Correct after Answer: Enhancing Multi-Span Question Answering with Post-Processing Method

Anonymous ACL submission

Abstract

Multi-Span Question Answering (MSQA) requires models to extract one or multiple answer spans from a given context to answer a question. Prior work mainly focus on designing specific methods or applying heuristic strategies to encourage models to predict more correct predictions. However, these models are trained on gold answers and fail to considier the incorrect predictions. Through a statistical analysis, we observe that models with stronger abilities do not predict less incorrect predictions compared to other models. In this work, we propose **Answering-Classifying-Correcting** (ACC) framework, which employs a post-processing strategy to handle with incorrect predictions. Specifically, the ACC framework first introduces a **classifier** to classify the predictions into three types and exclude "wrong predictions", then introduces a **corrector** to modify "partially correct predictions". Experiments on four datasets show that ACC framework significantly improves the EM F1 scores of several MSQA models, and further analysis demostrate that ACC framework efficiently reduces the number of incorrect predictions, improving the quality of predictions. [1]

Context:
Don't Hug Me I'm Scared (often abbreviated to DHMIS) is a live - action / animated surreal horror comedy web series created by British filmmakers Becky Sloan and Joseph Pelling ...

Question:
Who made Don't Hug Me I'm Scared?

Gold Answers:
Becky Sloan, Joseph Pelling

Predictions:
Joseph Pelling (correct)
filmmakers Becky Sloan (partially correct)
DHMIS (wrong)

Figure 1: An example of multi-span questions, this question has two gold answers: *Becky Sloan* and *Joseph Pelling*. "Joseph Pelling" is a correct prediction, "filmmakers Becky Sloan" is a partially correct prediction and "DHMIS" is a wrong prediction.

1 Introduction

Machine Reading Comprehension (MRC) requires models to answer a question based on a given context (Rajpurkar et al., 2018; Kwiatkowski et al., 2019; Lai et al., 2017). In a real-world scenario, a single question typically corresponds to multiple answers. To this end, Multi-Span Question Answering (MSQA) has been proposed (Ju et al., 2022; Li et al., 2022; Yue et al., 2023). Different from the traditional Single-Span Question Answering (SSQA), the goal of MSQA is to extract one or multiple non-overlapped spans from the given context. For example, In Figure 1, the question *"Who made Don't Hug Me I'm Scared?"* has two answers: *"Becky Sloan"* and *"Joseph Pelling"*.

Recently, a series of methods have been proposed to handle with MSQA. Some of them incorporate heuristic strategies based on traditional pointer models (Vinyals et al., 2015) to extract multiple answers (Yang et al., 2021; Hu et al., 2019); some of them convert MSQA task into a sequence-tagging task and utilize BIO tags to mark answers (Segal et al., 2020; Li et al., 2022); some of them enumerate all candidate answers and select the final answers with a learnable threshold (Huang et al., 2023a; Zhang et al., 2024).

Prior work mainly focus on designing specific methods or applying heuristic strategies to encourage models to predict more correct predictions. However, these models are trained on gold answers, and fail to considier the incorrect predictions. To further investigate the incorrect predictions predicted by these models, we classify the predictions into **correct predictions**, **partially correct pre-**

[1]Our code and data are available at `https://anonymous.4open.science/r/ACC-F6FB`.

dictions and **wrong predictions** based on whether they should be modified or excluded, and conduct a statistical analysis on some MSQA models (details in Section 2.3). We observe that models with stronger abilities (i.e., achieving higher F1 scores) do not predict less incorrect predictions compared to other models. This indicates that the performance of the MSQA models can be improved if the number of incorrect predictions can be reduced.

In this work, we propose **Answer-Classify-Correct** (ACC) framework, which employs a post-processing strategy to handle with incorrect predictions. The ACC framework simulates humans strategy in English examinations: listing candidate answers, reviewing and modifying. Specifically, we design the **classifier** to categorize candidate answers into "correct predictions", "partially correct predictions" or "wrong predictions", then we design the **corrector** to modify "partially correct predictions", finally we exclude "wrong predictions" and obtain final predictions. To train the classifier and the corrector, we also apply an automatic annotation approach which samples incorrect predictions from the training datasets and constructs the silver-labeled datasets.

We conduct experiments on four MSQA datasets. Experiment results show that the ACC framework significantly improves the performance. For instances, after applying the ACC framework, the EM F1 score increases from 60.74% to 67.78% for Tagger-BERT (Li et al., 2022) and from 69.05% to 72.26% for Tagger-RoBERTa (Li et al., 2022) on the MultiSpanQA dataset (Li et al., 2022). Further analysis on the predictions also indicate that the ACC framework effectively reduces the number of incorrect predictions and obtains more correct predictions, enhancing the qualities of predictions. In addition, We also conduct a pilot study with GPT-3.5 [2], demostrating that ACC framework can be applied to Large Language Models (LLMs) in a Chain-of-Thought (CoT) (Wei et al., 2022; Kojima et al., 2022) manner.

Our contributions are summarized as follows:

- We develop a three-fold taxonomy for the MSQA predictions based on whether a prediction should be modified or excluded. Then, we conduct a statistical analysis, revealing distributions over the three categories.
- Inspired by humans' strategies, we propose the ACC framework, which includes a classifier to exclude incorrect predictions and includes a corrector to modify imperfect predictions. We also design an automatical annotation approach to sample incorrect predictions and construct silver-labeled datasets.
- We conduct several experiments on four MSQA datasets. Results show that the ACC framework significantly enhances the quality of the MSQA predictions.

[2] https://platform.openai.com/.

2 Taxonomy of MSQA Predictions

2.1 Formalization

The MSQA task can be described as a triplet (Q, C, A): a question Q, its corresponding context C, and a set of gold answers $A = \{a_1, a_2, ..., a_n\}$, where each answer a_i is a contigious span from C. Existing methods utilize a model M to extract $P = \{p_1, p_2, ..., p_n\}$ from C as the predictions, shown as Eq (1).

$$P = M(C, Q) \tag{1}$$

2.2 Taxonomy

Intuitively, the predictions can be categorized as correct or incorrect predictions. However, some of incorrect predictions should be modified while others should be excluded. For example, assuming that one of gold answers is "a clever boy" and the predictions are "boy" and "girl", both of the predictions are incorrect but "boy" should be modified and "girl" should be excluded. Therefore, we further categorize incorrect predictions into "partially correct predictions" and "wrong predictions".

Based on above analysis, we category the prediction p_i into one of the following three types: **correct prediction**, **partially correct prediction** and **wrong prediction**.

Correct prediction The prediction p_i is one of the gold answers, which means $p_i \in A$.

Partially correct prediction The prediction p_i is not a correct prediction, but there exists a gold answer a_j which is similar to p_i, then p_i is defined as *partially correct prediction* and a_j is defined as its corresponding *similar gold answer*.

Considering that gold answers typically contain complicated grammar structures, we utilize both *Word Overlap* and *Semantic Similarity* to define partially correct predictions. Assuming that a prediction p_i contains k words $\{p_{i1}, p_{i2}, ..., p_{ik}\}$ and a gold answer a_j contains l words $\{a_{j1}, a_{j2}, ..., a_{jl}\}$, we define the word overlap WO and the semantic similarity SS as:

$$WO(p_i, a_j) = \frac{card(p_i \cap a_j)}{max(k, l)} \tag{2}$$

$$SS(p_i, a_j) = \frac{H_{p_i} H_{a_j}^{\top}}{|H_{p_i}||H_{a_j}|} \tag{3}$$

where $card(A)$ denotes the number of element in the set A, H_{p_i} and H_{a_j} are the representations of p_i and a_j from a Pre-trained Language Model, $|a|$ denotes the length of the vector a. [3].

For a prediction p_i, if there exists $a_j \in A$ which satisfies $WO(p_i, a_j) \geq \alpha$ and $SS(p_i, a_j) \geq \beta$, where α and β are hyper-parameters, the p_i is defined as the partially correct prediction.

Wrong prediction : If p_i could not satisfy the conditions of correct prediction and partially correct prediction, we define p_i as wrong prediction.

Figure 1 shows an example containing these three types of predictions. The gold answers are "Becky Sloan" and "Joseph Pelling". For the predictions, "Joseph Pelling" is a correct prediction; "filmmakers Becky Sloan" is a partially correct prediction because it is similar to "Becky Sloan", and "DHMIS" is a wrong prediction because it is not similar to any gold answer.

2.3 Analysis of MSQA Predictions

Based on our designed taxonomy, we conduct a statistical analysis on the dev set of MultiSpanQA (Li et al., 2022). We select four MSQA model: MTMSN (Hu et al., 2019), MUSST(Yang et al., 2021), Tagger (Li et al., 2022) and SpanQualifier (Huang et al., 2023a). We utilize BERT (Devlin et al., 2019) as the encoder. More details of these models are shown in Appendix A.2.

The statistical results are shown in Figure 2. Compared with MTMSN and MUSST, Tagger and SpanQualifier predict more correct predictions but also predict equal or more incorrect predictions. For example, Tagger predicts 1,212 correct predictions but also predict 748 wrong predictions, while MTMSN predicts 742 correct predictions and 459 wrong predictions. We also observe that Tagger and SpanQualifier outperform MTMSN and MUSST on several MSQA benchmarks. This indicates that the improvements of the existing MSQA models

[3] In practice, we utilize *BERTScore* (Zhang et al., 2020) to calculate semantic similarity.

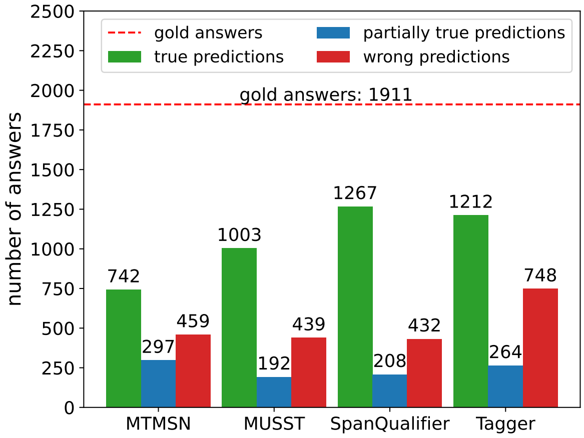

Figure 2: The prediction distributions of correct predictions, partially correct predictions and wrong predictions on the dev set of MultiSpanQA.

are derived from predicting more correct predictions rather than less incorrect predictions. Therefore, we believe that the post-processing method can effectively enhance the quality of predictions by reducing the number of incorrect predictions, resulting in better performance.

3 Method

In this section, we describe the ACC framework, which is designed to handle with partially correct predictions and wrong predictions. The architecture of the ACC framework is shown in Figure 3.

Similar to the humans' strategies, the post-processing procedure of the ACC framework consists of three steps: The first step is **answering**, where we employ a **reader** to obtain initial predictions P; The second step is **classifying**, where we employ a **classifier** to categorize each prediction p_i into one of the three classes: correct prediction, partially correct prediction and wrong prediction; The last step is **correcting**, where we employ a **corrector** to modify the partially correct predictions. We reserve correct predictions predicted by the classifier and the modified predictions from the corrector as the final predictions.

Next, we will provide more details of the reader, the classifier and the corrector. We will also introduce an automatic annotation approach which samples incorrect predictions and constructs training data for the classifier and the corrector.

3.1 Reader

The main function of the reader is to extract several text spans from context based on a given question. This process can be described as:

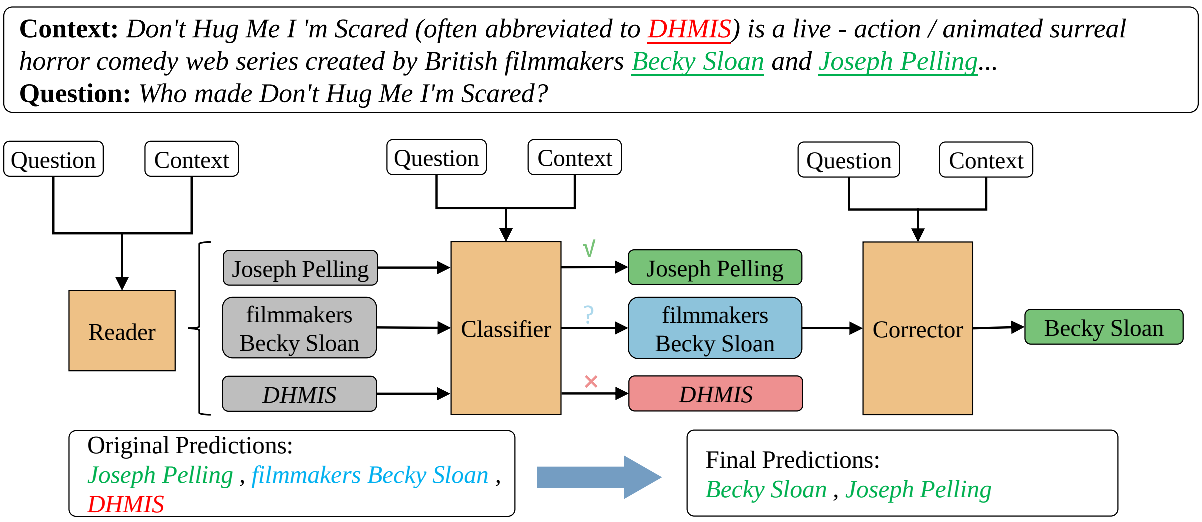

Figure 3: The overall architecture of our proposed ACC framework.

$$P = Reader(Q, C) \qquad (4)$$

where $P = \{p_1, p_2, ..., p_n\}$ are the predictions given by the reader, Q is the question and C is the corresponding context.

3.2 Classifier

The predictions of the reader may include partially correct predictions or wrong predictions (mentioned in 2.2). To this end, we design the classifier to classify them and exclude wrong predictions. Given the candidate predictions P, the classifier splits them into correct predictions P_c, partially correct predictions P_p and wrong predictions P_w. This process can be described as:

$$P_c, P_p, P_w = Classifier(P, Q, C) \qquad (5)$$

where P_c, P_p and P_w denote the correct predictions, partially predictions and wrong predictions predicted by the classifier, respectively.

Specifically, the classifier consists of a transformer (Vaswani et al., 2017) encoder and a classification head. The classification head includes an MLP layer to obtain logits of each class. Inspired by Zhu et al. (2022), we also add a cross-attention layer in the classification head which calculates the attention scores between the question and the context to enhance the representations of them.

3.3 Corrector

The classifier is able to exclude wrong predictions, however, there may still contain partially correct predictions which are imperfect and should be modified. Hence, we design the corrector to modify those partially correct predictions. This process can be described as:

$$\hat{P}_p = Corrector(P_p, Q, C) \qquad (6)$$

where P_p are the partially correct predictions given by the classifier and $\hat{P}_p$ are the predictions modified by the corrector.

We adpot traditional pointer model (Vinyals et al., 2015) to predict the start and end probabilities, st and ed. During the inference stage, for the text span starting at i-th token and ending at j-th token, we calculate its confidence score $score_{ij} = st_i + ed_j$ and obtain the best index pair (i, j) which maximizes $score_{ij}$, then extract its corresponding span as the modified prediction.

The final outputs of the ACC framework $\hat{P}$ consist of the correct predictions P_c predicted by the classifier and the modified predictions $\hat{P}_p$ from the corrector, described as:

$$\hat{P} = P_c \cup \hat{P}_p \qquad (7)$$

3.4 Data Annotations

To train the classifier and the corrector, we need both correct predictions and incorrect predictions. However, most MSQA datasets do not contain incorrect predictions. Inspired by Gangi Reddy et al. (2020), we adopt an automatical sampling method similar to K-fold cross-validation, to collect incorrect predictions from the MSQA datasets and construct our silver-labeled datasets.

	MultiSpanQA			MultiSpanQA-Expand			MAMRC			MAMRC-Multi		
	EM P	EM R	EM F1	EM P	EM R	EM F1	EM P	EM R	EM F1	EM P	EM R	EM F1
BERT-base												
MTMSN	51.76	41.69	46.18	60.88	51.46	55.78	72.65	77.41	74.96	71.50	76.71	74.01
+ACC	**67.75**	**49.52**	**57.22**	**67.77**	**54.91**	**60.66**	**81.60**	**77.40**	**79.44**	**85.55**	**79.32**	**82.32**
MUSST	61.44	53.74	57.33	67.48	59.71	63.36	76.28	79.00	77.62	75.68	78.12	76.88
+ACC	**68.84**	**54.39**	**60.76**	**69.62**	**60.05**	**64.48**	**81.94**	77.10	**79.45**	**85.87**	**78.38**	**81.95**
Tagger	56.66	65.46	60.74	52.81	55.92	54.30	77.15	81.83	79.42	74.71	76.74	75.70
+ACC	**68.52**	**67.05**	**67.78**	**62.74**	**58.83**	**60.71**	**82.56**	79.67	**81.10**	**85.80**	**77.58**	**81.48**
SpanQualifier	67.99	69.44	68.70	62.83	67.88	65.25	77.51	84.51	80.86	76.10	85.39	80.47
+ACC	**72.04**	67.82	**69.86**	**65.78**	67.13	**66.45**	**82.40**	80.76	**81.57**	**85.67**	83.37	**84.51**
RoBERTa-base												
MTMSN	59.86	49.97	54.47	63.39	56.00	59.47	73.94	78.36	76.08	71.69	77.47	74.46
+ACC	**71.75**	**55.87**	**62.82**	**68.95**	**58.81**	**63.48**	**81.84**	**77.70**	**79.72**	**85.13**	**79.82**	**82.39**
MUSST	69.82	61.94	65.64	69.29	63.16	66.08	78.01	79.71	78.85	76.69	77.16	76.92
+ACC	**73.07**	61.78	**66.96**	**70.54**	62.60	**66.33**	**82.75**	77.57	**80.08**	**86.10**	**77.48**	**81.56**
Tagger	66.22	72.14	69.05	64.35	65.66	64.99	79.47	83.59	81.48	75.85	78.19	77.00
+ACC	**72.39**	72.12	**72.26**	**68.70**	**66.21**	**67.43**	**83.62**	81.80	**82.70**	**85.77**	**78.36**	**81.90**
SpanQualifier	70.40	72.82	71.58	64.65	69.65	66.99	83.40	80.83	82.10	75.63	85.77	80.37
+ACC	**73.69**	71.32	**72.47**	**67.68**	68.53	**68.09**	**82.83**	**81.88**	**82.35**	**85.14**	83.77	**84.45**

Table 1: EM Scores on four MSQA datasets. "P" "R" "F1" refer to Precision, Recall and F1 score. "BERT-base" and "RoBERTa-base" refer to the encoders of the MSQA models. The results marked in **bold** means improvements after applying the ACC framework.

First, we randomly divide the training data D into K equal subsets: $D_1, D_2, ..., D_K$. We perform K iterations, in the i-th iteration we initialize an reader M and train it with all training data except D_i, then sampling the predictions of D_i with M. After K iterations, we utilize the gold answers from training dataset D to annotate all predictions, and construct the silver-labeled dataset. More details are shown in Appendix A.3.

4 Experiments

4.1 Experimental Setup

Datasets We evaluate the ACC framework on four datasets: **MultiSpanQA** (Li et al., 2022), **MultiSpanQA-Expand** (Li et al., 2022), **MAMRC** (Yue et al., 2023) and **MAMRC-Multi**. Details are shown in Appendix A.1.

MSQA models we set four MSQA models as the reader in the ACC framework: MTMSN (Hu et al., 2019), MUSST (Yang et al., 2021), Tagger (Li et al., 2022) and SpanQualifier (Huang et al., 2023a). Details are shown in Appendix A.2.

Evaluation Metrics We use **Exact Match Precision/Recall/F1 (EM P/R/F1)** (Li et al., 2022) as the main metrics in our experiments. EM assign a score of 1 when a prediction fully matches one of the gold answers and 0 otherwise.

Implementation Details For the classifier and corrector in the ACC framework, we use RoBERTa-base (Zhuang et al., 2021) as encoder. For MSQA models, we use both BERT-base (Devlin et al., 2019) and RoBERTa-base as encoder. For the hyper parameters mentioned in Section 2.2, we set $\alpha = 0.25$ and $\beta = 0.6$. See more training and inference details in Appendix A.3.

4.2 Main Results

Table 1 shows the main results on four MSQA datasets. After applying the ACC framework, all MSQA models gain improvements in EM F1 scores. For instances, the EM F1 score of Tagger (BERT-base) increases from 60.74% to 67.78%, and the EM F1 score of Tagger (RoBERTa-base) increases from 69.05% to 72.26% on the dev set of MultiSpanQA. We observe that presicion scores show significant improvements while some recall scores show slight declines, demonstrating that ACC framework may exclude incorrect predictions effectively but also exclude a small number of correct predictions. Ultimately, due to the greater degree of improvement in precision scores, the F1 scores are increased. In Section 5.2, we will investigate the performance of the classifier and the corrector, and analyze why the ACC framework improves the EM F1 scores.

We also evaluate the ACC framework with Partial Match P/R/F1 (PM P/R/F1), which considers

	MultiSpanQA		
	EM P	EM R	EM F1
Tagger BERT	56.66	65.46	60.74
+ cls only	64.90	63.98	64.44
+ cor only	62.49	**69.11**	65.63
+ cor & cls	67.14	67.44	67.29
+ binary cls & cor	68.58	66.56	67.56
+ cls & cor	**68.52**	67.05	**67.78**
Tagger RoBERTa	66.22	72.14	69.05
+ cls only	70.54	70.58	70.56
+ cor only	68.50	**73.09**	70.72
+ cor & cls	71.21	71.43	71.32
+ binary cls & cor	72.45	70.94	71.68
+ cls & cor	**72.39**	72.12	**72.26**

Table 2: Ablation study of ACC framework on the dev set of MultiSpanQA. The best performance is in **bold**.

	MultiSpanQA		
	EM P	EM R	EM F1
Tagger BERT	56.66	65.46	60.74
+ att cls & T5 cor	64.90	63.98	64.44
+ vanilla cls & ext cor	**68.54**	66.10	67.29
+ att cls & ext cor	68.52	**67.05**	**67.78**
Tagger RoBERTa	66.22	72.14	69.05
+ att cls & T5 cor	70.54	70.58	70.56
+ vanilla cls & ext cor	72.23	71.56	71.89
+ att cls & ext cor	**72.39**	**72.12**	**72.26**

Table 3: Comparison between diffent combinations of the classifier and the corrector on the dev set of MultiSpanQA. "att cls" refer to the classifier mentioned in Section 3.2, "vanilla cls" refer to the classifier without cross-attention layer, "ext cor" refer to the corrector mentioned in Section 3.3 and "T5 cor" refer to the T5 corrector. The best performance is in **bold**.

the overlap between the predictions and gold answers. Results are shown in Appendix B.1.

5 Discussions

5.1 Ablation Study

Roles of classifier and corrector. ACC framework uses the "answer-classify-correct" procedure with the classifier and the corrector. To investigate whether there exists better post-processing procedure, we conduct an ablation study by: 1. only employing the classifier or corrector (cls \ cor only); 2. changing the order of classifier and corrector (cor & cls); 3. modifying both correct predictions and partially correct predictions (binary cls & cor).[4]

Table 2 shows the results of the ablation study on the dev set of MultiSpanQA. The performance of "cls only" and "cor only" lag behind ACC framework, demonstrating the significance of the classifier and corrector. Changing the order between classifier and corrector also shows decline, the reason may be that using corrector first may lead to conceal wrong predictions, thereby the classifier may fail to categorize them as wrong predictions. We also observe that modifying both correct predictions and partially correct predictions does not achieve improvements, demostrating the necessity of distinguishing correct predictions and partially correct predictions and modifying partially correct predictions solely.

Comparison with different models. ACC framework uses a classifier with a cross-attention layer and a corrector based on the pointer model. However, ACC framework can also opt for alternative type of classifiers or correctors. To this end, we replace the classifier and the corrector with other models and compare their performance.[5]

Table 3 shows the results of the comparison between different model combinations on the dev set of MultiSpanQA. After replacing the classifier or the corrector, ACC framework shows declines, especially when applying a generative model, ACC framework lag behind other settings. This indicates that the generative models are less capable than traditional pointer models in correcting predictions.

5.2 Analysis on the Predictions

Accuracy of the classifier. To analyze the capability of the classifier, we conduct a statistical analysis on its classification results. Table 5 shows the accuracy of the classifier on the dev set of MultiSpanQA. The classifier achieves an high accuracy on the correct predictions (95.82% for Tagger-BERT and 95.45% for Tagger-RoBERTa), demonstrating that the ACC framework reserves most correct predictions. On the other hand, the classifier exclude about 1/3 wrong predictions, contributing to the imporvements on EM F1 scores, while the accuracies on the partially true predictions and the wrong predictions can be further improved.

Changes in answers by the corrector. To analyze the capability of the corrector, we also conduct a statistical analysis on how many prediction has been changed. Table 6 shows the changes of the partially correct predictions on the dev set of

[4]For "cls only", we only exclude wrong predictions; For "cor only", we correct all predictions; For "cor & cls", we first correct all predictions, then classify them and only exclude wrong predictions.

[5]for the classifier, we replace it with a vanilla classifier where we remove the cross-attention layer; for the corrector, we replace it with T5 (Raffel et al., 2020) which outputs texts as the corrected answers.

Context: The California State Legislature is the state legislature of the U.S. state of California. It is a bicameral body consisting of the lower house, the California State Assembly, with 80 members, and the upper house, the California State Senate, with 40 members...
Question: What are the two chambers of the California state legislation ?
Gold Answers: California State Assembly , California State Senate

	Original Predictions	**New Predictions**
MTMSN (RoBERTa-base)	the California State Assembly , the California State Senate	**California State Assembly** , **California State Senate**
MUSST (RoBERTa-base)	California State , Senate , lower house , the California State Assembly	**California State Assembly** , **California State Senate**
Tagger (RoBERTa-base)	Assembly , **California State Senate** , Senate , State Assembly	**California State Assembly** , **California State Senate**
SpanQualifier (RoBERTa-base)	Assembly Senate	Assembly , **California State Senate**

Table 4: Case study on the dev set of MultiSpanQA. "Original Predictions" refers to the predictions of the MSQA models and "New Predictions" refers to the predictions after applying the ACC framework. Correct predictions are in **bold**.

Tagger BERT			
label \ pred	wrong	partially	correct
wrong	**268 (37.85%)**	148 (20.9%)	292 (41.24%)
partially	16 (6.13%)	**98 (37.55%)**	147 (56.32%)
correct	26 (2.18%)	24 (2.01%)	**1145 (95.82%)**
Tagger RoBERTa			
label \ pred	wrong	partially	correct
wrong	**135 (27.44%)**	105 (21.34%)	252 (51.22%)
partially	22 (8.63%)	**83 (32.55%)**	150 (58.82%)
correct	27 (2.01%)	34 (2.54%)	**1280 (95.45%)**

Table 5: Accuracy of the classifier on the dev set of MultiSpanQA. The correct classifications of each types are in **bold**.

Tagger BERT		
cls \ cls & cor	not correct	correct
not correct	172 (63.00%)	**75 (27.47%)**
correct	**9 (3.3%)**	17 (6.23%)
Tagger RoBERTa		
cls \ cls & cor	not correct	correct
not correct	137 (61.43%)	**52 (23.32%)**
correct	**11 (4.93%)**	23 (10.31%)

Table 6: Changes in answers by the corrector on the dev set of MultiSpanQA.

MultiSpanQA. The corrector changes 30.77% of the answers for Tagger-BERT and 27% for Tagger-RoBERTa, respectively. For Tagger-BERT, 27.47% of the not-correct predictions are modified to the correct predictions, while 3.3% of the correct predictions are modified to the not-correct predictions. Furthermore, among all the partially correct predictions derived from the classifier, over 60% of the not-correct predictions remain unchanged, indicatisng a significant room for improvements.

5.3 Case Study

Table 4 illustrates the original predictions from MSQA models and the new predictions after applying the ACC framework. All the four MSQA models fail to provide precise predictions, but after applying ACC framework, the predictions of MTMSN, MUSST and Tagger are completely consistent with the golds answers (i.e. $EMF1 = 100\%, PMF1 = 100\%$), indicating that the ACC framework is able to provide better predictions.

5.4 Pilot Study with LLM

ACC framework utilizes a fine-tuned RoBERTa encoder as the backbone. To investigate whether our proposed method works on larger models, we conduct a pilot study by replacing the classifier or corrector with a prompted LLM. The implementation details and prompts are shown in Appendix B.3.

Table 7 shows the experiment results. After replacing the classifier or the corrector with LLM, the ACC framework still achieves improvements on Tagger-BERT and Tagger-RoBERTa, which proves that our post-processing strategies can be effectively applied to LLM.

5.5 Model Size and Inference Time

We analyze the model size and the inference time of the ACC framework. Results and analysis are shown in Appendix B.2.

6 Related Work

6.1 Multi-Span Question Answering

Recently, a series of MSQA benchmarks (Ju et al., 2022; Li et al., 2022; Yue et al., 2023) have been proposed to faclitate research on QA tasks that are closer to real-world scenarios. MSQA tasks

	MultiSpanQA		
	EM P	EM R	EM F1
Tagger BERT	56.66	65.46	60.74
+LLM cls & LLM cor	68.60	63.35	65.87
+LLM cls & FT cor	**70.04**	64.47	67.14
+FT cls & LLM cor	67.93	66.51	67.21
+FT cls & FT cor	68.52	**67.05**	**67.78**
Tagger RoBERTa	66.22	72.14	69.05
+LLM cls & LLM cor	72.71	68.10	70.33
+LLM cls & FT cor	**73.69**	68.97	71.25
+FT cls & LLM cor	71.71	71.48	71.59
+FT cls & FT cor	72.39	**72.12**	**72.26**

Table 7: Performance of ACC framework with LLM on the dev set of MultiSpanQA. "LLM cls/cor" refers to classifier/corrector replaced by LLM, and "FT cls/cor" refers to a fine-tuned model. The best performance is in **bold**.

require models to extract one or multiple answer spans from a given context. Therefore, traditional SSQA models (Seo et al., 2017; Yu et al., 2018) are not sufficient to handle multi-span questions.

Existing MSQA methods can be categorized into four categories: (1) pointer-network-based methods. MTMSN (Hu et al., 2019) predicts the number of answers, then extracts non-overlapped answer spans; MUSST (Yang et al., 2021) uses an autogressive approach to iteratively extract multiple answers. (2) sequence-tagging-based methods. Segal et al. (2020) first convert MSQA task to a sequence-tagging task and utilize BIO tags to mark answer spans; Furthermore, Li et al. (2022) introduce multi-task learning and achieve better performance. (3) span-enumeration-based methods. SpanQualifier (Huang et al., 2023a) utilizes Multi-Layer Perceptron (MLP) to obtain confidence scores for each candidate span and applies a learnable threshold to select answer spans; Similarly, CSS (Zhang et al., 2024) compares each candidate span with its corresponding question after scoring to obtain answers more similar to the question. (4) LLM-based methods. With the emergence of LLMs like ChatGPT and GPT-4, generative pre-trained language models have been widely applied to various NLP tasks. Zhang et al. (2023) employ CoT strategies to prompt LLM, and Huang et al. (2023b) add negative examples in the few-shot demonstrations.

Existing methods mainly focus on predicting more correct predictions, while the ACC framework takes a post-processing strategy which aims to reduce the number of incorrect predictions. By excluding or modifying incorrect predictions, the ACC framework achieves better performance.

6.2 Post-Processing Methods

The post-processing method refers to modifying the original of the model to obtain better predictions. Existing post-processing methods can be categorized into two types: rule-based methods and model-based methods.

Ruled-based methods typically involve mannually designed rules such as voting to process the outputs from models (Campos and Couto, 2021; Wang et al., 2023). On the other hand, model-based methods utilize additional models to modify the hidden states or outputs of the original model, which have been widely applied in Controlled Text Generation (CTG) (Yang and Klein, 2021; Krause et al., 2021; Kim and Cho, 2023). In addition to CTG methods, GRACE (Khalifa et al., 2023) applies a fine-tuned discriminator to guide language model towards correct multi-step solutions; Ohashi and Higashinaka (2023) utilize a generative model to rewrite the output from a dialogue system and optimize it with Reinforcement Learning (RL) algorithms (Stiennon et al., 2020).

The work most similar to ours is (Gangi Reddy et al., 2020), which utilizes a corrector to modify the outputs of the SSQA model. However, they only focus on partial matches in single-span questions. In constrast, we consider the correctness of multiple predictions in MSQA and additionally employ a classifier to exclude incorrect predictions.

7 Conclusion

In this work, we primarily focus on incorrect predictions of the MSQA models. Through a statistical analysis, we observe that models with better performance do not predict less incorrect predictions compared to other models. To this end, we propose ACC framework, which employ a post-processing strategy to exclude wrong predictions and modify partially correct predictions. Experiments and analysis show that the ACC framework significantly improving the performance by reducing the number of incorrect predictions and obtaining more correct predictions, enhancing the quality of the MSQA predictions.

8 Limitations and Future Works

In this work, we categorize incorrect predictions into "partially correct predictions" and "wrong predictions", based on whether the answer should be modified or excluded. However, for "partially correct predictions", there exists more complicated conditions, for example, an incorrect prediction may responses to multiple gold answers. However, the ACC framework can only obtain one modified prediction. In addition, we do not consider the gold answers that MSQA models fail to predict (i.e., "missing answers"), although the SOTA model still miss 1/3 gold answers. As for future works, we will design more effectively models to handle with "partially correct predictions" and "wrong predictions". we will also explore strategies to handle with "missing answers".

References

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Margarida M. Campos and Francisco M. Couto. 2021. Post-processing biobert and using voting methods for biomedical question answering. In *Proceedings of the Working Notes of CLEF 2021 - Conference and Labs of the Evaluation Forum, Bucharest, Romania, September 21st - to - 24th, 2021*, volume 2936 of *CEUR Workshop Proceedings*, pages 258–273. CEUR-WS.org.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Revanth Gangi Reddy, Md Arafat Sultan, Efsun Sarioglu Kayi, Rong Zhang, Vittorio Castelli, and Avi Sil. 2020. Answer span correction in machine reading comprehension. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2496–2501, Online. Association for Computational Linguistics.

Minghao Hu, Yuxing Peng, Zhen Huang, and Dongsheng Li. 2019. A multi-type multi-span network for reading comprehension that requires discrete reasoning. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1596–1606, Hong Kong, China. Association for Computational Linguistics.

Zixian Huang, Jiaying Zhou, Chenxu Niu, and Gong Cheng. 2023a. Spans, not tokens: A span-centric model for multi-span reading comprehension. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, CIKM '23, page 874–884, New York, NY, USA. Association for Computing Machinery.

Zixian Huang, Jiaying Zhou, Gengyang Xiao, and Gong Cheng. 2023b. Enhancing in-context learning with answer feedback for multi-span question answering. In *Natural Language Processing and Chinese Computing*, pages 744–756, Cham. Springer Nature Switzerland.

Yiming Ju, Weikang Wang, Yuanzhe Zhang, Suncong Zheng, Kang Liu, and Jun Zhao. 2022. CMQA: A dataset of conditional question answering with multiple-span answers. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 1697–1707, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.

Muhammad Khalifa, Lajanugen Logeswaran, Moontae Lee, Honglak Lee, and Lu Wang. 2023. GRACE: Discriminator-guided chain-of-thought reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 15299–15328, Singapore. Association for Computational Linguistics.

Heegyu Kim and Hyunsouk Cho. 2023. GTA: Gated toxicity avoidance for LM performance preservation. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 14747–14763, Singapore. Association for Computational Linguistics.

Takeshi Kojima, Shixiang (Shane) Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. In *Advances in Neural Information Processing Systems*, volume 35, pages 22199–22213. Curran Associates, Inc.

Ben Krause, Akhilesh Deepak Gotmare, Bryan McCann, Nitish Shirish Keskar, Shafiq Joty, Richard Socher, and Nazneen Fatema Rajani. 2021. GeDi: Generative discriminator guided sequence generation. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 4929–4952, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Matthew Kelcey, Jacob Devlin, Kenton Lee, Kristina N. Toutanova, Llion Jones, Ming-Wei Chang, Andrew Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural questions: a benchmark for question answering research. *Transactions of the Association of Computational Linguistics*.

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. 2017. RACE: Large-scale ReAding comprehension dataset from examinations. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 785–794, Copenhagen, Denmark. Association for Computational Linguistics.

Haonan Li, Martin Tomko, Maria Vasardani, and Timothy Baldwin. 2022. MultiSpanQA: A dataset for multi-span question answering. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1250–1260, Seattle, United States. Association for Computational Linguistics.

Atsumoto Ohashi and Ryuichiro Higashinaka. 2023. Enhancing task-oriented dialogue systems with generative post-processing networks. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 3815–3828, Singapore. Association for Computational Linguistics.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for SQuAD. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 784–789, Melbourne, Australia. Association for Computational Linguistics.

S. E. Robertson and S. Walker. 1994. Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In *SIGIR '94*, pages 232–241, London. Springer London.

A. Rosenfeld and M. Thurston. 1971. Edge and curve detection for visual scene analysis. *IEEE Transactions on Computers*, C-20(5):562–569.

Elad Segal, Avia Efrat, Mor Shoham, Amir Globerson, and Jonathan Berant. 2020. A simple and effective model for answering multi-span questions. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3074–3080, Online. Association for Computational Linguistics.

Min Joon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2017. Bidirectional attention flow for machine comprehension. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. 2020. Learning to summarize with human feedback. In *Advances in Neural Information Processing Systems*, volume 33, pages 3008–3021. Curran Associates, Inc.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35, pages 24824–24837. Curran Associates, Inc.

Junjie Yang, Zhuosheng Zhang, and Hai Zhao. 2021. Multi-span style extraction for generative reading comprehension. In *Proceedings of the Workshop on Scientific Document Understanding co-located with 35th AAAI Conference on Artificial Inteligence, SDU@AAAI 2021, Virtual Event, February 9, 2021*, volume 2831 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Kevin Yang and Dan Klein. 2021. FUDGE: Controlled text generation with future discriminators. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3511–3535, Online. Association for Computational Linguistics.

Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V. Le. 2018. Qanet: Combining local convolution with global self-attention for reading comprehension. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.

Zhiang Yue, Jingping Liu, Cong Zhang, Chao Wang, Haiyun Jiang, Yue Zhang, Xianyang Tian, Zhedong Cen, Yanghua Xiao, and Tong Ruan. 2023. Mamrc: A multi-answer machine reading comprehension dataset. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '23, page

2144–2148, New York, NY, USA. Association for Computing Machinery.

Chen Zhang, Jiuheng Lin, Xiao Liu, Yuxuan Lai, Yansong Feng, and Dongyan Zhao. 2023. How many answers should I give? an empirical study of multi-answer reading comprehension. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 5811–5827, Toronto, Canada. Association for Computational Linguistics.

Penghui Zhang, Guanming Xiong, and Wen Zhao. 2024. Css: Contrastive span selector for multi-span question answering. In *PRICAI 2023: Trends in Artificial Intelligence*, pages 225–236, Singapore. Springer Nature Singapore.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. Bertscore: Evaluating text generation with bert. In *International Conference on Learning Representations*.

Pengfei Zhu, Zhuosheng Zhang, Hai Zhao, and Xiaoguang Li. 2022. Duma: Reading comprehension with transposition thinking. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 30:269–279.

Liu Zhuang, Lin Wayne, Shi Ya, and Zhao Jun. 2021. A robustly optimized BERT pre-training approach with post-training. In *Proceedings of the 20th Chinese National Conference on Computational Linguistics*, pages 1218–1227, Huhhot, China. Chinese Information Processing Society of China.

A More Details of Experimental Setup

A.1 Datasets

MultiSpanQA (Li et al., 2022) and **MultiSpanQA-Expand** (Li et al., 2022): MultiSpanQA and MultiSpanQA-Expand focus on multi-span questions. The raw questions and contexts are extracted from the Natural Question dataset (Kwiatkowski et al., 2019). MultiSpanQA only contains multi-span questions, while MultiSpanQA-Expand contains both multi-span questions, single-span questions and unanswerable questions.

MAMRC (Yue et al., 2023) and **MAMRC-Multi**: MAMRC is a large-scale dataset containing over 100,000 questions, including both multi-span questions and single-span questions. To investigate the performance on the multi-span questions, we select multi-span questions from MAMRC and obtain MAMRC-Multi.

Since the official test sets of these four datasets are not public, we report the performance on dev sets. Some statistics about the four datasets are shown in Table 8.

A.2 Baselines

MTMSN (Hu et al., 2019): MTMSN adds a classification head to predict the number of answers. During the inference stage, for each question, MUSST first obtains top-20 predictions and predict answer number K, then applies Non-Maximum Sampling algorithm (Rosenfeld and Thurston, 1971) to extract K non-overlapped spans.

MUSST (Yang et al., 2021): MUSST adds m linear layer to predict the start position and end position of m spans, where m is the maximum answer number in the training dataset. During the inference stage, MUSST applies an autogressive decoding strategy, where in each iteration MUSST masks out predicted spans and chooses top-1 predictions. The iterative process terminates when the model predicts no more answers or the number of predictions reaches the maximum answer number.

Tagger: Following the implementation of (Li et al., 2022), we utilize BIO tags to label each token in context: the first token of the answer is labeled with "B", the other tokens of the answer are labeled with "I" and the tokens not in an answer are labeled with "O".

SpanQualifier (Huang et al., 2023a) SpanQualifier enumerates all possible answer spans and obtains their corresponding confidence scores as correct predictions, then utilizes a learnable threshold to select the correct prediction spans, achieving state-of-the-art performance on MultiSpanQA-Expand dataset.

A.3 Implementation Details

When sampling training data for ACC framework, we set split number $K = 3$, which means in each iteration, we use two-thirds of the training data for training and sample the predictions on the remaining data. for the classifier, we maintain a balanced ratio of 1:1:1 among the three answer categories for the classifier, and for the corrector, we added examples that require no modifications and maintained a ratio of 2:1 between examples requiring modifications and examples requiring no modifications, considering that corrector may not necessarily modifies all the input predictions.

During training stage of classifier and corrector, for MultiSpanQA and MultiSpanQA-Expand, we set $learning\_rate = 3e^{-5}$, $batch\_size = 48$, $epochs = 10$ and $max\_length = 512$; For MAMRC and MAMRC-Multi, we set $learning\_rate = 3e^{-5}$, $batch\_size = 96$,

	#train	#dev	answer number prop.			avgerage answer number	average context length	avgerage question length
			≥2	1	0			
MultiSpanQA	5,230	658	100.0%	0.0%	0.0%	2.89	279	10
MultiSpanQA-Expand	15,690	1,959	33.4%	33.3%	33.3%	1.30	251	10
MAMRC	110,108	13,764	58.7%	41.3%	0.0%	1.77	69	10
MAMRC-Multi	64,625	8,081	100.0%	0.0%	0.0%	2.31	77	10

Table 8: Dataset statistics.

	MultiSpanQA			**MultiSpanQA-Expand**			**MAMRC**			**MAMRC_Multi**		
	PM P	PM R	PM F1	PM P	PM R	PM F1	PM P	PM R	PM F1	PM P	PM R	PM F1
BERT-base												
MTMSN	69.97	79.23	74.30	73.29	73.46	73.37	84.59	89.62	87.03	84.68	89.97	87.25
+ACC	**81.10**	66.77	73.24	**77.20**	67.04	71.76	**88.45**	**85.86**	**87.13**	**90.74**	**85.68**	**88.14**
MUSST	76.39	68.76	72.38	77.79	70.99	74.22	87.25	88.25	87.74	87.75	87.69	87.72
+ACC	**81.25**	65.68	**72.64**	**78.36**	68.65	73.17	**88.68**	85.46	87.04	**90.93**	84.61	87.66
Tagger	78.27	77.92	78.09	70.60	65.75	68.05	88.81	89.05	88.92	88.23	84.98	86.57
+ACC	**83.30**	77.29	**80.19**	**74.06**	**66.64**	**70.14**	**89.07**	87.13	88.09	**90.85**	83.54	**87.04**
SpanQualifier	81.17	79.70	80.43	74.01	76.73	75.34	87.75	90.94	89.31	87.55	91.90	89.67
+ACC	**84.26**	77.70	**80.84**	**76.20**	75.15	**75.67**	**88.83**	87.94	88.38	**90.78**	88.37	89.56
RoBERTa-base												
MTMSN	77.57	82.29	79.86	76.36	76.80	76.58	85.77	89.72	87.70	85.15	90.18	87.60
+ACC	**85.65**	72.12	78.30	**78.88**	69.93	74.14	**88.74**	**86.21**	87.46	**90.45**	86.08	**88.21**
MUSST	83.44	75.72	79.39	80.22	73.36	76.63	88.64	88.44	88.54	88.65	86.64	87.63
+ACC	**85.41**	73.24	78.86	79.99	70.83	75.13	**89.42**	85.95	87.65	**91.17**	83.89	87.38
Tagger	83.97	83.92	83.94	77.91	75.43	76.64	90.09	90.22	90.15	88.07	85.90	86.98
+ACC	**86.60**	82.67	**84.59**	**79.43**	74.62	**76.95**	89.92	89.01	89.46	**90.81**	84.20	**87.38**
SpanQualifier	83.85	83.17	83.50	76.77	78.62	77.65	89.82	88.19	89.00	87.27	92.14	89.63
+ACC	**86.39**	81.27	**83.74**	**78.69**	76.67	**77.66**	**89.34**	**88.98**	**89.16**	**90.49**	88.82	**89.65**

Table 9: PM scores on four MSQA datasets.

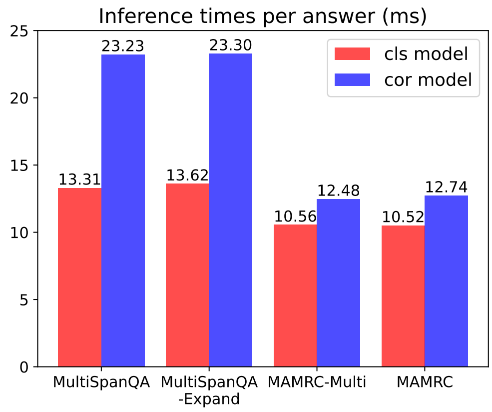

Figure 4: Inference times on four datasets.

$epochs = 5$ and $max\_length = 256$. We choose the best classifier and corrector on our sliver-labeled dev sets. All the baselines were trained with three different seeds and we report the mean results. We perform our experiments on a single Tesla V-100 GPU(32GB).

model	**BERT-base**	**RoBERTa-base**
MTMSN	110M	125M
MUSST	110M	125M
Tagger	109M	125M
SpanQualifier	115M	131M
classifier	-	128M
corrector	-	124M

Table 10: Model sizes of baselines model, the classifier and the corrector.

B Additional Experiments and Discussions

B.1 Partial Match Results

The Partial Match results are shown in Table 9. While EM F1 scores show significant improvements after applying the ACC framework, PM F1 scores achieve less improvements and even decline in some cases. The main reason may be that PM scores consider the overlaps between predictions and gold answers, as a result, incorrect predictions may contribute to PM F1 score (i.e., $EM\ F1 = 0, PM\ F1 > 0$). However, such predictions are not desired and may be excluded by the ACC framework, limiting the improvements in

PM F1 scores.

B.2 Model Size and Inference Time

We compare model sizes between MSQA models and the ACC framework, shown in Table 10. The ACC framework improves the performance of baselines without applying large-size models, avoiding consuming excessive computational resources.

We also analyze inference times of the ACC framework, shown in Figure 4. The results demostrate that the ACC framework is time-effective, especially when the input length is short (we set $max\_length = 256$ for MAMRC and MAMRC-Multi and we set $max\_length = 512$ for MultiSpanQA and MultiSpanQA-Expand).

B.3 Implementation details of pilot study with LLM

We use OpenAI's official API [6] and select the model gpt-3.5-turbo-0120 for our pilot study. Due to the poor performance in zero-show settings, we apply In-Context Learning (ICL) (Brown et al., 2020) and utilize a BM25 retriever (Robertson and Walker, 1994) to select the demonstrations which is similar to the questions. When replacing the classifier, we select one demonstration for each answer type; when replacing the corrector, we select two demostractions for answers requiring modification and requiring no modification. The prompts are shown in Table 11.

[6] https://platform.openai.com/.

cls model prompt:
For this task, we will provide you a passage and a question. The question contains one or multiple answers and these answers are in the passage. We will also provide you a candidate answer from our AI model. You should read the passage, the question and classify the candidate answer into one of three classes: "correct prediction", "partially correct prediction" and "wrong prediction". Correct prediction refers to a completely correct prediction; Partially correct prediction refers to a prediction that is basically correct but still requires some modifications. Wrong prediction refers to a prediction that is completely incorrect and should be excluded.\You should output your answer in a json format like "{{"answer":"your_answer"}}", DO NOT include any explanations in your responses. Example1: Passage: ... Question: ... Candidate Answer:... Output: {"answer":"correct prediction"} ... Query: Passage: ... Question: ... Candidate Answer: ... Output:
cor model prompt:
For this task, we will provide you a passage and a question. The question contains one or multiple answers and these answers are in the passage. We will also provide a candidate answer that our AI model believes needs some modifications. You should read the passage, the question and judge whether the candidate answer requires modifications. If no modifications are needed, you should output the candidate answer as is. Otherwise, you should modify it by adding or deleting some words, and the modified prediction must be a part of the passage and similar to the original candidate answer.\You should output your answer in a json format like "{{"answer":"your_answer"}}", DO NOT include any explanations in your responses. Example1: Passage: ... Question: ... Original Answer: ... Output: {"answer":"xxx"} ... Query: Passage: ... Question: ... Candidate Answer: ... Output:

Table 11: Prompts for pilot study with LLM