Dual Multi-head Co-Attention For Abstract Meaning Reading Comprehension

Jiachen Jiang
jiachenj@umich.edu

Chengtian Zhang
zctchn@umich.edu

Abstract

This paper describes our system used in SemEval-2021 shared task 4: Reading Comprehension of Abstract Meaning, which requires the participating system to fill in the the correct answer from five candidates of abstract concepts in a cloze-style to replace the @Placeholder in the question. In this report, we basically choose ELECTRA as our encoder, and try to add Multihead Attention Multichoice Classifier(MAMC) and DUal Multi-head Co-Attention (DUMA) classifier as a one-layer classifier. They both achieve higher performance than ELECTRA itself. On conclusion, our ELECTRA + DUMA approach tends to perform out other methods as our best result, it ranks 3rd for task 1 and 5th for task 2 with the accuracy of 89.95%, 91.41%. Our github repository is avilable here.

1 Introduction

We plan to work on SemEval-2021 shared task 4: Reading Comprehension of Abstract Meaning (ReCAM)(Zheng et al., 2021).
Machine reading comprehension (MRC) tasks is designed to help evaluate the ability of machines in representing and understanding human languages and reasoning. Given a passage, the machine is expected to give the answers of questions related with this passage. Specifically, SemEval-2021 shared task 4 require the participating system to fill in the the correct answer from five candidates of abstract concepts in a cloze-style to replace the *@Placeholder* in the question. Instead of predicting concrete concepts in the previous work, in this task we ask models to choose abstract words removed from human-written summaries. There are there subtasks to evaluate the performance of the model based on two aspects of *abstractness*, *imperceptibility* and *nonspecificity*.

- **Subtask 1** focus on evaluating the system's ability in understanding *imperceptibility*, which are words that cannot be directly perceived in the physical world, e.g. service/economy compared with trees/red;.
- **Subtask 2** aims to measuring the system's ability in comprehending *nonspecificity*, which are nonspecific concepts located high in a hypernym hierarchy given the context of a passage, e.g. vertebrate compared with monkey.
- **Subtask 3** focus on the model's transferability over the two types of *abstractness*, we need to train the model on the Subtask 1 evaluate it on Subtask 2 and vice versa.

2 Related Work

According to the type of the answer, Machine reading comprehension (MRC) tasks can be divided into the following three categories Chen, 2018.

- **Span prediction**. Extractive question answering requires the system to extract a suitable range of text fragments from a given original text based on the question as to the answer.(Hermann et al., 2015; Hill et al., 2016; Onishi et al., 2016; Rajpurkar et al., 2016; Trischler et al., 2017)
- **Free-form answer**: This task require models to generate an answer. It requires the system to mine deep-level contextual semantic information according to a given question to give the best answer.(Nguyen et al., 2016; He et al., 2018;Kociský et al., 2017)
- **Cloze-style**: The system must choose a word or entity from the set of candidate answers to fill in the "@placeholder" in the question to make the sentence complete.(Hermann et al., 2015; Hill et al., 2016; Onishi et al., 2016;)

This task is similar to the Cloze-style task. Unlike previous work, ReCAM questions specifically focus on abstract words.

3 Approaches

We propose an approach based on pre-trained language models(LMs) and DUal Multi-head Co-Attention (DUMA) multi-choice classifier with negative data augment to get the final result. The overall architecture of our system mainly consists of three parts: Negative Augmentation, Pre-trained LMs and DUMA multi-choice classifier.

3.1 Pretrained LMs

Deep learning based NLP models require much larger amounts of data. A task-specific datasets contain only a few thousand or a few hundred thousand human-labeled training examples, which is not sufficient for training process. To help bridge this gap in data, researchers have developed various techniques for training general purpose language representation models using the enormous piles of unannotated text on the web (this is known as pre-training). These general purpose pre-trained models can then be fine-tuned on smaller task-specific datasets, e.g. ReCAM, to complete the Abstract Meaning Reading Comprehension Task.

Bidirectional Encoder Representations from Transformers (BERT) is a pre-trained language model which can be finetuned with just one additional output layer to create state-of-the-art models for a wide range of tasks. It makes use of a novel technique called Masked Language Models (MLM): it randomly masks words in the sentence and then it tries to predict them. Previous language models, e.g. GRU or LSTM, would have looked at this text sequence during training from either left-to-right or right-to-left in one direction. Bert is bidirectional and takes both the previous and next tokens into account at the same time. It would predict the masked word based on both previous context and next context.

3.1.1 ROBERTa

Robustly optimized BERT approach (ROBERTa) is based on BERT, BERT uses the now ubiquitous transformer architecture.

During training, BERT uses two objectives: masked language modeling and next sentence prediction.

Masked Language Model (MLM) A random sample of the tokens in the input sequence is selected and replaced with the special token [MASK].
More precisely speaking, before feeding word sequences into BERT, for each sentence 15% of their words are replaced with a [MASK] token. The model then attempts to predict the original value of the masked words, based on the context provided by the other, non-masked, words in the sequence. In technical terms, the prediction of the output words requires: 1) Adding a classification layer on top of the encoder output. 2) Multiplying the output vectors by the embedding matrix, transforming them into the vocabulary dimension. 3) Calculating the probability of each word in the vocabulary with softmax.

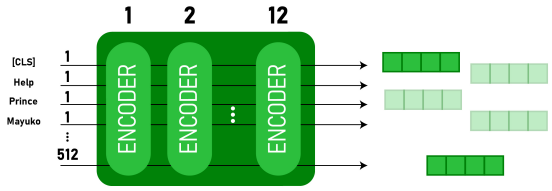

Figure 1: BERT output as Embeddings

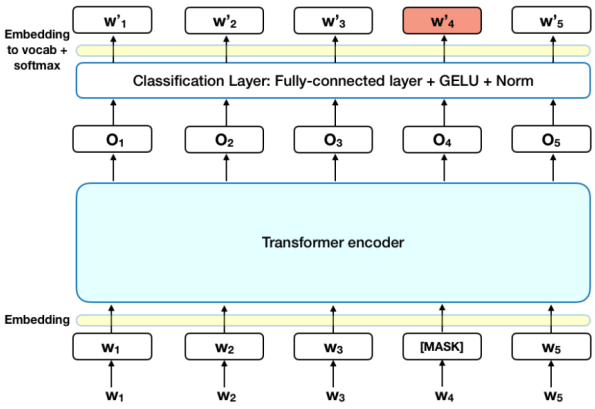

Figure 2: MLM

Next Sentence Prediction (NSP) NSP is a binary classification loss for predicting whether two segments follow each other in the original text. Positive examples are created by taking consecutive sentences from the text corpus. Negative examples are created by pairing segments from different documents. Positive and negative examples are sampled with equal probability. NSP works well in tasks where relationship between two sentences are needed to be detected.

Roberta(Liu et al., 2019) is an improved recipe for training BERT models that can match or exceed the performance of all of the post-BERT methods.

The modifications are simple: (1)training the model longer, with bigger batches, over more data; (2) removing the next sentence prediction objective; (3) training on longer sequences; (4) dynamically changing the masking pattern applied to the training data.

3.1.2 ELECTRA

Several models have since been introduced that have improved on the success of BERT — like RoBERTa and XLNet — but all of them rely on bigger networks and larger datasets. Efficiently Learning an Encoder that Classifies Token Replacements Accurately (ELECTRA) model has been introduced as a new approach to training that produces a model with comparable or better performance than even the best transformers, and requires just a fraction of the compute power. Instead of using Masked Language Models (MLM), it uses a more efficient approach called "replaced token detection" to train the transformer. It contains both a generator and a discriminator.

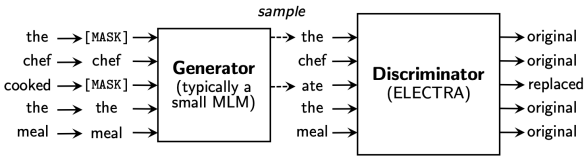

Figure 3: An overview of replaced token detection

The generator is a MLM model which would guess the value of the [MASK] in the sentence. Then the discriminator would identify which tokens are original and which are fakes that have been placed there by the generator. After training is completed, we would discard the generator model and using the discriminator as our new ELECTRA transformer model. Then we can apply ELECTRA as the encoder for our following specific tasks. The "replaced token detection" approach is more efficient than MLM because the model must consider every single token in every single sample. But MLM only requires the model to focus on [MASK] tokens. This is not a GAN-type architecture because the generator is not optimized to increase the loss of the discriminator.

3.2 Multi-head Attention Multi-choice Classifier(MAMC)

In this part, we tried to add a multi-head attention multi-choice classifier (MAMC) as described in (Vaswani et al., 2017).

3.2.1 Attention

An attention function can be described as mapping a query and a set of key-value pairs to an output, where the query, keys, values, and output are all vectors. The output is computed as a weighted sum of the values, where the weight assigned to each value is computed by a compatibility function of the query with the corresponding key.

3.2.2 Scaled Dot-Product Attention

We call our specified attention "Scaled Dot-Product Attention", like shown in Figure 4.

The input consists of queries and keys of dimen-

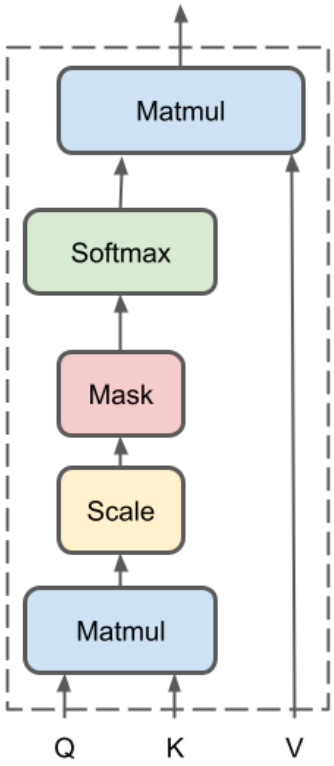

Figure 4: Scaled Dot-Product Attention.

sion d_k, and values of dimension d_v. We compute the dot products of the query with all keys, then divide each by $\sqrt{d_k}$. After that we apply a softmax function to obtain the weights on the values. For a set of queries, we compute the attention function simultaneously and pack results together into a matrix Q. The keys and values are also packed together into matrices K and V . So that the output

can be calculated as below:

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V, \quad (1)$$

where $\frac{1}{d_k}$ is the scaling factor.
In comparison with additive attention (Bahdanau et al., 2016), dot-product attention is faster and more space-efficient in practice though they are similar in complexity because dot-product attention can be implemented using highly optimized matrix multiplication code.

3.2.3 Multi-Head Attention

Instead of performing a single attention function with d_{model}-dimensional keys, values and queries, we found it beneficial to linearly project the queries, keys and values h times with different, learned linear projections to d_k, d_k and d_v dimensions respectively. On each of these projected triples we then perform the attention function, yielding d_v-dimensional output values. These are concatenated and projected again and output the final values, as depicted in Figure 5. Multi-head attention allows

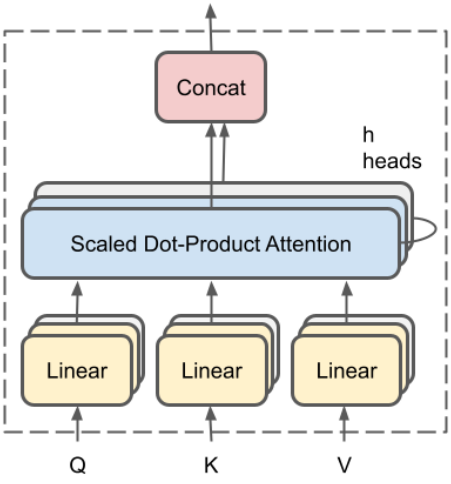

Figure 5: Multi-Head Attention consists of several attention layers running in parallel.

the model to jointly obtain information from different representation subspaces, while with a single attention head averaging inhibits this. And the output can be calculated as:

$$MultiHead(Q, K, V) = concat(head_1, head_2, ...head_h)W^O, \quad (2)$$

$$head_i = Attention(QW_i^Q, KW_i^K, VW_i^V), \quad (3)$$

where $W_i^Q \in \mathbb{R}^{d_{model} \times d_k}$, $W_i^K \in \mathbb{R}^{d_{model} \times d_k}$, $W_i^V \in \mathbb{R}^{d_{model} \times d_v}$, and $W_i^O \in \mathbb{R}^{hd_v \times d_{model}}$.

3.2.4 MAMC

While putting into usage in MRC, we firstly take the encoder output as input Q, K, V of muti-head attention layer. Then calculate the attention representations from the concatenated embeddings. Finally take the out put as the input of decoder and get the option with highest possible rate as our prediction. And the structure is showed in Figure 6 below.

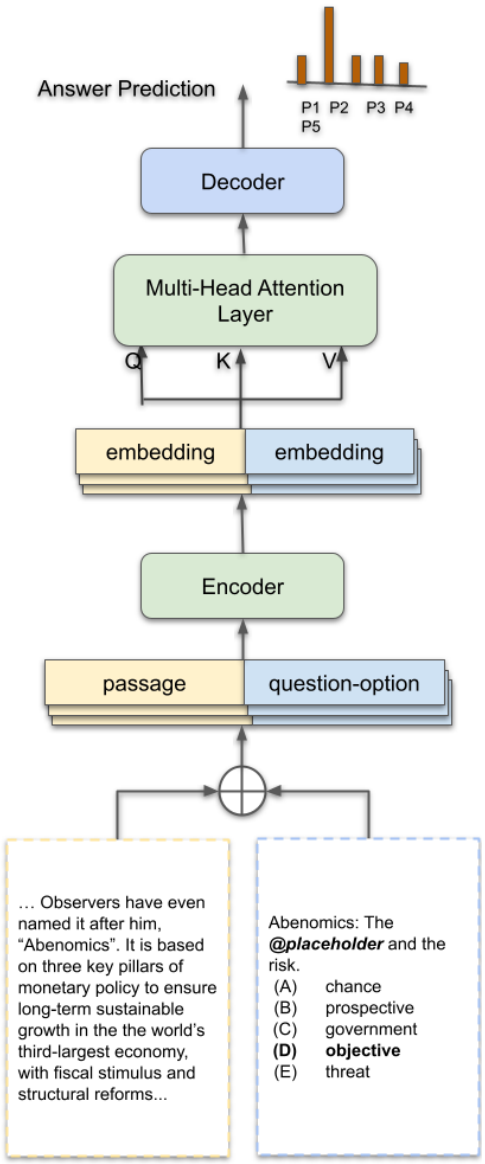

Figure 6: An overview of the overall architecture of MAMC

3.3 DUMA Multi-choice Classifier

In this part, we would add an extra attention layer which is Dual Multi-head Co-Attention (DUMA) module as described in (Zhu et al., 2020). Basically, it involves 1) splitting the output sequence from the encoder into question-answer sequence and passage sequence; 2) calculating two attention representations from the two sequences, one from the passage attending the question-answer, the other vice versa; 3) concatenate the two attention representations together after individually mean-pooled; 4) The representations would be sent to the classifier. The answer option with the highest probability is picked as the predicted answer. We would use the Cross Entropy function between the ground truth and the predicted probabilities to compute the loss.

3.3.1 Encoder

The encoder would encode input tokens into representations. The encoder can either be context-free or context-based. Context-free models like Word2Vec would generate a single word embedding representation (a vector of numbers) for each word in the vocabulary. The same word may have different meanings in different context. The context-based models like Pretrained LMs can generate a representation of each word that is based on the other words in the sentence, which gets a better understanding of words in the context. Therefore we choose Pretrained LMs, e.g. ROBERTa and ELECTRA, as our encoder. We fill the **@Placeholder** in the question with options and concatenate it with the corresponding passage form one sequence and then feed it into the encoder. Let $P = [p_1, p_2, ..., p_m]$, $Q = [q_1, q_2, ..., q_n]$, $O = [o_0, o_1, o_2, o_3, o_4]$ be passage, question and options, where p_i,q_i and o_i are token ids in passage, question and options. The concatenation of P, Q, O would be as follows. The passage would be truncated or padding to ensure the input length of the token ids would be 5×256. The encoding output E has a form $[e_1, ..., e_{5\times256}]$, where e_i is a vector of fixed dimension d_{hidden_size} that represents the respective token.

$$
\begin{aligned}
&[CLS] + [p_1, p_2, ..., p_m] + [SEP] + [q_1, q_2, ..., + [o_0] + q_n] \\
&[CLS] + [p_1, p_2, ..., p_m] + [SEP] + [q_1, q_2, ..., + [o_1] + q_n] \\
&[CLS] + [p_1, p_2, ..., p_m] + [SEP] + [q_1, q_2, ..., + [o_2] + q_n] \\
&[CLS] + [p_1, p_2, ..., p_m] + [SEP] + [q_1, q_2, ..., + [o_3] + q_n] \\
&[CLS] + [p_1, p_2, ..., p_m] + [SEP] + [q_1, q_2, ..., + [o_4] + q_n]
\end{aligned}
$$

Figure 7: The concatenation of passage, question and options.

3.3.2 Dual Multi-head Co-Attention

Dual Multi-head Co-Attention(DUMA) would reuse the architecture of Multi-head Attention layer. However, different from MAMC which contains only one Multi-head Attention layer, DUMA would use two Multi-head Attention layers in parallel and calculate the attention representations in a bi-directional way. It would separate the representation from the encoder into passage embedding as E_P and question-option embedding as E_{QO}.

Then for one Multi-head Attention layer, it would take E_P as $Query$ and Key and take E_{QO} as $Value$. For the other one, it would take E_{QO} as $Query$ and Key and take E_P as $Value$.

$$
\begin{aligned}
MHA_1 &= MultiHead(E_P, E_P, E_{QO}) \\
MHA_2 &= MultiHead(E_{QO}, E_{QO}, E_P)
\end{aligned} \tag{4}
$$

$$
DUMA(E_P, E_{QO}) = POOL(MHA_1, MHA_1) \tag{5}
$$

Where MHA_1 and MHA_2 are output of the two Multi-head Attention Layers. $POOL()$ uses mean pooling to pool the sequence outputs, and $DUMA()$ is our Dual Multi-head Co-Attention Layer.

3.3.3 Decoder

Our model decoder takes the outputs of DUMA and computes the probability distribution over answer options.

$$
O = DUMA(E_P, E_{QO}) \tag{6}
$$

$$
P(O_{correct}|P, QO) = Softmax(DropOut(W^T O)) \tag{7}
$$

Where O are output of DUMA layer, and W^T is a learnable parameter in linear layer. $DropOut()$ layer adds 5 DropOut layer with dropout rate of 0.5 to prevent overfitting. $P(O_{correct}|P, QO)$ is the probability of the 5 options. Highest probability indicates the correct answer.

4 Evaluation

4.1 Dataset

- **ReCAM**. Dataset for the SemEval-2021 Task 4. Data is stored one-question-per-

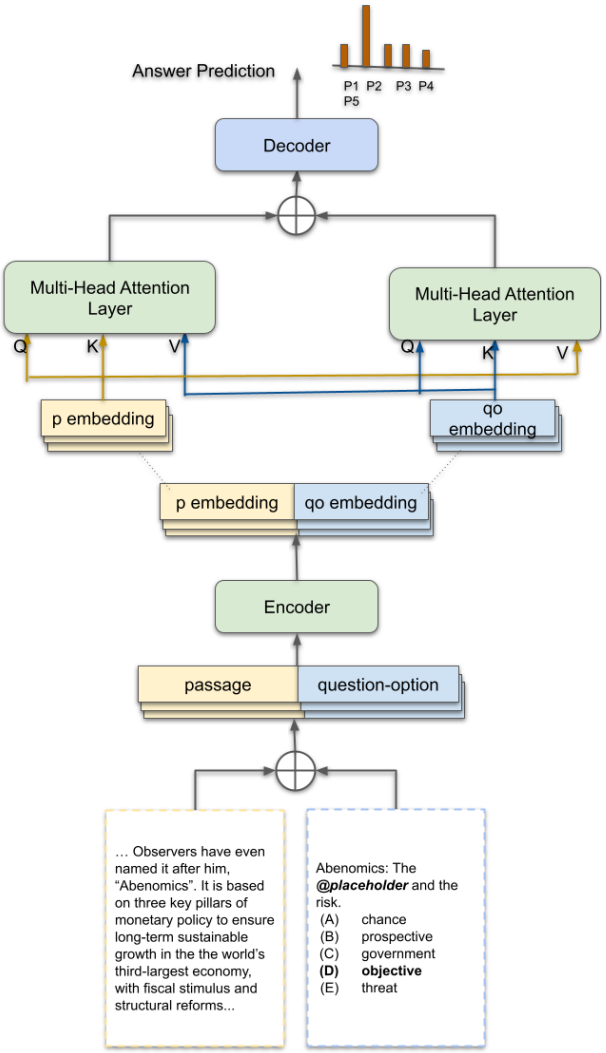

Figure 8: An overview of the overall architecture of DUMA

line in json format, including article, question, options and label. In Subtask 1, the training/ trail/ development/ test contains 3,227/1,000/837/2,025 instances. In Subtask 2, the training/ trail/ development/ test contains 3,318/1,000/851/2,017 instances.

- **CNN/Daily Mail**. It consists 300k unique news articles as written by journalists at CNN and the Daily Mail. We would use it to implement task-adaptive pretraining.

4.2 Implementation Plan

We implement the our method for the following three steps:

- **Data Pre-processing** For each passage, there is a summary with five candidate answers. We substitute the options into the summary to form several complete sentences. Then we concatenate the option-filled sentence and passage tokens as input samples, wrapped by [CLS] token and [SEP] tokens.

- **Task-adaptive Pretraining** We need to apply task-adaptive pretraining on LMs, to get better embedding Gururangan et al., 2020. Most LMs are trained in the general domain corpus such as Wikipedia. Task adaptive pre-training would use domain-specific data, e.g. CNN daily, to make the model better fit the distribution in the domain.

- **Fine-tuning** We would fine-tune our PLM encoder on ReCAM. We plan to implement with PyTorch to get the pre-trained language models, then use AdamW optimizer to fine-tune the models. We pick the best learning rate from the dev set and set batch size small due to the limit of GPU memory of Google Colab.

4.3 Network Layers

Table 1 shows our Electra network structure, the parameter amount and whether they are trainable. We use Electra model as encoder and a linear trainable layer as classifier.

Name	Type	Params	Trainable
electra	ElectraModel	334 M	No
classifier	Linear	1.0 K	Yes

Table 1: Network layers of Electra and parameter size

Table 2 shows network layers of Electra + MAMC model. In this structure, a trainable Mutihead attention layer is added after encoder layer, the parameter added is only 4.2M. And a dropout layer is added to avoid overfitting as well.

Name	Type	Params	Trainable
electra	ElectraModel	334 M	No
mamc	MultiheadAttentionLayer	4.2 M	Yes
dropouts	ModuleList	0 M	None
classifier	Linear	1.0 K	Yes

Table 2: Network layers of Electra + MAMC and parameter size

Table 3 shows network layers of Electra + DUMA model. In this structure, a trainable Dual

Multi-head Co-Attention layer is added after encoder layer. Since the co-attention layer shoulders double workload compared with that in MAMC, the parameter added is 8.4M. And a dropout layer is added to avoid overfitting as well.

Name	Type	Params	Trainable
electra	ElectraModel	334 M	No
duma	DUMALayer	8.4 M	Yes
dropouts	ModuleList	0 M	None
classifier	Linear	1.0 K	Yes

Table 3: Network layers of Electra + DUMA and parameter size

4.4 Hyperparameter Setting

Table 4 shows our fine-tuned hyperparameters. In the initial tokenizer layer, we set the token max length to 256 to limit the computation load. And we only need the truncation at the first time, so the truncation is set to "only first" option, and the padding is set to "max length" option to normalize sentence sizes. While training, we set learning rate to 1e-4 after many tries and set batch size to 2 due to GPU limitaion. To accelerate the speed as well as the performance, we set gradient accumulation steps to 32 so that we can break GPU memory boundaries even with large batch sizes.

Layer	Hyperparameter	Value
Tokenizer	token max length	256
	truncation	"only first"
	padding	'max length'
Trainer	learning rate	1e-4
	train batch size	2
	eval batch size	2
	train epochs	1.0
	val check interval	0.2
	dropout rate	0.5
	gradient accumulation steps	32
Optimizer	type	AdamW
	lr	1e-4
	weight decay	0.01

Table 4: Hyperparameter Setting

Finally we choose AdamW as our optimizer and set the leraning rate and weight decay seperately to 1e-4 and 0.01.

4.5 Result

For all three tasks, table 5 shows the accuracy on two baseline pretrained models selected: ROBERTa-large and ELECTRA-large. As ELECTRA over performs ROBERTa sharply, we choose ELECTRA as the encoder of our eventual solution.

Pretrained Model	Task1	Task2	Task3
ROBERTa-large	64.47%	70.47%	68.47%
ELECTRA-large	85.89%	88.00%	89.06%

Table 5: Validation Accuracy Comparison For different pretrained models of all three tasks.

And then we implement MAMC and DUMA into the structure and do several experiments on each of the tasks and finally take an average accuracy to avoid unfairness. Table 6, 7, 8 shows the accuracy after applying the MAMC layer and DUMA layer. Both methods have a enhancement compared to only ELECTRA-large model in 3 tasks. On average over all tasks, the accuracy improves by 0.86% on MAMC and 3.00% on DUMA.

Model	Task1
ELECTRA-large	85.89%
ELECTRA-large + MAMC	87.08% (+1.19%)
ELECTRA-large + DUMA	89.95% (+4.06%)

Table 6: Validation Accuracy For task1 using MAMC and DUMA

Model	Task2
ELECTRA-large	88.00%
ELECTRA-large + MAMC	89.29% (+1.29%)
ELECTRA-large + DUMA	91.41% (+3.41%)

Table 7: Validation Accuracy For task2 using MAMC and DUMA

Model	Task3
ELECTRA-large	89.06%
ELECTRA-large + MAMC	89.18% (+0.12%)
ELECTRA-large + DUMA	90.59% (+1.53%)

Table 8: Validation Accuracy For task3 using MAMC and DUMA

5 Discussion

In this section, we consider and discuss on approach comparison and future developments.

5.1 Comparison of Approaches

As illustrated before, among our targeted tasks, task 1 focuses on evaluating the system's ability in understanding imperceptibility and task 2 aims to measuring the system's ability in comprehending nonspecificity. Meanwhile task 3 focuses on the model's transferability. Hence in task 3 we take the training data of task 1 to train a model and evaluate it with the task 2 evaluate data. Surprisingly, the result of task3 is even higher than task 1 and task 2. From those data We can clearly get a conclusion that the transferability of ELECTRA is great.
In Figure 9, we can know that ELECTRA + DUMA model performs the best in all three tasks.

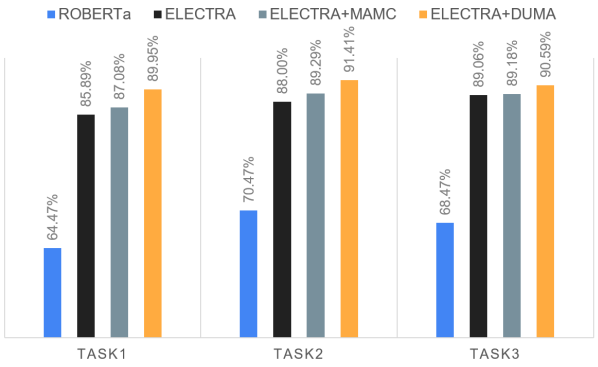

Figure 9: Comparison over tasks

Multi-choice Machine Reading Comprehension (MRC) requires a model to decide the correct answer from candidate options when given a passage and a corresponding question. Thus in addition to a powerful Pre-trained Language Model (PrLM) as encoder, multi-choice MRC especially relies on a matching network design which is supposed to effectively capture the relationships among the triplet of passage, question and answers. While the newer and more powerful PrLMs have shown their mightiness even without the support from a matching network, we implement the newly-proposed DUal Multi-head Co-Attention (DUMA) model, which is inspired by human's transposition thinking process solving the multi-choice MRC problem: respectively considering each other's focus from the standpoint of passage and question. The proposed DUMA has been shown effective and is capable of generally promoting PrLMs. Our proposed method is evaluated on two benchmark multi-choice MRC tasks, DREAM and RACE, showing that in terms of powerful PrLMs, DUMA can still boost the model to reach new state-of-the-art performance.

5.2 Future Works

In addition, when focusing on the result of task 3, we could notice that the high accuracy is due to the contribution of ELECTRA rather than MAMC or DUMA. In comparison with task 1 (+4.06%) and task 2 (+3.41%), we get lower enhancement of accuracy in task 3 (+1.53%). In order to tackle with this problem, we mainly proposed 3 procedures below for better generalization:

- **Data repartition** Data repartitioning (mix the train/dev sets, and randomly split into new train/dev sets by 8:2 or 9:1) aims to smooth the distribution difference among different train/dev data partition.
- **Data Augmentation** Augmenting the task data itself for fine-tuning, to mask different word than the original gold option (if there exists) using Task-adaptive Pretraining. The accurracy remains almost the same after adding the task augmented data. This suggests that our automatic augmentation method makes lower quality samples than the labelling data, while not too noisy that it can contribute to the robustness of the model.
- **Weight Averaging** Stochastic Weight Averaging (Izmailov et al., 2019) can be done across multiple checkpoints in the same run to get better generalization as well, so we also add this method to list.
- **Negative Augmentation** According to Chen et al., 2020, stronger negative samples will help the model learning with better performance. So we plan to generate some negative words using the pre-trained LMs to help train the models. Specifically, we replace the *@placeholder* with [MASK] to reconstruct the input and ask the BERT model to predict the word token at the [MASK]. These generated words are used as negative candidates.

6 Conclusion

Our system takes the large pre-trained LM ELECTRA with a multi-head multiple-choice clas-

sifier(MAMC) or Dual DUal Multi-head Co-Attention (DUMA) classifier on top. Firstly, we apply task-adaptive pretraining on different LMs (BERT, ROBERTa and ELECTRA) using CNN daily dataset to make the model fit the distribution in this domain. Secondly, We fine-tune them and compared the benchmark performance of different pre-trained LMs on the SemEval-2021 task 4, the result shows that ELECTRA outperforms other LMs in understanding abstractness of both the imperceptibility and nonspecificity. Therefore, we choose ELECTRA as our encoder and get a significant improvement on validation accuracy for about 20% . Thirdly, we try two kinds of on top classifiers, both MAMC and DUMA, to replace the original softmax classifier. MAMC would get an improvement on validation accuracy for about 1.24%. DUMA would get an improvement on validation accuracy for about 3.74%. Finally, we evaluate the generalization ability in task3. we find that Electra model have already performed well in generalization since it can captured global contextual sentence meaning. Our on top classifier improve the system's performance not so much than task1 and task2.

7 Division of Work

Here are seperate tasks for this project:

1. Build up and run task-adaptive models, such as BERT, ROBERTa, ELECTRA etc.
2. Data preprocessing.
3. Implement task-adaptive pre-train of BERT and ELECTRA on CNN daily.
4. Construct the system and do fine-tune training using ReCAM.
5. Apply on top classifier of MAMC.
6. Apply on top classifier of DUMA.
7. Get final Accuracy result and compare it with other papers.
8. Prepare for the presentation and final report.

The are two people in our group and the work division are as follows:

- **Jiachen Jiang**: 1, 2, 3, 4, 6, 7, 8.
- **Chengtian Zhang**: 1, 2, 3, 4, 5, 7, 8.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2016. Neural machine translation by jointly learning to align and translate.

Danqi Chen. 2018. *Neural Reading Comprehension and Beyond*. Ph.D. thesis. Copyright - Database copyright ProQuest LLC; ProQuest does not claim copyright in the individual underlying works; Last updated - 2021-09-28.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey E. Hinton. 2020. A simple framework for contrastive learning of visual representations. *CoRR*, abs/2002.05709.

Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. 2020. Don't stop pretraining: Adapt language models to domains and tasks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8342–8360, Online. Association for Computational Linguistics.

Wei He, Kai Liu, Jing Liu, Yajuan Lyu, Shiqi Zhao, Xinyan Xiao, Yuan Liu, Yizhong Wang, Hua Wu, Qiaoqiao She, Xuan Liu, Tian Wu, and Haifeng Wang. 2018. DuReader: a Chinese machine reading comprehension dataset from real-world applications. In *Proceedings of the Workshop on Machine Reading for Question Answering*, pages 37–46, Melbourne, Australia. Association for Computational Linguistics.

Karl Moritz Hermann, Tomáš Kočiský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend.

Nathan R. Hill, Samuel T. Fatoba, Jason L. Oke, Jennifer A. Hirst, Christopher A. O'Callaghan, Daniel S. Lasserson, and F. D. Richard Hobbs. 2016. Global prevalence of chronic kidney disease – a systematic review and meta-analysis. *PLOS ONE*, 11(7):1–18.

Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. 2019. Averaging weights leads to wider optima and better generalization.

Tomás Kociský, Jonathan Schwarz, Phil Blunsom, Chris Dyer, Karl Moritz Hermann, Gábor Melis, and Edward Grefenstette. 2017. The narrativeqa reading comprehension challenge. *CoRR*, abs/1712.07040.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach.

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. MS MARCO: A human generated machine reading comprehension dataset. *CoRR*, abs/1611.09268.

Takeshi Onishi, Hai Wang, Mohit Bansal, Kevin Gimpel, and David McAllester. 2016. Who did What: A Large-Scale Person-Centered Cloze Dataset. *arXiv e-prints*, page arXiv:1608.05457.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text.

Adam Trischler, Tong Wang, Xingdi Yuan, Justin Harris, Alessandro Sordoni, Philip Bachman, and Kaheer Suleman. 2017. NewsQA: A machine comprehension dataset. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*, pages 191–200, Vancouver, Canada. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need.

Boyuan Zheng, Xiaoyu Yang, Yu-Ping Ruan, Zhen-Hua Ling, Quan Liu, Si Wei, and Xiaodan Zhu. 2021. Semeval-2021 task 4: Reading comprehension of abstract meaning. *CoRR*, abs/2105.14879.

Pengfei Zhu, Hai Zhao, and Xiaoguang Li. 2020. Dual multi-head co-attention for multi-choice reading comprehension. *CoRR*, abs/2001.09415.