A Flexible and Extensible Framework for Multiple Answer Modes Question Answering

Cheng-Chung Fan
Shang-Bao Luo
Kuang-Yu Chang
Meng-Tse Wu
Tzu-Man Wu
Chao-Chun Liang
Kuan-Yu Chen◆
Keh-Yih Su
Institute of Information Science, Academia Sinica
▲ Research Center for Information Technology Innovation, Academia Sinica
{jjfan, newsboy3423, simonc, cwhsu, moju, doublebite, tzum.wu, alsm, ccliang, whm, kysu}@iis.sinica.edu.tw,

Chia-Chih Kuo◆
Pei-Jun Liao
Chiao-Wei Hsu
Shih-Hong Tsai
Aleksandra Smolka
Hsin-Min Wang
Yu Tsao▲
◆ Department of Computer Science and Information Engineering, National Taiwan University of Science and Technology
{jerrykuo7727, s2w81234}@gmail.com
kychen@mail.ntust.edu.tw
yu.tsao@citi.sinica.edu.tw

Abstract

This paper presents a framework to answer the questions that require various kinds of inference mechanisms (such as Extraction, Entailment-Judgement, and Summarization). Most of the previous approaches adopt a rigid framework which handles only one inference mechanism. Only a few of them adopt several answer generation modules for providing different mechanisms; however, they either lack an aggregation mechanism to merge the answers from various modules, or are too complicated to be implemented with neural networks. To alleviate the problems mentioned above, we propose a divide-and-conquer framework, which consists of a set of various answer generation modules, a dispatch module, and an aggregation module. The answer generation modules are designed to provide different inference mechanisms, the dispatch module is used to select a few appropriate answer generation modules to generate answer candidates, and the aggregation module is employed to select the final answer. We test our framework on the 2020 Formosa Grand Challenge Contest dataset. Experiments show that the proposed framework outperforms the state-of-the-art Roberta-large model by about 11.4%.

Keywords: QA, Framework, Divide-and-Conquer strategy, Answer Aggregation, Inference mechanism

1 Introduction

Natural Language Inference (NLI) is an important topic in the *Artificial Intelligence* (AI) field, and any NLI related issue can be checked by asking an appropriate corresponding question (Chen, 2018). Therefore, the *Question Answering* (QA) task has become a very suitable testbed for evaluating NLI models and checking the progress of current techniques. Accordingly, the Ministry of Science and Technology of Taiwan has organized the *Formosa Grand Challenge Open Contest* series[1] (FGC) in 2018, which mainly evaluates the reasoning/inference capability on natural texts, to promote the AI progress in Taiwan. Specifically, this open contest covers a variety of *answer modes*; that is, it needs different *inference mechanisms* (such as *Extraction*, *Entailment-Judgement*, *Aggregative-Operation*,

[1] https://fgc.stpi.narl.org.tw/activity/techai2018

etc.) to get the desired answer. As a result, the system/framework must be able to handle various answer modes at the same time.

The previous frameworks for the QA task could be classified into two main categories according to the number of answer modules adopted: (1) Single answer generation module (Trischler et al., 2017; Chen, 2018; Shoeybi et al., 2020; Zhang et al., 2020), which involves only one answer mode, and allows merely one type of replying format (such as identifying a span within the given passage, giving YES/NO answer, free text reply, etc.). (2) Multiple answer generation modules (Ferrucci, 2012; Andor et al., 2019; Hu et al., 2019), which adopts several answer generation modules, and each module conducts a specific inference mechanism (or, answer mode) with a specific replying format.

Since the first category only considers one answer mode, the types of questions that can be handled are quite limited. For example, it is not suitable for handling the FGC-2020 QA task[2], which covers various question types and needs different answer modes to get the desired answers. In contrast, the approaches under the second category adopt the *divide-and-conquer* strategy, which adopts a different answer generation module for each specific answer mode. Since each answer generation module only needs to consider a specific answer mode, it will be easier to design and add new inference mechanisms.

Among those second category approaches, the framework of Watson (Ferrucci, 2012) is not designed for end-to-end training; therefore, it is not suitable for modern neural-network multi-task learning due to the complicated flow/architecture under its statistics-based architecture. Also, the framework adopted in either (Andor et al., 2019) or (Hu et al., 2019) does not have an aggregation layer/module to merge the answers generated from different answer generation modules (i.e., the output is only picked from a specific module, and merging is not allowed). Therefore, their approaches not only have the error accumulation problem[3] (i.e., once a wrong module is selected, this error will propagate to the next answer-generation stage), but also lose the advantage of combining the strength of different inference mechanisms. Additionally, all modules will be activated in parallel under their frameworks (Andor et al., 2019), so computing resources on those modules that should not be activated for a given question would be wasted.

To overcome the problems mentioned above, a flexible and extensible framework is proposed in this paper. It adopts a divide-and-conquer strategy, and possesses the following main modules/functionalities: (1) A *supporting evidences locating module*, which extracts supporting evidences from the passage to narrow down the searching space. (2) A *dispatch module*, which would select and activate several appropriate answer generation modules; also, the answer type distribution will be provided to each answer generation module as a reference, based on the answer mode. (3) A set of *answer generation modules*, each of them generates a few local/module outputs (i.e., possible answers) if it is activated. (4) An *aggregation module*, which picks the best answer at the final stage by merging the answer candidates from those activated answer generation modules.

The strengths of the proposed framework are summarized as follows: (1) With the dispatch module, it is flexible for handling different question types with the same framework; as a result, it is extensible for adding more answer modes in the future. (2) With the aggregation module, it is able to merge the results from various modules; it thus possesses the capability of combining the strength of different inference mechanisms, and also reduces the error accumulation problem. (3) It is designed to fit the neural-network based end-to-end multi-task learning framework; therefore, it can be implemented with an appropriate neural network without much effort. (4) Since the dispatch module only activates the corresponding modules according to the given question, it will not waste computing resources on those modules that are irrelevant and should not be activated.

In comparison with IBM Watson framework, which adopts a complicated flow/architecture with probabilistic models, our proposed framework

[2] https://scidm.nchc.org.tw/dataset/grandchallenge2020

[3] The error accumulation problem of this kind of approaches is hard to avoid, as it is difficult to know which inference mechanism should be adopted before we actually see the related supporting statements (e.g., span-extraction mechanism is usually preferred if the desired answer is explicitly given in the supporting sentence; otherwise, a more complicated mechanism must be adopted).

adopts the neural-network based approach and can be optimized by the end-to-end training strategy. In comparison with the approaches from Andor et al. (2019) and Hu et al. (2019), which lack the mechanism to merge different answer candidates, our proposed framework only activates several possible/responsible modules and has the ability to aggregate the outputs from various modules.

The proposed framework is tested on the FGC-2020 QA dataset, which contains 1,322 questions. This dataset covers eight different answer modes (i.e., *Single-Span-Extraction*, *Multi-Span-Extraction*, *Yes/No*, *Aggregative-Operation*, *Arithmetic-Operations*, *Date-Duration*, *Kinship*, and *Summarization*) and ten different answer types (i.e., *Yes/No*, *Number-Measure*, *Kinship*, *Person*, *Date-Duration*, *Location*, *Organization*, *Object*, *Event*, and *Misc*). The experiment results show that our system outperforms the baseline RoBERTa-large (Liu et al., 2019) model by 11.4%.

In summary, this paper makes the following contributions: (1) We propose a novel *modular* framework/model that is more flexible for handling/adding various inference mechanisms. (2) We propose a novel aggregation model to merge various answer candidates. (3) We conduct experiments to show that the proposed framework outperforms the state-of-the-art RoBERTa-large model on the FGC-2020 QA dataset.

2 The Proposed Approach

In this section, the proposed divide-and-conquer QA model is first described in Section 2.1. The descriptions of the architecture of the proposed model is then presented in Section 2.2. Afterwards, Section 2.3 provides the concepts and principles of designing each answer generation module.

2.1 The Proposed Divide-and-Conquer QA Model

Given a Document D, Question Q, Wikipedia W_k and some external Knowledge Resources R (such as WordNet and ConceptNet), we would like to find out the most likely answer. To reduce the computation cost, we will first extract related Wikipages with an off-the-shelf IR tool (e.g., the Apache Lucene™ searching engine[4]). Let *Wps* denote the set of extracted Wikipages, the problem of finding the desired Answer $\hat{A}$ thus can be formulized as Equation (1). For conciseness, we will only use one notation (e.g., "*D*" (Document)) to denote both its content and its associated embedding vector when it can be interpreted without confusion.

$$\begin{aligned}\hat{A} &= \operatorname{argmax} P(A|D,Q,W_k,R)\\ &\equiv argmax\, P\big(A\big|D,Q,W_{ps},R\big),\end{aligned} \tag{1}$$

where A is a specific answer candidate, and $\hat{A}$ denotes the desired answer which can be: (1) A list of *string/NE/number/date* directly extracted from the document. This list might contain only one element, or even empty (The string "UNKNOWN" will be output in this case). (2) An aggregation result (such as *Summarization, Speaker's View, Arithmetic Result*, *Count/Min/Max/Avg, Entailment/Sentiment Judgment,* etc.) induced from the given document.

Since we will encounter various scenarios that request different answer modes (among which each adopts a different strategy to obtain the desired answer), a *Divide-and-Conquer* framework is thus proposed to convert a given complicated problem into a set of simple sub-problems:

$$\begin{aligned}&P\big(A\big|D,Q,W_{ps},R\big)\\ &=\textstyle\sum_{M,T,E_s,G_s} P\big(A,M,T,E_s,G_s\big|D,Q,W_{ps},R\big),\end{aligned} \tag{2}$$

where M denotes a specific answer mode, T refers to a specific answer type that can be used for verification in each answer generation module, E_s stands for a specific set of supporting evidences, and G_s represents a specific set of paragraphs. By doing so, each answer generation module/model concentrates only on a specific answer mode. The probability $P(A,M,T,E_s,G_s|D,Q,W_{ps},R)$ can be further decomposed into five terms:

$$\begin{aligned}&P\big(A,M,T,E_s,G_s\big|D,Q,W_{ps},R\big)\\ &= P\big(A\big|M,T,E_s,G_s,D,Q,W_{ps},R\big)\\ &\times P\big(M\big|T,E_s,G_s,D,Q,W_{ps},R\big)\\ &\times P\big(T\big|E_s,G_s,D,Q,W_{ps},R\big)\times P\big(E_s\big|G_s,D,Q,W_{ps},R\big)\\ &\times P\big(G_s\big|D,Q,W_{ps},R\big)\\ &\approx P(A|M,T,E_s,Q,R)\times P(M|T,E_s,Q)\times\\ &P(T|E_s,Q)\times P(E_s|G_s,D,Q,W_{ps})\times P(G_s|D,Q,W_{ps}),\end{aligned} \tag{3}$$

where $P(A|M,T,E_s,Q,R)$ will be generated by each specific answer generation module, both $P(M|T,E_s,Q)$ and $P(T|E_s,Q)$ will be generated by

[4] https://lucene.apache.org/

the Dispatch module, $P(E_s|G_s, D, Q, W_{ps})$ will be generated by the Supporting-Evidence-Locating module, and $P(G_s|D, Q, W_{ps})$ will be generated by another Paragraph-Locating module (Section 2.2).

Finally, $\sum_{M,T,E_s,G_s} P(A, M, T, E_s, G_s|D, Q, W_{ps}, R)$ will be taken care by the Aggregation module, which aggregates various answer-candidates generated by different answer generation modules to obtain the final answer. It predicts the best answer based on those obtained answer-module sextuplets (i.e., <answer mode M, the probability of the answer mode M_p, answer type T, the probability of the answer type T_p, answer-candidate A, its associated confidence-scores F_s>, to be specified later), where M, M_p, T, and T_p are from the Dispatch module, both A and F_s are from a specific activated answer generation module. Therefore, Equation (2) can be re-written as

$$P(A|D, Q, W_{ps}, R) = \sum_{M,T,E_s,G_s} P(A, M, T, E_s, G_s|D, Q, W_{ps}, R) \equiv \text{softmax}\,\sigma\left(H\left(\begin{matrix}(M; M_p; T; T_p; F_s)_{A,1}, \dots, \\ (M; M_p; T; T_p; F_s)_{A,K}\end{matrix}\right)\right) \quad (4)$$

The above Eq (4) is implemented with a pre-processor, which first merges the same answer-candidate from various answer generation modules; afterwards, for each specific merged answer-candidate A (among a varying number of different merged candidates), it concatenates the corresponding information from each answer generation module[5] to form the input to a mapping function H. This mapping function H is mainly used to assign an overall-confidence-score to the given merged answer-candidate if it is supported/merged by/from several modules.

Specifically, for each merged answer-candidate A, we will have K different $(M; M_p; T; T_p; F_s)$ quintuplets, where K is a pre-specified/fixed number of available answer generation modules. Note that the relative position of each answer generation module within the concatenation is fixed (so that the corresponding NN weights can be learnt). The overall-confidence-score of A is input to a specific non-linear activation function σ, then a *softmax* function is used to normalize the obtained scores over various merged answer-candidates.

2.2 The Architecture and Operation Flow

Based on Equations (3) and (4), Figure 1 summarizes the proposed divide-and-conquer QA framework. Sequentially, the *Preprocessing-layer* first locates the related Wikipages and annotates the given question/passage (also those Wikipages) with their associated linguistic information via off-the-shelf language tools (e.g., the Stanford CoreNLP toolkit).

Afterwards, the *Embedding-layer* obtains contextual word embeddings through a pre-trained language model (e.g., BERT, RoBERTa (Liu et al., 2019) or XLNet (Yang et al., 2019)), and generates the associated hierarchical embeddings (including the document embedding, paragraph embeddings, and sentence embeddings). The hierarchical embeddings will be shared among subsequent layers.

The *Paragraph-Locating-layer* then narrows down the searching space to only refer to those closely related paragraphs/passages within documents/pages via the so-called "semantic retrieval" model (Nie et al., 2019).

The *Supporting-Evidence-Locating-layer* identifies the associated *Supporting Evidences* and also outputs an associated score of the specified configuration. Basically, only content similarity is considered here, and no reasoning is conducted (which will be done later in the Answer-Generation-layer). It can be implemented by a BERT-based model with output vectors connected to a binary classifier.

The *Dispatch-layer* generates the corresponding answer mode and answer type probability distributions for the given question-passage pair, and then activates the answer-generation-modules associated with the *top-D* answer modes; also, the answer type probability distribution will be sent to each answer generation module for reference. Please note that one answer mode can activate several corresponding answer generation modules simultaneously if the ensemble approach is adopted; also, all those activated answer generation modules will be operated in parallel. In the current implementation, the *Dispatcher-layer* is a BERT-based classification model.

[5] Please note that the corresponding information from all answer generation modules will be input to fix the input format (i.e., regardless of whether they are activated by the Dispatch module or not; however, for those inactivated modules, their associated fields will be set to null/zero).

The *Answer-Generation-layer* includes various answer generation modules and generates the local/module output (i.e., the answer-candidate) from each selected answer generation module. Furthermore, each module is expected to generate *top-N* answer-candidates with their associated confidence scores (Details are given in Section 2.3).

The *Aggregation-layer* generates the desired final answer via aggregating various local/module answer-candidates (Section 2.4). Please note that an answer mode may be handled by several different answer generation modules at the same time, if an ensemble approach is adopted. The influence of each answer generation module is implicitly decided by its associated NN weights of a feedforward neural network adopted in this layer.

The *External-Resources* and their accessing utilities/tools provide additional information (to supplement the training data-set and those on-line retrieved documents) to increase the knowledge coverage of the test data. Currently, they include WordNet, ConceptNet, Wikipedia, and other available resources/tools (e.g., Stanford CoreNLP).

Last, the *Online-Working-Memory* is a working-memory used to save the intermediate/linguistic-analysis results (e.g., Hierarchy Embeddings about the question/related-passages, POS/NE annotation, dependency-tree, etc.) that can be shared among various layers/modules later.

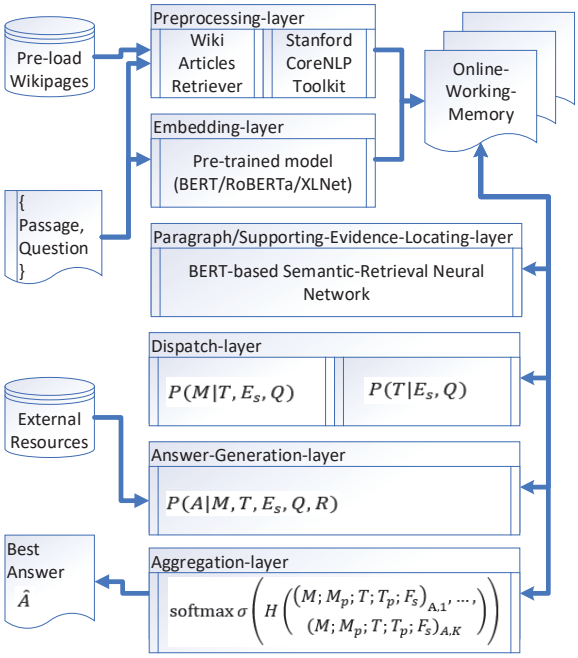

Figure 1. The proposed DNN system architecture

2.3 The Adopted Answer Generation Modules

Figure 2 shows the answer generation modules adopted in this work. Since this paper mainly addresses the framework design, we will only briefly sketch the adopted implementation of each module. The *Single-Span-Extraction* module adopts an ensemble approach. It is implemented by choosing 12 best RoBERTa-large models with AdaBoost algorithm (Yang et al., 2018). The implementation of the *Multi-Span-Extraction* module is based on the tag-based multi-span extraction model (Segal et al., 2020), which treats the task as a sequence tagging problem (i.e., for each token in the passage, decide whether it is part of the answer span). Since the implementations of the *Arithmetic-Operation* and *Date-Duration* modules are similar, we merge these two functionalities into one module in this task. In this merged module, a RoBERTa-base model is first used to extract top K candidates, and then a rule-based procedure is adopted for performing some arithmetic operations such as calculating the duration from the beginning and ending dates.

Furthermore, the *Entailment-Judgement* module is implemented by using a pre-trained BERT mode and fine-tuning it for the *Yes-No* task (Devlin et al., 2019). The *Common-Sense-Inference* is implemented with a template-based approach to answer *Kinship* questions. Firstly, the given question is tokenized by Stanford CoreNLP toolkit. The Chinese kinship associated terms (e.g., father, son, etc.) collected from related Wikipages are added to the dictionary of that toolkit to increase its accuracy rate. Afterwards, a rule-based procedure tries to fill in the slots of the question template with appropriate tokens. Last, the *Summarization* module is implemented by modifying an existing BERT-based extractive summarization algorithm (Liu, 2019)

Please note that some of the answer generation modules are not implemented here, which include the *Compare-Members* module and the *Speaker-View* modules, since they do not occur in the FGC-2020-pre dataset. Also, the *Aggregative-Operation* module is merged into *Multi-Span-Extraction* module, since there are only few questions in this dataset (and the Aggregative-Operation could be subsequently taken on the members that are extracted from the Multi-Span-Extraction module).

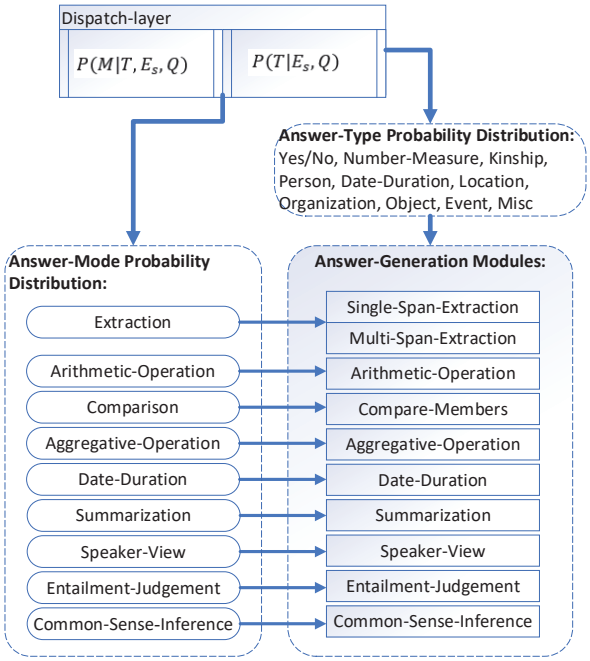

Figure 2. The adopted answer generation modules.

2.4 The Proposed Aggregation Module

As described in section 2.1, this module will adopt a pre-processor to first merge answer candidates from various answer generation modules. Figure 3 shows an example of the merging process. Suppose we have three answer generation modules (i.e., M_1, M_2, M_3) and pick top-3 answer candidates from each answer generation module, where C_{ij} denotes the *rank-j* answer candidate in answer generation *module-i*. After the merging process, there are four *merged answer-candidates* (i.e., MC_1, MC_2, MC_3, MC_4) left. For example, MC_1 groups two answer candidates C_{11} and C_{33} as they are identical.

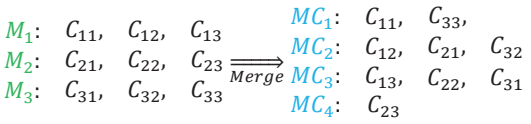

Figure 3. An example of merging answer candidates from different answer generation modules.

The mapping function H is implemented by a Feed-Forward network and its output is connected to a binary classifier (*T/F*) as showed in Figure 4. The *overall-confidence-score* of each merged answer candidate is given by the score of the output T. Take MC_1 as an example, we will have two quintuplets input from *module-1* and *module-3* while other modules are with zero vectors.

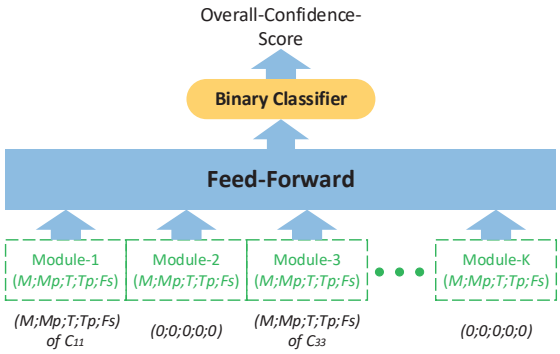

Figure 4. The NN-based aggregation module.

3 Evaluation

To verify the validity and effectiveness of the proposed framework, we have tested it on the FGC-2020 dataset. The details of the dataset and various experiments conducted are presented below.

3.1 Dataset

Officially, FGC-2020 organizer had released both FGC-2020-pre dataset, which is mainly used to let each team train their own model, and FGC-2020-final test set, which is mainly used to evaluate the final round performance. Since the FGC-2020-final test set is not open to various teams before the final contest, the following description is mainly for the FGC-2020-pre dataset. Each released question in the FGC-2020-pre dataset is associated with an official category tag among *Elementary*, *Advanced*, and *Argumentation*[6]. Table 1 shows the statistics of those question categories. Also, as those Argumentation questions do not have the golden answers provided by the FGC organizer, we exclude them from the FGC-2020-pre dataset.

Question Category	Count	Percentage
Elementary	929	70.27%
Advanced	378	28.59%
Argumentation	15	1.14%
Total	1,322	100.00%

Table 1. The statistics of the question categories in the FGC-2020-pre dataset.

To train the models and get a sense about our performance before the final competition, we further divide the remaining FGC-2020-pre data into our own training/development/test three subsets. To avoid distribution mismatch problem, we keep

[6] https://fgc.stpi.narl.org.tw/activity/2020_Talk2AI

the distributions of question categories in each subset as similar as possible while dividing them. The statistics of each subset are shown in Table 2.

Dataset	Count	Percentage
Training	875	66.94%
Development	242	18.52%
Test	190	14.54%
Total	1,307	100.00%

Table 2. The statistics of training/development/test subsets in the FGC-2020-pre dataset.

Figure 5 shows the distributions of answer mode and answer type in the training/development/test subsets, where the vertical axis displays various answer modes/types and the horizontal axis indicates their corresponding percentages. It is observed that the distributions of answer mode in training/development/test subsets are similar but that of answer type are significantly different (especially in the test subset); it is due to that we divide the dataset based on the given documents (and then adjust them according to answer modes), but each document is associated with a varying number of questions/types.

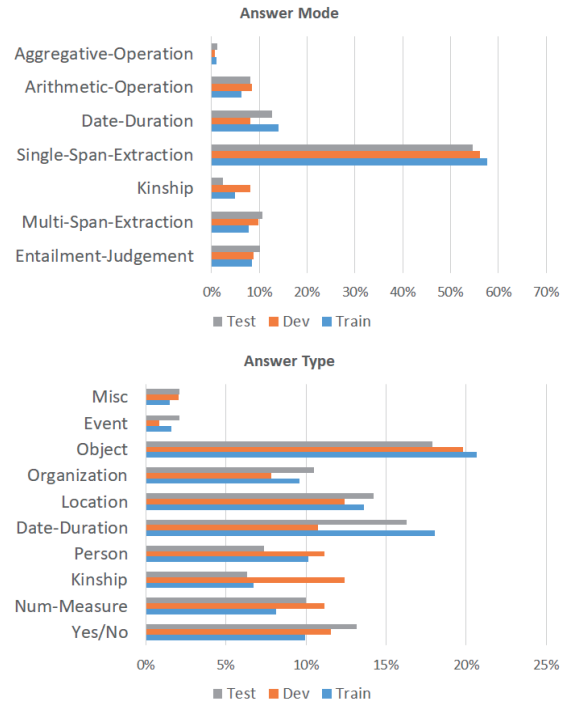

Figure5. The distributions of answer mode and answer type in the training/development/test subsets of the FGC-2020-pre dataset.

3.2 The Baseline Adopted

Since RoBERTa (Liu et al., 2019) is the state-of-the-art pre-trained model for single-span extraction (if ensemble approaches are excluded) on both SQuAD (Rajpurkar et al., 2016) and DRCD (Shao et al., 2018) datasets when we were preparing for the FGC preliminary round (2019/12/24), it was chosen as our baseline model.

3.3 Overall System Performance on Official Pre-released Dataset

Table 3 gives the performances of our proposed model and the above baseline (RoBERTa-large) on both the FGC-2020-pre test-set and the FGC-2020-final test-set. In comparison with the baseline, we have enjoyed 11.4% (= 70.5% - 59.1%) overall improvement on the FGC-2020-pre test-set. This shows when the dataset contains the questions with various answer modes, customizing the model architecture for each specific answer mode (which needs a different inference mechanism) is better than adopting a monolithic architecture (and then applying it to various answer modes). The advantage of adopting the proposed Divide-and-Conquer framework is thus shown.

Furthermore, the top-1 and top-2 accuracy rates of the answer mode are 98.9% and 100.0%, respectively; and those of the answer type are 93.7% and 95.3%, respectively. This shows that the Dispatch-layer is quite promising. The performance of answer type prediction is inferior to that of answer mode, as we have more answer types than answer modes.

Dataset	Baseline	Proposed
FGC-2020-pre test-set	59.1%	70.5%
FGC-2020-final test-set	36.9%	39.1%

Table 3. The EM (Exact Match) scores of the baseline and the proposed model on the FGC-2020-pre and the FGC-2020-final test-sets.

Last, an intuitive approach to implement the Aggregation-layer is to simply pick up the answer candidate with the highest score (which is calculated by multiplying its associated confidence score and the corresponding answer mode probability) among various candidates. It is surprised to find that this intuitive approach (with EM 70.5%) is 0.6% better than our proposed NN-based approach (with EM 69.9%) in this test-set. A possible reason could be that there is almost no overlapping among various top-3 candidate-sets (obtained from different answer generation modules) in this dataset; as the result, the advantage of merging the

same answer-candidate generated from different inference mechanisms thus disappears.

3.4 The Performance on Official Final Test-set

Since we have got FGC-2020-final test-set after the contest, we also show its distributions of answer mode and answer type in Figure 6. It includes total 46 question-passage pairs (again, 4 Argumentation questions are excluded). It is observed that the distributions of both answer mode and answer type in the final run are very different from those in the FGC-2020-pre dataset. This indicates that we have a serious mismatch problem in both answer mode and answer type, which implies that shallow statistical information (which BERT mainly utilizes) would be less useful and deep understanding would be more demanding.

The obtained performance is given in Table 3. In comparison with the baseline, we only got 2.2% (= 39.1% - 36.9%) overall improvement. Comparing with the improvement obtained on the FGC-2020-pre test-set (11.4%), the gap shrinks considerably because the problems in the FGC-2020-final test-set is much more difficult (and thus beyond not only the capability of the baseline but also the capability of our proposed approach).

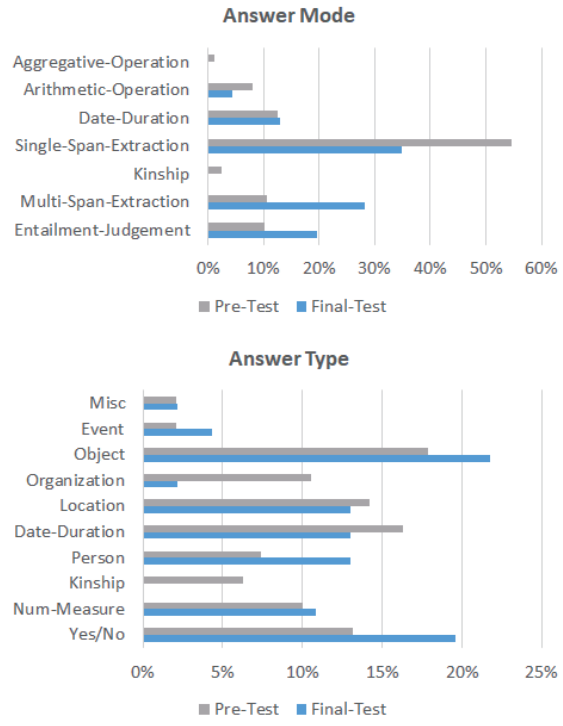

Figure 6. The distributions of answer mode and answer type in the FGC-2020-pre and FGC-2020-final test-sets.

Figure 7 further shows the overall system performance on the FGC-2020-pre and FGC-2020-final test-sets in each category. Surprising in coincidence, the accuracy rates on Elementary, Advanced, and Overall categories are 0.391, 0.391, and 0.391, respectively. In comparison with the overall performance of the FGC-2020-pre test-set, the accuracy rate drops 0.314 (from 0.705 to 0.391). Figure 8 additionally shows the accuracy rates associated with various answer-modes (Please note that there is no Kinship answer mode question in this test-set). We even have 0% and 15.4% accuracy rates for the Arithmetic-Operation and Multi-Span-Extraction answer modes, respectively. The obtained poor performances clearly indicate that these two answer-modes are more difficult to handle, which fits our intuition.

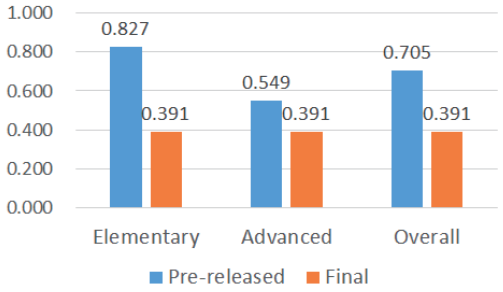

Figure 7. The overall system accuracy rate on the FGC-2020-pre and FGC-2020-final test-sets.

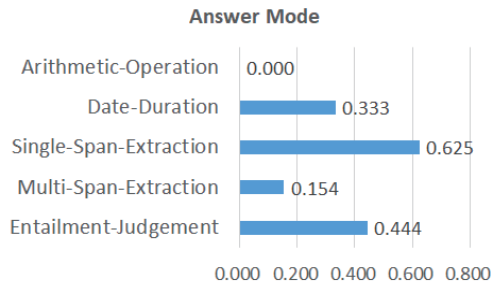

Figure 8. The accuracy rates associated with various answer modes on the FGC-2020-final test-set.

4 Error Analysis and Discussion for Official Final Test-set

As Figure 7 shows, the overall system performance degrades significantly (down 0.314, from 0.705 to 0.391) when we move from FGC-2020-pre test-set to FGC-2020-final test-set. It is mainly because the questions in the FGC-2020-final test-set is generally more difficult than that in the FGC-2020-pre test-set. And it is also because the involved topics (also their associated lexicons), the distributions of

both answer mode and answer type drift significantly from FGC-2020-pre test-set to FGC-2020-final test-set (as shown in Figure 6).

Since almost all our current answer generation modules adopt BERT-based approaches, and it is well-known that BERT conducts the inference mainly based on surface-clues/hidden-distribution-bias (Naik et al., 2018; Poliak et al., 2018; Jiang and Marneffe, 2019; McCoy et al., 2019), the mismatch of those surface-clues/distributions thus causes serious degradation. On the other hand, it also implies that BERT-based approaches, although they have become state-of-the-art models, are still not capable to handle the FGC-2020 kind of tests (which require deep reasoning and cannot be falsely solved simply with surface-clues/distribution-bias).

Specifically, the performance of the Elementary questions drops more (down 0.436, from 0.827 to 0.391) in comparison with that of Advanced ones (down 0.158, from 0.549 to 0.391). The performance of the Advanced questions is less affected because those questions require deeper reasoning, and is thus less affected by the drift of topics and the distribution of answer mode/answer type mentioned above.

If we zoom into various answer modes, it is observed that the Multi-Span-Extraction causes most overall degradation in the FGC-2020-final test-set, which is mainly due to both its low accuracy rate (15.4% in Figure 8) and its high answer mode portion (28% in Figure 6)). It seems that the tag-based approach (Section 2.3) is not capable of handling the Multi-Span-Extraction questions involved in this dataset, as getting a multi-span answer needs to locate various list-members via matching the structures (Gentner and Markman, 1997) of the question and the passage, not just regarding it as a sequence-tagging task.

5 Conclusion

We proposed a divide-and-conquer model/framework for answering the questions in FGC-2020 QA dataset, which covers various answer modes. With the proposed Dispatch-layer, the proposed framework is flexible for handling various answer modes with different modules simultaneously, and is extensible for adding new answer modes and answer types in the future. Also, with the proposed Aggregation-layer, the proposed framework can take advantage of different inference mechanisms, and also reduce the error accumulation problem. Last, due to its design for fitting the end-to-end multi-task learning framework, the proposed framework could be implemented with an appropriate neural network and is thus more suitable for end-to-end optimization without much effort.

We have tested the proposed framework on 2020 Formosa Grand Challenge Contest QA dataset. The experiment results show that our system outperforms the baseline RoBERTa-large model about 11.4% on the FGC-2020-pre test-set. However, the overall system performance drops significantly (about 31.4%) from the FGC-2020-pre test-set to the FGC-2020-final test-set. On the other hand, together with our another dialog sub-system (tested on the FGC-2020-final *Dialog* test-set), we obtained 44.1 total score (out of 100; the human performance is 68.2), which outperforms that of the official top one system (announced in this contest) 7.4 points.

References

Daniel Andor, Luheng He, Kenton Lee, and Emily Pitler. 2019. *Giving BERT a Calculator: Finding Operations and Arguments with Reading Comprehension*. arXiv:1909.00109v2.

Dan-Qi Chen. 2018. *Neural Reading Comprehension and Beyond.* Ph.D. Dissertation. Stanford University.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. arXiv:1810.04805v2.

David Ferrucci. 2012. *Introduction to 'This is Watson'*. IBM J. RES. & DEV. VOL. 56 NO. 3/4 PAPER 1.

Dedre Gentner and Arthur B. Markman. 1997. *Structure mapping in analogy and similarity.* American Psychologist, 52(1):45–56.

Ming-Hao Hu, Yu-Xing Peng, Zhen Huang, and Dongsheng Li. 2019. *A Multi-Type Multi-Span Net-work for Reading Comprehension that Requires Discrete Reasoning*. arXiv:1908.05514v2.

Chu-Ren Huang, Shu-Kai Hsieh, Jia-Fei Hong, Yun-Zhu Chen, I-Li Su, Yong-Xiang Chen, and Sheng-Wei Huang. 2008. Chinese Wordnet: Design, Implementation, & Application of an Infrastructure for Cross-lingual Knowledge Processing. In *Chinese Lexical Semantic Workshop 2008*.

Nan-Jiang Jiang and Marie-Catherine de Marneffe (2019). Evaluating BERT for natural language inference: A case study on the Commitment Bank. In *Proceedings of the 2019 Conference on Empirical*

Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing. Association for Computational Linguistics, pages 6086–6091.

Cheng-Ru Li, Chi-Hsin Yu, and Hsin-Hsi Chen. 2011. Predicting the Semantic Orientation of Terms in E-HowNet. In *Proceedings of the 23rd Conference on Computational Linguistics and Speech Processing*, pages 151–165.

Yang Liu. 2019. *Fine-tune BERT for Extractive Summarization*. arXiv:1903.10318v2.

Yin-Han Liu, Myle Ott, Naman Goyal, Jing-Fei Du, Mandar Joshi, Dan-qi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. *RoBERTa: A Robustly Optimized BERT Pretraining Approach*. arXiv:1907.11692v1.

Tom McCoy, Ellie Pavlick, and Tal Linzen. 2019. Right for the Wrong Reasons: Diagnosing Syntactic Heuristics in Natural Language Inference. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics. Association for Computational Linguistics*, pages 3428–3448.

Aakanksha Naik, Abhilasha Ravichander, Norman Sadeh, Carolyn Rose, and Graham Neubig. 2018. Stress Test Evaluation for Natural Language Inference. In *Proceedings of the 27th International Conference on Computational Linguistics. Association for Computational Linguistics*, pages 2340–2353.

Yi-Xin Nie, Song-He Wang, and Mohit Bansal. 2019. *Revealing the importance of semantic retrieval for machine reading at scale*. arXiv:1909.08041v1.

Adam Poliak, Jason Naradowsky, Aparajita Haldar, Rachel Rudinger, and Benjamin Van Durme. 2018. Hypothesis only baselines in natural language inference. In *Proceedings of the Seventh Joint Conference on Lexical and Computational Semantics. Association for Computational Linguistics*, pages 180–191.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. *SQuAD: 100,000+ Questions for Machine Comprehension of Text*. arXiv: 1606.05250v3.

Sebastian Ruder. 2017. *An Overview of Multi-Task Learning in Deep Neural Networks*. arXiv:1706.05098v1.

Elad Segal, Avia Efrat, Mor Shoham, Amir Globerson, and Jonathan Berant. 2020. *A Simple and Effective Model for Answering Multi-span Questions*. arXiv:1909.13375v1.

Chih-Chieh Shao, Trois Liu, Yu-Ting Lai, Yi-Ying Tseng, and Sam Tsai. 2018. *DRCD: a Chinese Machine Reading Comprehension Dataset*. arXiv:1806.00920.

Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. 2020. *Megatron-LM: Training Multi-Billion Parameter Language Models Using Model Parallelism.* arXiv:1909.08053v4.

Robyn Speer, Joshua Chin, and Catherine Havasi. 2018. *ConceptNet 5.5: An Open Multilingual Graph of General Knowledge*. arXiv:1612.03975v2.

Adam Trischler, Tong Wang, Xing-Di Yuan, Justin Harris, Alessandro Sordoni, Philip Bachman, and Kaheer Suleman. 2016. *NewsQA: A Machine Comprehension Dataset*. arXiv:1611.09830v3.

Dong-Dong Yang, Sen-Zhang Wang, and Zhou-Jun Li. 2018. Ensemble Neural Relation Extraction with Adaptive Boosting. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*, pages 4532–4538.

Zhi-Lin Yang, Zi-Hang Dai, Yi-Ming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019. *XLNet: Generalized Autoregressive Pretraining for Language Understanding*. arXiv:1906.08237v2.

Yu Zhang and Qiang Yang. 2021. *A Survey on Multi-Task Learning*. arXiv:1707.08114v3.

Zhuo-Sheng Zhang, Jun-Jie Yang, and Hai Zhao. 2020. *Retrospective Reader for Machine Reading Comprehension*. arXiv: 2001.09694v3.