# Multiple Choice Reading Comprehension with Small Transformer Models

**Jiayue Guo, Bo Wu, Yilun Wu**
{jiayueg, bw1, yilunw}@andrew.cmu.edu
Language Technology Institute
Carnegie Mellon University
Pittsburgh, PA 15213

## Abstract

Multiple Choice Reading Comprehension is a language comprehension task where given an article and a question about the article, the machine must select the correct answer from a fixed number of options. There has been significant literature on using large transformer models to obtain accuracies of up to 90%, but very few work has been done for small and efficient models. We fill this gap by identifying some issues with using small models, and implement fixes to obtain state-of-the-art results for small models. Our models and experiments are based on ALBERT[1], DUMA[2], and end-to-end memory network[3], and it is trained and evaluated on the RACE [4] dataset. The code used to obtain our results can be found at `https://github.com/CanYouTeachMeHowToCode/Multiple-Choice-Reading-Comprehension-with-Small-Transformer-Models`.

## 1 Introduction

In recent years, machine reading comprehension (MRC) has become an increasingly popular and heated topic in NLP research. MRC is a fundamental and long-standing goal of the Question Answering (QA) task, which aims to teach machine to answer question automatically according to the content of the input passages/texts [5]. Reading comprehension is a challenging task for machines as it requires both understanding of natural language and knowledge in the context of different situations in our world.[6].

Early research on MRC focused on extracting the correct answers of certain relevant questions based on the content of the sample passage. These models are mostly trained on the SQuAD dataset, where the span of answer is marked in the passage, as seen in Figure 1. Such models could only handle the questions with expected answers in the form of a segment or short consecutive pieces of the corresponding passage. Multiple choice reading comprehension questions like those present in standardized tests such as TOEFL, SAT, and GRE typically require further reasoning or passage summarization as evident in 2, and show the insufficiency of previous models.

In this paper, we focus on the Multiple-choice Machine Reading Comprehension (MMRC) task, and utilize deep neural network models to figure out the correct answer of a reading comprehension question from several different choices based on the passage content. More specifically, we focus on improving the performance of MMRC using small models, as recent literature are all heavily focused on extremely large models, and neglects work on small models.

We first discuss the relevant research and models regarding Multiple-choice MRC. We then combine several of those ideas and pre-existing models into some new models that we test for our task. We

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

In meteorology, precipitation is any product of the condensation of atmospheric water vapor that falls under **gravity**. The main forms of precipitation include drizzle, rain, sleet, snow, **graupel** and hail... Precipitation forms as smaller droplets coalesce via collision with other rain drops or ice crystals **within a cloud**. Short, intense periods of rain in scattered locations are called "showers".

What causes precipitation to fall?
**gravity**

What is another main form of precipitation besides drizzle, rain, snow, sleet and hail?
**graupel**

Where do water droplets collide with ice crystals to form precipitation?
**within a cloud**

Figure 1: Question-answer pairs for a sample passage in the SQuAD dataset [6]

**Passage**: *Runners in a relay race pass a stick in one direction. However, merchants passed silk, gold, fruit, and glass along the Silk Road in more than one direction. They earned their living by traveling the famous Silk Road. ...* ***The Silk Road was made up of many routes, not one smooth path.*** *They passed through what are now 18 countries. The routes crossed mountains and deserts and had many dangers of hot sun, deep snow and even battles...*

**Question**: *The Silk Road became less important because _ .*

A. *it was made up of different routes*
B. *silk trading became less popular*
C. ***sea travel provided easier routes***
D. *people needed fewer foreign goods*

Figure 2: Example of a MC question that requires reasoning [5]

also add some fixes to the problems that happen when applying techniques for big models to small models.

## 2 Related Works

### 2.1 MRC Model Evolution & Categorization

Due to the limitation of rule-based models and the scarcity of well-formed datasets, early MRC models generally have poor performance. However, following the appearance of neural networks, especially attention mechanisms [7] and transformers, MRC models learn to capture more complicated relations between text. At present, the standard workflow for MRC is the extraction of the embeddings of the given tokenized inputs and the features from the embeddings through transformer models such as [8], ALBERT [1], GPT [9], T5 [10], and RoBERTa.[11]

One particular attribute of MRC is that it requires the interaction between question and context features to mimic the human reading behavior–people usually refer to passages to find relevant information when solving reading comprehension questions. Attention mechanism [7] is used in MRC to perform such interactions. Essentially, in MRC, there are two kinds of attention: unidirectional and bidirectional. Unidirectional readers attend passage (context) by question to extract the most relevant part of the context given the question. Modules that use unidirectional attention include Attentive Reader and Impatient Reader [12], where the module first extracts features using LSTM and attends them to get the final embedding. Bidirectional attention for MRC, on the other hand, also attends the question by the context to provide complementary information. Several models rely upon bidirectional attention. Attention-over-Attention Reader [13], sum over the row and column of attention scores in the first iteration and performs attention for the second time. Bi-DAF [14],

calculates context-to-query attention and query-to-context attention. Dual Multi-head Co-Attention (DUMA) [2] eparates the sequence into passage and question/answer parts and does co-attention among the two sequences to produce representations in awareness of the context. Dual Co-Matching Network for Multi-choice Reading Comprehension (DCMN) [5] separates the hidden sequence produced by the pre-trained model into hidden-state vectors for passage, question, and option, and then performs pair-wise attention.

According to how many times the module calculates attention scores between context and query, we can further categorize MRC models into one-hop interaction models and multi-hop interaction models. The models mentioned above (except Impatient Readers) are one-hop models. Examples of multi-hop models include the Multi-stage Multi-task learning framework for Multi-choice reading comprehension (MMM) [15], which employs a multi-step attention network, and End-To-End Memory Network Model [16], which applies recurrent attention.

### 2.2 State-of-the-art MRC Model

Researchers are now trying to incorporate knowledge-base into MRC models. Knowledge bases such as WordNet [17], Freebase [18], and Wikidata [19] introduce external knowledge that helps MRC models apprehend the semantic meanings of some entities in the passage and boost the performance.

Most state-of-the-art models for MRC fine-tune on pre-trained transformer-based models like BERT, RoBERTa, T5, and XLNet. There are two ways to fine-tune the model: 1) directly use the pooler output (hidden vector for [CLS] token) produced by the pre-trained model and linearly project the pooler output to learn the probability distribution for each option. 2) build additional components on top of the pre-trained model. Instead of simply using the pooler output, the extra components usually manipulate the entire sequence of hidden vectors produced by the pre-trained model and produce the probability distribution for each option. For example, DUMA separates the sequence into passage and question/answer sequences and does co-attention among the two sequences to produce representations aware of the context. As another example, DCMN separates the hidden sequences produced by the pre-trained model into hidden-state vectors for passage, question, and choice, and then performs pair-wise attention.

## 3 Task Specification

The specific NLP problem that we choose to tackle is Multi-choice Machine Reading Comprehension (MRC), which belongs to text-based QA. A passage $P$, a question $Q$ is proposed with a set of candidate answers $\{A_1, ..., A_i, ..., An\}$ is provided. Among all the answers, one is correct and we refer the reference answer as $A_r$. Our algorithm needs to select the correct answer with the information presented: correctly predicting $r$ from $(P, Q, \{A_1, \ldots, A_n\})$.

We also restrict the transformers we use to be small enough so that training takes a reasonable amount of time on only one gpu. For this reason, and just to investigate the viability of small models, we choose to use the base versions for our transformer backbones.

## 4 Model

We chose to adapt the models ALBERT, DUMA, sentence selection from DCMN, and memory network for use in our task. ALBERT is used as the backbone transformer to encode the sequence of inputs, and DUMA is added on top of ALBERT as a targeted post-processing layer for MMRC task. The latter two models are used to solve a problem in which the input gets prematurely truncated.

### 4.1 ALBERT Pre-trained Encoder

As mentioned in Section 2, one drawback of transformer-based models is that they have hundred of millions of parameters, which require tremendous computation power and huge memory. To tackle this problem, ALBERT [1] was proposed as a similar structured alternative to BERT. To reduce the number of parameters and make scale better, it employs the following tricks:

- cross-layer parameter sharing

  - ALBERT enables hyper-parameter sharing across different layers, which significantly reduce the number of parameters. The experiment result shows that although parameter sharing reduces the performance of model, the reduction in performance is not severe and can be easily compensated by increasing the size of the model. Also, parameter sharing can help stabilize training.
- factorized embedding parameterization
  - Traditionally, the dimension of the word embedding ($E$) is the same as the hidden dimension of the model in the following layers ($H$). This means that the word embedding matrix will be of size $O(VH)$. Since $V$ is large, the word embedding matrix will be large. ALBERT reduces the number of parameters in word embedding matrix by decoupling word embedding dimension from the hidden dimension (first project the one-hot vector for word into vector of dimension $E$, and then project it to a vector of dimension $H$). By doing this, the number of parameters become $O(VE + EH)$.

Apart from reducing the number of parameters, ALBERT changes the next-sentence-prediction(NSP) task of BERT to sentence-order-prediction (SOP). BERT generates negative samples by combining two segments from different corpus, thus confounding topic prediction and coherence prediction, and makes NSP an easy task. In contrast, ALBERT generates negative training samples by drawing two segments from the same corpus with their order switched, explicitly requiring model to focus or the coherence between texts.

### 4.2 DUMA Layer for Fine-tuning

Besides using ALBERT for encoding, we also choose a DUMA [2] layer to fine-tune the encoded sequences. The specific procedures of DUMA is listed below:

1. Separate the output representation from Encoder to obtain $E^P = [e_1^p, e_2^p, ..., e_{l_p}^p]$ and $E^QA = [e_1^{qa}, e_2^{qa}, ..., e_{l_{qa}}^{qa}]$, where $e_i^p, e_j^{qa}$ denote the *i*-th and *j*-th token representation of passage and question-answer respectively and $l_p$, $l_{qa}$ are the length.
2. Calculate the attention representations in a bi-directional way, that is, take 1) $E^P$ as ***Query***, $E^{QA}$ as ***Key*** and ***Value***, and 2) $E^P$ as ***Key*** and ***Value***, $E^{QA}$ as ***Query***. The exact formula is shown below:

$$\text{Attention}(E^P, E^{QA}, E^{QA}) = \text{softmax}\left(\frac{E^P(E^QA)^T}{\sqrt{d_k}}\right)E^{QA}$$
$$\text{head}_i = \text{Attention}(E^PW_i^Q, E^{QA}W_i^K, E^{QA}W_i^V)$$
$$\text{MHA}(E^P, E^{QA}, E^{QA}) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)W^O$$
$$\text{MHA}_1 = \text{MHA}(E^P, E^{QA}, E^{QA})$$
$$\text{MHA}_2 = \text{MHA}(E^{QA}, E^P, E^P)$$
$$\text{DUMA}(E^P, E^{QA}) = \text{Fuse}(\text{MHA}_1, \text{MHA}_2)$$

   , where $W_i^Q \in \mathbb{R}^{d_{model} \times d_q}$, $W_i^K \in \mathbb{R}^{d_{model} \times d_k}$, $W_i^V \in \mathbb{R}^{d_{model} \times d_v}$, $W_i^O \in \mathbb{R}^{hd_v} \times d_{model}$ are parameter matrices, $d_q$, $d_k$, $d_v$ denote the dimension of ***Query*** vectors, ***Key*** vectors and ***Value*** vectors, $h$ denotes the number of heads, MHA($\cdot$) denotes Multi-head Attention and DUMA($\cdot$) denotes our Dual Multi-head Co-Attention.The Fuse($\cdot$,$\cdot$) function first uses mean pooling to pool the sequence outputs of MHA($\cdot$), and then aggregates the two pooled outputs through a fusing method.
3. (Decoder part) Take the output of DUMA and computes the probability distribution over answer options. Denote $A_i$ as the *i*-th answer option, $O_i \in \mathbb{R}^l$ as the output of *i*-th $\langle P, Q, A_i \rangle$ triplet, and $A_r$ as the correct answer option, then the loss function is computed as:

$$O_i = DUMA(E^P, E^{QA_i})$$
$$L(A_r|P, Q) = -\log \frac{\exp(W^TO_r)}{\sum_{i=1}^{s} \exp(W^TO_i)}$$

   , where $W \in \mathbb{R}^l$ is a learnable parameter and $s$ denotes the number of candidate answer options.

### 4.3 End-To-End Memory Network for Summarization

One significant drawback of transformer model is that there is a fixed length limit on the input sequence during the tokenization process. One way to deal with this problem is to separate the passage into sentences and perform summarization over all the sentence embeddings.

In End-To-End Memory Network paper [16], the author proposes a recurrent attention network and several variations that will be able to perform soft sentence selection and passage summarization. The model performs attention in a recurrent manner and add residual connections among hops. The exact procedures is listed below:

1. Given a set of sentences $\{x_1, x_2, ..., x_i\}$, the model first converts the tokenized sentences to key vectors $\{k_i\}$ using a learnable embedding matrix $A$. It also uses another embedding matrix $C$ to produce a set of value vectors $\{v_i\}$ for the sentences.
2. For each question, an embedding matrix $B$ is used to convert the tokenized question into question embedding $q$, which serves as query.
3. The attention score $p_i$ is calculated as following:
$$p_i = Softmax(q^T k_i)$$
We then take a weighted sum over value vectors $\{v_i\}$ to get a new query embedding $q$:
$$q_{new} = q_{old} + \sum_i p_i * v_i$$
4. The procedure above is called a "hop," in every iteration, we recurrently performs attention to get an internal embedding, and we also add residual connection from the original query embedding to the internal embedding to produce the new query embedding for next iteration. The weights for all embedding matrices are shared across hops.

The End-To-End Memory Network module is ideal for summarizing the whole passage, since the attention mechanism instructs it to focus on specific sentences. The recurrent hierarchical structure produces a more fine-grained summarizing vector for the passage.

### 4.4 Sentence Selection using Cosine Similarity

Alternatively, instead of attempting at summarizing chunks of the article and preserving all the information, we could selectively choose important sentences to keep or delete unimportant sentences to alleviate the fixed length input problem. The DCMN paper uses sentence selection, and uses the following method:

1. Define $P$ the passage to be composed of $k$ sentences $p_i$: $P = [p_1, p_2, \ldots, p_k]$. $Q$ stays the same and define $A$ to be the concatenation of all the answers: $A = [A_i, \ldots, A_n]$.
2. For each sentence $p_i$, repeat steps 3~5 to obtain a score:
3. Encode $p_i$, $Q$, $A$ into word embedding sequences $H_{p_i}$, $H_Q$, $H_A$ using a pretrained transformer model. The length of $H_x$ is the same as the number of tokens in $x$.
4. Calculate cosine similarity between each pair of words between $(p_i, Q)$ and $(p_i, A)$:
$$D^{p_i q} = cosine(H_{p_i}, H_Q) \in R^{|p_i| \times |Q|}$$
$$D^{p_i a} = cosine(H_{p_i}, H_A) \in R^{|p_i| \times |A|}$$
5. Pool the similarity into a score:
$$score = mean(max(D^{p_i q}, dim = 0)) + mean(max(D^{p_i a}, dim = 0))$$
This first takes the max over all words in $p_i$, and then takes the mean over all words in $A$ or $Q$.
6. Retain in the article only sentences with the top $K$ scores.

Based on experimentation, we found that only selecting sentences until $K$ are left performed non-ideally and made the input length very inconsistent. Instead of following the exact method used by DCMN, we change the last step: we set a predefined maximum sequence length (max-seq-len) to our transformer, and delete sentences with the lowest scores until the article fits within max-seq-len. We perform experiments for varying max-seq-len.

# 5 Experiment

## 5.1 Dataset selection

In this paper, we choose RACE: Large-scale ReAding Comprehension Dataset From Examinations [4] as our target dataset for model evaluation and analysis.

### 5.1.1 RACE Dataset Introduction

RACE is a new large dataset for benchmark evaluation of MRC models. The main content of the dataset is the collection of the reading comprehension problems from English exams for middle school and high school Chinese students from 12 to 18 [4]. RACE dataset consists of near 28,000 passages and about 100,000 questions generated by English instructors (representative of human experts on reading comprehension tasks) [4]. Furthermore, this dataset covers a variety of topics range from news, story to ads, which are all carefully designed for evaluating the students' ability in reasoning and comprehension [2][4].

### 5.1.2 Reasoning Types of the Questions

The questions in the RACE dataset are classified into 5 classes as follows with ascending order of difficulty [4]:

1. Word matching: The question exactly matches a span in the article. The answer is self-evident.
2. Paraphrasing: The question is entailed or paraphrased by exactly one sentence in the passage. The answer can be extracted within the sentence.
3. Single-sentence reasoning: The answer could be inferred from a single sentence of the article by recognizing incomplete information or conceptual overlap.
4. Multi-sentence reasoning: The answer must be inferred from synthesizing information distributed across multiple sentences.
5. Insufficient/Ambiguous: The question has no answer or the answer is not unique based on the given passage.

The statistics of the proportion of questions with each reasoning type in RACE, CNN, SQuAD, and NewsQA based on 1000 samples per dataset is summarized in Table 1 below:

| Dataset | RACE-M | RACE-H | RACE-all | CNN | SQuAD | NewsQA |
|---|---|---|---|---|---|---|
| Word matching | 29.4% | 11.3% | 15.8% | 13.0% | 39.8% | 32.7% |
| Paraphrasing | 14.8% | 20.6% | 19.2% | 41.0% | 34.3% | 27.0% |
| Single-sentence reasoning | 31.3% | 34.1% | 33.4% | 19.0% | 8.6% | 13.2% |
| Multi-sentence reasoning | 22.6% | 26.9% | 25.8% | 2.0% | 11.9% | 20.7% |
| Insufficient/Ambiguous | 1.8 % | 7.1% | 5.8% | 25.0% | 5.4% | 6.4% |

Table 1: Statistics of Reasoning Type Proportions [4][6][20][21]

According to Table 1, the reasoning questions with higher difficulty level in the RACE dataset has higher proportion than that in other datasets in general. Therefore, RACE is more suitable for the evaluation of the Multiple-choice MRC models comparing to other datasets.

## 5.2 Evaluation Metric

Following suit with previous works, we evaluate the performance of our model by question answering accuracy. A question is answered correctly if the model prediction is same as the correct answer choice. The final accuracy is $N_{correct}/N_{total}$ over all of the test set.

### 5.3 Experimental Setting

For all experiments, we use albert-base-v2 as our backbone transformer, since this is the smallest transformer model offered by the ALBERT family. We then run four series of separate experiments for each of the models discussed above.

For the vanilla ALBERT model, we adopt a baseline from the paper. Then for more precise comparisons with the other models, we re-run baselines with different parameters. In one setting, we simply mean-pool the sequence of hidden vectors produces by ALBERT and feed it to the classifier. In the other setting, we feed the hidden vector of [CLS] token to the classifier instead of manipulating on the hidden sequence. We experiment with different maximum sequence lengths for ALBERT encoder to see its impact on the accuracy.

For DUMA experiments, we simply and add one DUMA layer on top of the ALBERT encoder. DUMA layer operates on the entire encoded sequence of hidden vectors instead of solely on the pooler output (hidden vector for [CLS] token). On the top of the model, five linear layers that share parameters are used in parallel, each followed by a dropout layer with dropout probability 0.5. We average the logits produced by the five linear classifiers to get the final logits. Cross-entropy loss is then applied on the logits.

For End-To-End Memory Network, during the tokenization process, we return all the truncated sequences for a single passage. Since each passage is of different length, the number of sequences for each sample is different. Due to the difference in number of sequences dimension, we set the batch size to one and set $accumulate\_grad\_batches$ to four. The initial learning rate is 2e-5. We clipped the gradient to 0.8 to prevent gradient explosion.

For sentence selection, we perform the selection process on the dataset prior to passing it through any models. The pretrained transformer used in the experiment is albert-base-v2, although using other models, or models from previously MMRC training, could yield better performances. After the dataset is processed, we pass it through vanilla ALBERT with different maximum sequence lengths.

During training, the initial learning rate is $2\mathrm{e}{-5}$ and the batch size is set to $4$. We train the model for 10 epochs in 6 hours. We use warm-up scheduler with warm-up proportion 0.1 to stabilize the training in the initial few epochs. We use FP16 mixed-precision training to speed up the process.

### 5.4 Results

| Model | Test accuracy | Notes |
|---|---|---|
| Baseline ALBERT$_{base}$ | 64.0 | result from [1] |
| ALBERT$_{base}$(hidden sequence) | 53.94 | max-seq-len=128 |
| ALBERT$_{base}$(hidden sequence) | 61.21 | max-seq-len=256 |
| ALBERT$_{base}$(hidden sequence) | 65.84 | max-seq-len=384 |
| ALBERT$_{base}$([CLS] pooler output) | 66.53 | max-seq-len=512 |
| ALBERT$_{base}$([CLS] pooler output) | **70.96** | max-seq-len=512, extensive HP tuning |

Table 2: Results of baseline vanilla ALBERT

| Model | max-seq-len | Baseline | +DUMA |
|---|---|---|---|
| ALBERT$_{base}$ | 128 | 53.94 | 54.41 |
| | 256 | 61.21 | 62.73 |
| | 384 | 65.84 | 67.93 |
| | 512 | 66.53 | 69.74 |

Table 3: Results of ALBERT$_{base}$+ DUMA

| Model | Test acc(RACE-all) | Notes |
|---|---|---|
| Baseline ALBERT$_{base}$ | 64.0 | result from [1] |
| ALBERT$_{base}$([CLS] pooler output) | 66.53 | max-seq-len=512 |
| E2E Memory Network (hidden sequence, 1 hops) | 60.68 | trunk-len=160 |
| E2E Memory Network (hidden sequence, 4 hops) | 53.38 | trunk-len=160 |
| E2E Memory Network (CLS token, 4 hops) | 63.64 | trunk-len=160 |

Table 4: Results of ALBERT$_{base}$+End2End Memory Network

| Model | max-seq-len | Baseline no selection | Sentence selection |
|---|---|---|---|
| ALBERT$_{base}$ | 128 | 53.94 | 60.40 |
| | 256 | 61.21 | 66.74 |
| | 384 | 65.84 | 69.54 |
| | 512 | 69.64 | 69.52 |

Table 5: Results of ALBERT$_{base}$+ Sentence Selection

# 6 Analysis

According to the results in Table 2, 3, 4, and 5, Our vanilla experiments performed at around the same level or slightly better than the baseline given by [1]. We therefore take our vanilla experiments as the new baseline to test other models. From those tables we see that adding the DUMA layer consistently outperformed the baseline at each maximum sequence length. The end-to-end memory network did not perform as expected, and we provide an error analysis as to the possible reasons. Sentence selection performed either around or above the baseline at different sequence lengths.

However, the best model out of all, to our surprise, was a vanilla ALBERT with extensive hyperparameter tuning. From this result, we expect that other models, with more hyperparameter tuning, could yield better results than those listed here. This is consistent with the results of [22], where AutoML tuning increased the accuracy by 1.7%. Due to resource limitations, we could not hyperparameter tune every single model. Hence we leave this for future works to investigate, and we analyze each individual model, without the hyperparameter tuning, below.

## 6.1 Error Analysis for End2End Memory Network

In our assignment 3 implementation, we stated that maximum sequence length seems to affect performance. In Table 3, 4, and 5 we see that the models with higher maximum length produced better results under all conditions. So we made the assumption that longer sequences contain more information that the model can use to its advantage, and shorter sequence length can sometimes discard critical information that makes it impossible to predict the answer at all. Based on this assumption, we made another assumption that models that take in all the contexts as inputs (typically summarization model) will inevitably out-perform models that take in truncated inputs. This motivates us to employ End2End Memory Network [16], which takes in the whole passage sentences embeddings and performs recurrent attention with question embedding. We also expected that with multiple hops, the model can achieve better performance than single hop, since during each attention iteration, the model should be able to learn more fine-grained embedding.

However, as shown in table 3, the inclusion of memory network with single hop decreases the performance. We speculate two reasons for this: one is that our initialization strategy changed the distribution of pre-trained ALBERT model during the initial phases of training. We use xavier normal initialization to initialize all the weight matrices in End2End Memory Network module and trained it end-2-end with the pretrained ALBERT model. This process might shift the distribution of pre-trained ALBERT model parameters, thus degrading the performance. We used the similar strategy when re-implementing ALBERT+DUMA; however, since it worked fine with DUMA, we did not pay attention to it when we incorporated memory network.

Also, End-To-End Memory Network only performed simple dot-product attention, and DUMA performed multi-head attention. Although DUMA losses information due to truncation, the multi-head attention mechanism compensates it by learning more features, which End-To-End Memory Network fails to do.

Another observation is that the increase in number of hops decreases the performance. Initially we planned to implement dropout in End-To-End Memory Network. However, since we are using parallel training in pyTorch lightning, we found that when dropout is included, we always got 'nan' result, so we removed the dropout layer. During the training, the training loss kept decreasing, so we suspect that the increase in hops also increases the chance of over-fitting.

### 6.2 Analysis for Sentence Selection

One interesting observation that we found during the evaluation of models with sentence selection is that sentence selection boosts the model performance more significantly when the maximum sequence length is small. Since more information will be lost by truncation for small maximum sequence length in these models, sentence selection can help model focus on the most relevant part in the remaining passage and gather more useful information for solving the questions.

Another interesting observation we have got when evaluating models with sentence selection is that the model performances are not differing so much when comparing models with sentence selection with maximum sequence length equals to 384 and that with maximum sequence length equals to 512. The primary reason of this phenomenon is that the maximum sequence length of 512 nearly covers all the content of many passages in the dataset, as shown in 3, and even though the maximum sequence length of 384 does not cover all the content of passages, the selected sentences from the passages are mostly the same for the two models. Therefore, the model performance of them are pretty similar.

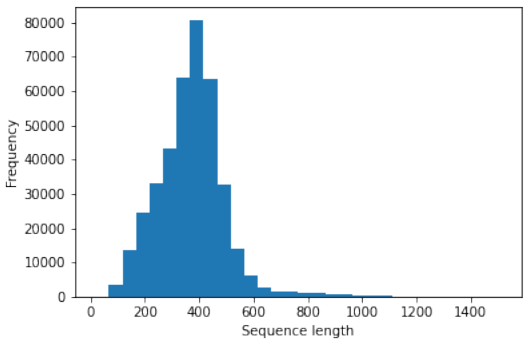


Figure 3: Lengths of the input sequences in RACE datasetLengths of the input sequences in RACE dataset

## References


[1] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. ALBERT: A lite BERT for self-supervised learning of language representations. *CoRR*, abs/1909.11942, 2019.

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

[3] Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. Weakly supervised memory networks. *CoRR*, abs/1503.08895, 2015.

[4] Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard H. Hovy. RACE: large-scale reading comprehension dataset from examinations. *CoRR*, abs/1704.04683, 2017.

[5] Shuailiang Zhang, Hai Zhao, Yuwei Wu, Zhuosheng Zhang, Xi Zhou, and Xiang Zhou. DCMN+: dual co-matching network for multi-choice reading comprehension. *CoRR*, abs/1908.11511, 2019.

[6] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100, 000+ questions for machine comprehension of text. *CoRR*, abs/1606.05250, 2016.

[7] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017.

[8] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018.

[9] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared 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 M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. *CoRR*, abs/2005.14165, 2020.

[10] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer, 2020.

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

[12] Karl Moritz Hermann, Tomás Kociský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. Teaching machines to read and comprehend. *CoRR*, abs/1506.03340, 2015.

[13] Yiming Cui, Zhipeng Chen, Si Wei, Shijin Wang, Ting Liu, and Guoping Hu. Attention-over-attention neural networks for reading comprehension. *CoRR*, abs/1607.04423, 2016.

[14] Min Joon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. Bidirectional attention flow for machine comprehension. *CoRR*, abs/1611.01603, 2016.

[15] Di Jin, Shuyang Gao, Jiun-Yu Kao, Tagyoung Chung, and Dilek Hakkani-Tür. MMM: multi-stage multi-task learning for multi-choice reading comprehension. *CoRR*, abs/1910.00458, 2019.

[16] Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. Weakly supervised memory networks. *CoRR*, abs/1503.08895, 2015.

[17] George A. Miller. WordNet: A lexical database for English. In *Human Language Technology: Proceedings of a Workshop held at Plainsboro, New Jersey, March 8-11, 1994*, 1994.

[18] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: A collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, SIGMOD '08, page 1247–1250, New York, NY, USA, 2008. Association for Computing Machinery.

[19] Denny Vrandečić and Markus Krötzsch. Wikidata: A free collaborative knowledgebase. *Commun. ACM*, 57(10):78–85, sep 2014.

[20] Adam Trischler, Tong Wang, Xingdi Yuan, Justin Harris, Alessandro Sordoni, Philip Bachman, and Kaheer Suleman. Newsqa: A machine comprehension dataset. *CoRR*, abs/1611.09830, 2016.

[21] Danqi Chen, Jason Bolton, and Christopher D. Manning. A thorough examination of the cnn/daily mail reading comprehension task. *CoRR*, abs/1606.02858, 2016.

[22] Yufan Jiang, Shuangzhi Wu, Jing Gong, Yahui Cheng, Peng Meng, Weiliang Lin, Zhibo Chen, and Mu Li. Improving machine reading comprehension with single-choice decision and transfer learning. *CoRR*, abs/2011.03292, 2020.