer) based on user's input (question) through the usage of Natural Language Processing (NLP). Relatively similar to chatbot [13–15], QA system will produce answer based on rationale (a span of text) extracted from a given text passage/corpus. To find a correct rationale, system must understand the question's context whether it is direct (no polysemy involved), or indirect (has some sort of polysemy or hidden meaning). To understand context in natural human language, the system must first understand the language itself (primarily the meaning of each word in the sentence). Over the years, NLP researchers have come up with a system called Language Model that can "understand" human language by learning word co-occurance patterns.

Mathematically, Language Model (LM) is a probability distribution system that produces probability values for each word in the text sequence (typically large sequence such as collection of documents). By predicting occurrence probability of words-against-words, relationship of semantic emerges ("a word is characterized by the company it keeps"—[5]). This relationship is what makes LM "understand" human language, that is not perceiving word as an atomic item, but as surrounding words that relate to it. While LM is good at understanding language, it still needs to be fine-tuned in order to perform well in task specific processes (such as QA). This is because LM only understands language but not how to use it (like human understand cars but need to learn how to drive it).

To fine-tune LM for QA system, a specific QA dataset is used. QA dataset generally is a collection of human answers towards human questions in regard to text passages (a reading comprehension exercise). Three most referred QA datasets to date are SQuAD (The Stanford Question Answering Dataset) [19], CoQA (A Conversational Question Answering Challenge) [20] and QuAC (QuAC: Question Answering in Context) [3]. For all datasets, fine-tuned LMs (with additional components) are shown to perform well [8, 9]. In general, evaluation is done based on how well the system performs towards overall data points. With majority correctness as main target, minor error is not given much attention. One of such error is identified as "counting" problem [8].

Counting problem in QA system is an error where the produces answer for "How many" type question is not numerical. As an example, consider the question "How many items are on sale?". With good fine-tune LM, QA system can retrieved following rationale from a given passage "Currently shampoo, soap and conditioner are on sale". Without solution to counting problem, QA system then will answer "Shampoo, soap and conditioner" while in fact the ground truth answer is "Three". Because the QA system is not able to solve the counting problem, it can only produce answer that is a list of relevant items based on identified context in the question.

Following are contributions of this paper:

– A proposal of QA system architecture that can solve the counting problem.
– A proposal of three independent components in modular system setup that can be assembled in order to solve the counting problem.

2 Related Work

Current trend in NLP is pretrained Language Model (pLM), that is a generalization of LM. pLM is an LM that can independently be trained to relatively understand any textual human language. After being trained, pLM can be fine-tuned in order to make it perform well on specific downstream Natural Language Understanding (NLU) tasks. Example of such task are single and pair sentence classification, sentence tagging, reading comprehension, and so on [21]. Among currently famous pLM are GPT-3 [2], BERT (and it's variations) [4, 7, 10–12], XLNET [25] and ELMo [18]. For QA systems, "reading comprehension" is the core NLU downstream task that needs to be addressed. To make pLM able to perform well on this task, fine-tuning is done using QA datasets such as SQuAD [19], CoQA [20] or QuAC [3]. In general, those datasets contain three interrelated data: (1) A text passage about a particular subject, (2) A factual question related to the passage (multiple questions per passage), and (3) An answer to each question (to be noted that each dataset has its own quirks but basic data structure is fairly similar).

Fine-tuning pLM might address basic QA system requirement but there is still more to be improved. The basic two steps approach for QA system in answering question is: (1) Looks for factual information (a rationale) in text passage based on semantic information in the question (using learned QA relationship patterns), and (2) Produce answer based on retrieved rationale (rationale is a span of text from the referred passage). Because each dataset/domain has its own quirks and features, researchers augment pLM with various modules to make it perform better in regards to each dataset. Following are some of those augmented implementation: Wen et al. augment pLM with specific module to address why-question in clinical domain QA system [23], Banarjee et al. augment abductive information retrieval method to pLM in order to address open book QA reasoning [1], Wang et al. augment pLM with scoring mechanism for multi-passage answer retrieval in attempt to improve open domain QA system [22], Yang et al. augment specific information retrieval toolkit called Anserini to address QA system with large passage sequence [24], and Godbole et al. augment IR technique that able to address multi-hop QA requirement [6].

Among those augmented modules, none has addressed "the counting problem" issue. Although not presented much in those datasets (only 5.1% in CoQA [20] dataset and even smaller percentage in others), the counting problem is everywhere in everyday human conversation. One such conversation domain is buying-and-selling where "how many" question type/style is used a lot when dealing with items on sale, in stock, in packaging and so on. With this realization, this paper intents to demonstrate a general idea of how the counting problem can be solved using a modular system. With modularization architecture, proposes system's modules can be augmented into current system as to improve SoTA results on "how many" type questions. Proposes system is also unbiased towards knowledge domain, that is because its targeted on non-contextual word for question type classification (further explanation is in subsection 3.2 "Question type classifier (C1)").

3 Architecture

This paper proposes a QA system architecture that is designed upon two main objectives: (1) To solve the counting problem in system's answer, and (2) To become reusable in other systems. While objective one is very clear (the main objective), objective two requires some elaboration. To make a system that is reusable for other systems, it's components need to be modular (independent units). To achieve this, the propose system must maintain the fundamental process of how QA system produce an answer, that is by retrieving rationale from the given passage. By maintaining this process, answers that do not suffer from the counting problem can produce similarly as before. Following Fig. 1 depicts the proposed architecture (the whole system) in data flow format.

Referring to Fig. 1, general input-output process for proposed architecture is fundamentally similar to other QA system where system accepts two input texts (question and passage) and produce one output text (answer). Following the data flow, input will go through four system components: (1) Fine-tune language model (ftLM), (2) Question type classifier (Classifier 1 or C1), (3) Answer type classifier (Classifier 2 or C2), and (4) Counting method (Method 1 or M1). Apart from normal flow (going through all four components), data can also flow directly to the last stage from C1 or C2 component. These are cases where system can bypass subsequent component if it has been identified that the inputs did not suffer from the counting problem (further explanation is in subsection 3.4 "Counting method (M1)"). To be noted that ftLM is a required component for C1 and C2 (required for language and

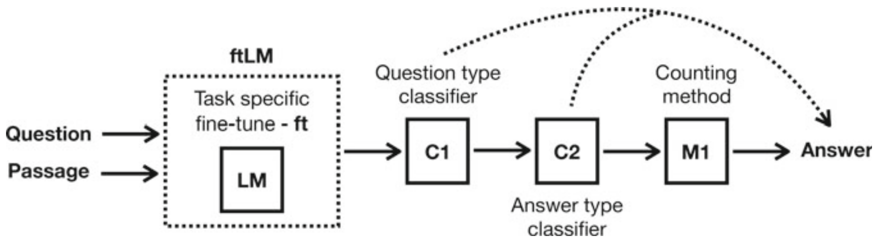

Fig. 1 General architecture

sentence comprehension task). As such, if C1 or C2 is to be used independently, ftLM is needed to be included as preprocessing module. Next subsection will elaborate more on each component.

3.1 Fine-Tune Language Model (ftLM)

ftLM is the first component that system's inputs need to go through. ftLM main objective is to retrieve rationale from input passage given input question's context. In details, ftLM needs to perform two interrelated processes which are: (1) Reading comprehension and (2) Rationale retrieval. Referring back to Fig. 1, ftLM (big dashed-line box) contains LM (small solid-line box) surrounded by "Task specific fine-tune" process *ft*. In correlation, LM is reading comprehension (process 1) module, and *ft* is rationale retrieval (process 2) module.

In QA system context, ft is a process of fine-tuning LM using QA dataset such as SQuAD, CoQA and QuAC. Conclusively, LM is used for system to comprehend pre-trained language (e.g. English language) and ft is used for system to guess semantic relation between question text and passage text. As overall architecture is designed to be modular, rationale extraction can be improved by embedding more component towards base ftLM module.

3.2 Question Type Classifier (C1)

C1 is a component that takes question and passage text as inputs, and classify it into one of two question types: (1) Question that does not requires numeric answer, and (2) Question that requires numeric answer (a binary classification problem). Following Fig. 2 depicted the propose C1 component design.

Figure 2 denoted ftLM as fine-tune Language Model (as in Fig. 1), Q as question text, R as ftLM's resulting rationale, A1 as preprocessor, A2 as Neural Network (NN), and A3 as Sigmoid function. As explained in the previous section, ftML will retrieve rationale from input passage given input question's context. From ftML, A1 will receive rationale and also the original question text as its inputs. A1 objective is to remove all context-related words from question input in order to get not-in-context

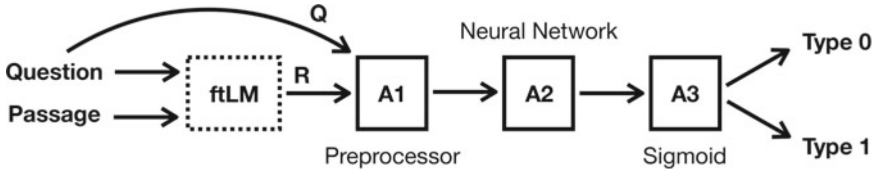

Fig. 2 Question type classifier

words which is the question words (e.g. “What”, “Where”, “How many” and so on). Following is the proposed algorithm for A1 subcomponent:

1. Get question text Q and rationale R as inputs
2. Get each word embedding for Q and R
3. Calculate cosine similarity $\cos(\theta)$ between each word in Q, towards each word in R
4. Remove word in Q that $\cos(\theta)$ value is within threshold t
5. Pass remaining Q words embedding to the next process.

Upon completion, A1 subcomponent module will pass embeddings of not-in-context question words into A2. A2 is a multilayer neural network with Sigmoid neuron (A3) at the end. Standard logistic function (where k = 1, L = 1, $x^0 = 0$) will be used for Sigmoid as C1 needs to predict binary class for question type which are: Type 0 - Question that does not requires numeric answer, and Type 1 - Question that requires numeric answer. Sigmoid calculation will produce a value within 0 to 1 range, that is a classification of Type 0 for 0 to 0.49 range value, and classification of Type 1 for 0.5 to 1 range value. For C1 training purpose, QA dataset will needs a label value where Type 0 is labeled 0, and Type 1 is labeled 1 (supervised ML model).

3.3 Answer Type Classifier (C2)

The purpose of C2 component is to classify whether ftLM (that has been fine-tune for QA system) produce a numeric answer or not. Similar to C1, C2 needs to solve binary classification problem with natural language text as its inputs. Following Fig. 3 depicted the propose C2 component design.

Figure 3 denote ftLM as fine-tune Language Model (as in Figs. 1 and 2), B1 as Neural Network (NN), and B2 as Sigmoid function. From ftML, B1 will receive rationale text embeddings as input. Similar to A2 (Fig. 2), B1 is a multilayer neural network with standard logistic function Sigmoid neuron (B2) at the end. Binary classes to be classified by B1 are: Type 0—Answer that is semantically not numeric, and Type 1—Answer that is semantically numeric. To be a “semantically numeric” answer, rationale text must contains numerical-valued text, be it in actual number (e.g. 1, 2, 3) or text representing number (e.g. one, two, three). As for training process, C2

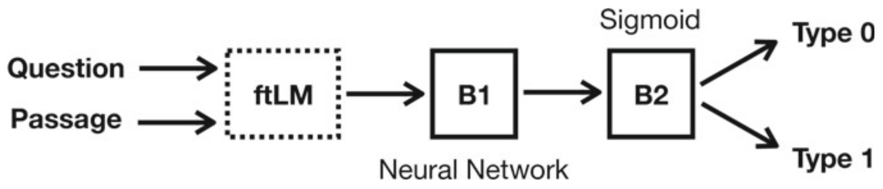

Fig. 3 Answer type classifier

will use standard QA dataset with additional label value that is similar to C1 training requirement where Type 0 is labeled 0, and Type 1 is labeled 1 (also a supervised ML model).

3.4 Counting Method (M1)

To perform a counting process, C1 and C2 components need to satisfy following two conditions: (1) Input question is a type that requires numeric answer (Type 1 for C1 classifier), and (2) ftLM outputted rationale is not semantically numeric (Type 0 for C2 classifier). By basic logic, condition one is prerequisite to condition two. This is because the counting process is only needed for question that requires a numeric answer. Following Fig. 4 depict path that needs to be satisfied by the data flow in order for M1 to be activated.

M1 component objective is to receive natural language text as input (in a list of items format), and produce the summation of items as output. As powerful as ML is, it is still bound to basic probability principle which is “predicting a value”. In mathematical calculation, the produced answer is definite therefore no amount of data can be feed into ML to predicts every possible outcome of mathematical calculation (because real numbers are infinite). As such, M1 requires traditional computing method in calculating summation of items in a list. Following is proposed algorithm for M1 component:

1. Get answer text A as input
2. Segment A text into item array Ar through lexical analysis
3. Loop Ar to count items presented in A (count++)
4. Output final count value.

When complete, M1 will produce a count value (the summation of items) in number format. As QA system usually uses text format, a conversion is needed. In usual formal writing, number 1 to 10 is written in text format while number 11 and above is written in number format. On that account, count value only needed to be converted into text when its value is under 10. Maintaining this scope will make it simpler for conversion process as real numbers are infinite.

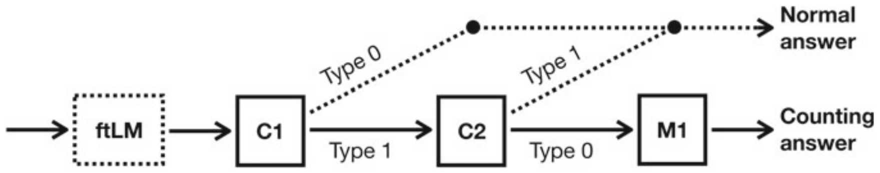

Fig. 4 M1 activated path

4 Discussion and Future Work

This paper proposes a QA system general architecture to solve the counting problem using modular four parts components. This proposed system differs from previous works in two regards: (1) A QA system that specifically intended to solve the counting problem while previous works focused on achieving SoTA result on whole datasets, (2) A modular system component that can also be implemented in other case study (such as binary question/answer text classification), while previous works proposed a close architecture system for specific QA dataset only.

Similar to other conceptual proposal, proof of concept implementation is crucial. For presented architecture, first task is to independently test and verify the three proposed components (C1, C2 and M1). Other than to make sure each component works as intended, this task is also to justify the modular design proposal. After all components are throughly validated, next task is to ensemble all components to become one functional QA system. With this ensemble system, standard QA system benchmark data can be used to test and evaluate the whole architecture. Keep in mind that this system might not achieve SoTA result since its main focus is to solve counting problem. It is however possible for other components to be incorporated later (in order to achieve new SoTA) as the system is modular in design. In addition to standard research datasets, this system can also be validated using real world human conversational datasets as to further justify the needed modules for solving the counting problem in QA domain.

5 Conclusion

This paper proposes a conceptual architecture for QA system that intend to solve counting problem in system's generated answer. Proposed architecture is designed to be modular in a sense that each component can work independently. This is to allow other systems to able to embed just the needed component (without the whole system) effortlessly. To solve the counting problem, three components (excluding generalized language model) are proposed within one ensemble system. Those components are (1) Question type classifier, (2) Answer type classifier and (3) Counting method. Component one and two are machine learning based, while component three is traditional programming method. Collectively, all components can be used to solve counting problem while independently, each component can be used to solve its predefined purpose (binary classification for component one and two, and syntactic mathematical addition problem for component three).

References

1. Banerjee P, Pal KK, Mitra A, Baral C (2019) Careful selection of knowledge to solve open book question answering. arXiv preprint arXiv:1907.10738
2. Brown TB, Mann B, Ryder N, Subbiah M, Kaplan J, Dhariwal P, Neelakantan A, Shyam P, Sastry G, Askell A, Agarwal S (2020) Language models are few-shot learners. arXiv preprint arXiv:2005.14165
3. Choi E, He H, Iyyer M, Yatskar M, Yih WT, Choi Y, Liang P, Zettlemoyer L (2018) Quac: question answering in context. arXiv preprint arXiv:1808.07036
4. Devlin J, Chang MW, Lee K, Toutanova K (2018) Bert: pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805
5. Firth JR (1957) A synopsis of linguistic theory, 1930–1955. Studies in linguistic analysis
6. Godbole A, Kavarthapu D, Das R, Gong Z, Singhal A, Zamani H, Yu M, Gao T, Guo X, Zaheer M, McCallum A (2019) Multi-step entity-centric information retrieval for multi-hop question answering. arXiv preprint arXiv:1909.07598
7. Joshi M, Chen D, Liu Y, Weld DS, Zettlemoyer L, Levy O (2020) Spanbert: improving pre-training by representing and predicting spans. Trans Assoc Comput Linguistics 8:64–77
8. Ju Y, Zhao F, Chen S, Zheng B, Yang X, Liu Y (2019) Technical report on conversational question answering. arXiv preprint arXiv:1909.10772
9. Lan Z, Chen M, Goodman S, Gimpel K, Sharma P, Soricut R (2019) Albert: a lite bert for self-supervised learning of language representations. arXiv preprint arXiv:1909.11942
10. Lee J, Yoon W, Kim S, Kim D, Kim S, So CH, Kang J (2020) BioBERT: a pre-trained biomedical language representation model for biomedical text mining. Bioinformatics 36(4):1234–40
11. Liu W, Zhou P, Zhao Z, Wang Z, Ju Q, Deng H, Wang P (2020) K-BERT: enabling Language Representation with Knowledge Graph. arXiv preprint arXiv:1909.07606
12. Liu Y, Ott M, Goyal N, Du J, Joshi M, Chen D, Levy O, Lewis M, Zettlemoyer L, Stoyanov V (2019) Roberta: a robustly optimized bert pretraining approach. arXiv preprint arXiv:1907.11692
13. Lokman AS, Ameedeen MA (2019) Modern chatbot systems: a technical review. In: Arai K, Bhatia R, Kapoor S (eds) Proceedings of the Future Technologies Conference (FTC) 2018. FTC 2018. Advances in Intelligent Systems and Computing. Springer, Cham, vol 881, pp 1012–1023
14. Lokman AS, Ameedeen MA, Ghani NA (2020) A conceptual IR chatbot framework with automated keywords-based vector representation generation. IOP Conf Ser Mater Sci Eng 769(1):012020 IOP Publishing
15. Lokman AS (2011) Chatbot development in data representation for diabetes education. MCS Thesis, Universiti Malaysia Pahang, Pahang, Malaysia
16. Ong MIU, Ameedeen MA, Azmi ZR, Kamarudin IE (2018) Systematic literature review: 5 years trend in the field of software engineering. Adv Sci Lett 24(10):7278–7283
17. Ong MIU, Ameedeen MA, Kamarudin IE (2018) Meta-requirement method towards analyzing completeness of requirements specification. In: Arai K, Bhatia R, Kapoor S (eds) Proceedings of the Future Technologies Conference (FTC) 2018. FTC 2018. Advances in intelligent systems and computing. Springer, Cham, vol 881, pp 444–454
18. Peters ME, Neumann M, Iyyer M, Gardner M, Clark C, Lee K, Zettlemoyer L (2018) Deep contextualized word representations. arXiv preprint arXiv:1802.05365
19. Rajpurkar P, Jia R, Liang P (2018) Know what you don't know: unanswerable questions for SQuAD. arXiv preprint arXiv:1806.03822
20. Reddy S, Chen D, Manning CD (2019) CoQA: a conversational question answering challenge. Trans Assoc Comput Linguistics 7:249–266
21. Wang A, Singh A, Michael J, Hill F, Levy O, Bowman SR (2018) Glue: A multi-task benchmark and analysis platform for natural language understanding. arXiv preprint arXiv:1804.07461
22. Wang Z, Ng P, Ma X, Nallapati R, Xiang B (2019) Multi-passage bert: a globally normalized bert model for open-domain question answering. arXiv preprint arXiv:1908.08167

23. Wen A, Elwazir MY, Moon S, Fan J (2019) Adapting and evaluating a deep learning language model for clinical why-question answering. arXiv preprint arXiv:1911.05604
24. Yang W, Xie Y, Lin A, Li X, Tan L, Xiong K, Li M, Lin J (2019) End-to-end open-domain question answering with bertserini. arXiv preprint arXiv:1902.01718
25. Yang Z, Dai Z, Yang Y, Carbonell J, Salakhutdinov R, Le QV (2019) XLNet: generalized autoregressive pretraining for language understanding. arXiv preprint arXiv:1906.08237