Learning to Distract: A Hierarchical Multi-Decoder Network for Automated Generation of Long Distractors for Multiple-Choice Questions for Reading Comprehension

Kaushal Kumar Maurya
Indian Institute Of Technology Hyderabad
Hyderabad, India
cs18resch11003@iith.ac.in

Maunendra Sankar Desarkar
Indian Institute Of Technology Hyderabad
Hyderabad, India
maunendra@iith.ac.in

ABSTRACT

The task of generating incorrect options for multiple-choice questions is termed as distractor generation problem. The task requires high cognitive skills and is extremely challenging to automate. Existing neural approaches for the task leverage encoder-decoder architecture to generate long distractors. However, in this process two critical points are ignored - firstly, the methods use Jaccard similarity over a pool of candidate distractors to sample the distractors. This often makes the generated distractors too obvious or not relevant to the question context. Secondly, some approaches did not consider the answer in the model, which caused the generated distractors to be either answer-revealing or semantically equivalent to the answer.

In this paper, we propose a novel Hierarchical Multi-Decoder Network (HMD-Net) consisting of one encoder and three decoders, where each decoder generates a single distractor. To overcome the first problem mentioned above, we include multiple decoders with a dis-similarity loss in the loss function. To address the second problem, we exploit richer interaction between the article, question, and answer with a SoftSel operation and a Gated Mechanism. This enables the generation of distractors that are in context with questions but semantically not equivalent to the answers. The proposed model outperformed all the previous approaches significantly in both automatic and manual evaluations. In addition, we also consider linguistic features and BERT contextual embedding with our base model which further push the model performance.

CCS CONCEPTS

• **Computing methodologies → Natural language generation.**

KEYWORDS

Natural language generation, Question-Answering, Distractor generation

ACM Reference Format:
Kaushal Kumar Maurya and Maunendra Sankar Desarkar. 2018. Learning to Distract: A Hierarchical Multi-Decoder Network for Automated Generation of Long Distractors for Multiple-Choice Questions for Reading Comprehension. In *Woodstock '18: ACM Symposium on Neural Gaze Detection, June 03–05, 2018, Woodstock, NY.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

Reading comprehension (RC) is recognized as an advanced cognitive task in Natural Language Processing (NLP), which involves both shallow and deep understanding of articles to carry out complex inferences. A person can demonstrate his/her understanding of an article by answering questions about the article. Multiple-Choice Questions (MCQ) from reading comprehension is a popular assessment technique to judge human understanding. It provides several advantages, including fast, unbiased, quick and consistent evaluation. In the classical convention, MCQ consists of a triplet: (1) question, (2) correct answer, and (3) *distractors* or the incorrect answers to confuse examinees [2]. Out of the three components, the creation of high-quality distractors is a important, challenging, and time-consuming task [28]. According to [6], 'ideal' distractors should be semantically related (but not semantically equivalent) to the answer and in the context of the question. Therefore, automation of distractor generation process is challenging but, at the same time, beneficial to target audiences. Our aim is to leverage modern deep learning models to generate long, grammatically correct and non-obvious distractors for MCQs.

In order to generate good distractors that are semantically correct but not equivalent to the correct answer, we try to understand how humans usually extract distractors. According to our understanding, humans generally follow a two-step generation process - (a) search for article sentences that are in context with the question and (b) avoid sentences which are semantically equivalent to the answer. The resultant sentences are potential candidates for distractor generation. Inspired by the human approach, we adopted a data-driven, sequence-to-sequence learning framework to address the problem of automated distractor generation. We propose a novel hierarchical multi-decoder network (**HMD-Net**). In HMD-Net, we first obtain the contextual word-level and sentence-level representations of the article by a hierarchical encoder. Word-level representations are learned for the question and answer. Then we use *SoftSel* operation and *Gated Mechanism* to capture rich semantic relations among these components. At the decoder side, we used three different decoders with a dis-similarity loss to generate three distractors. Decoders learn to generate three distractors from candidate sentence(s) in such a way that they are from the same context but are not too similar.

We carefully reviewed the generated text from HMD-Net and observed that there are sentences that have gender errors, morphological errors, etc. A few examples are *"She is good at solving maths."* where based on the context, the correct sentence should be: *"He is good at solving maths."* and *"Mr. Robert went last months."* where the correct sentence is: *"Mr. Robert went last month."* It shows that the model sometime fails to learn the linguistic properties of the word. To eliminate such problems, we externally included linguistic feature representation along with word representation in HMD-Net as used in the task of machine translation in [24]. Finally, to capture the contextual representation of words, we leveraged the popular BERT [3] model. We evaluated our system on two datasets RACE DG [5] and RACE++ DG dataset (prepared by us) across seven automatic (word-over lap based (BLEU, ROUGE, and METEOR), embedding based and BERT score based) metrics and two manual metrics (grammatical correctness and distractibility). Additionally, we checked how confusing the generated distractors are by performing the human assessment. Our extensive experimentation exhibits that our model consistently outperformed all the previous baselines. Our key contributions in this work are listed below:

(1) We propose a novel **H**ierarchical **M**ulti-**D**ecoder **Net**work (HMD-Net) to tackle the task of automated distractor generation. It is an end-to-end, data driven model to generate three distinct distractors from three decoders.

(2) We release a new high-quality distractor generation dataset RACE++ DG [1], prepared from RACE++ dataset by leveraging contextual similarities among the different components of the data instances by using a state-of-the-art representation learning algorithm (BERT). Our extensive experiments and analysis validate the quality of the dataset.

(3) We introduce a novel dis-similarity loss in HMD-Net for distractor generation and a new BERT cosine similarity (BERT-CS) based metric for automated evaluation.

2 RELATED WORK

The task of automated distractor generation (DG) is a part of multiple choice question (MCQ) generation. Traditionally, MCQ models use various approaches like similarity measures, ontology, embedding, etc. for selecting distractors [2]. However, in almost all cases, the granularity of generated distractors is limited to word-level or phrase-level. Due to recent advancements in deep learning, the generation of long and coherent distractors using learning-based approaches is gaining a lot of attention. A brief overview of traditional and deep learning-based approaches for automated distractor generation problem is given below:

• **Traditional Approaches:** Traditional methods for DG used linguistic resources like WordNet [18]and Thesaurus [26] for determining conceptual similarity, and used that information to generate the distractors. Later, linguistic properties like morphological and phonetic similarities [20], n-gram co-occurrence [8], context similarities [21], etc. were used for extracting distractors. More popular traditional approaches use embedding based similarities [7, 9] between text representations obtained using GloVe, word2vec, etc. A method proposed by [30] follows a two step-process: 1) compute the ranking of potential candidate texts by a weighted combination of different similarity metrics and 2) check the reliability of candidate distractors using contextual information.

• **Learning Based Approaches:** Early neural approaches for DG were oriented towards 'learning to rank' framework. Few popular approaches [14, 23] viewed the distractor generation problem as multi-class classification problem. The model proposed in [14] learns distractor-distribution conditioned on the question using generative adversarial nets (GANs). A method to generate distractors for fill-in-the-blank questions was proposed in [23]. In [13], the authors use feature-based ensemble and neural net-based models to rank distractors.

Two recent works close to our line of research are presented in [5] and [31]. The work in [5] focuses on static and dynamic attention from a hierarchical encoder-decoder model. Static attention helps to identify candidate sentences form the article and dynamic attention is then used to generate distractors. In an improvement over this, [31] exploits information across article and question using the co-attention mechanism. They apply Jaccard Similarity (JS) to sample three distractors over a pool of distractors generated by beam search. It is observed that because of Jaccard similarity, final distractors are different at the lexical level, but either they are not in context with the question or too obvious for end-user to eliminate. There were no precautions taken by [31] to ensure that generated distractors should not be answer-revealing or semantically-equivalent to the answer as they did not consider answer text in the model. Our novel framework fills these limitations and generates long, robust, and confusing distractors.

3 FRAMEWORK DESCRIPTION

3.1 Problem Statement

For the task of automated Distractor Generation (DG), we aim to generate long, coherent, and grammatically correct wrong options given a triplet <article, question, correct answer>. Generated distractors should be in the context with the question but should not be semantically equivalent to the answer. Formally, let $S =< s_1, s_2, \dots s_p >$ denote the input article with p sentences; Each sentence s_i is word sequence of length k i.e., $s_i =< w_{i,1}, w_{i,2}, \dots w_{i,k} >$. The question is denoted by Q and is a sequence of n words: $< q_1, q_2, \dots q_n >$. The word sequence $A =< a_1, a_2, \dots a_l >$ of length l denotes the answer. The three distractors are represented as $D_i =< d_{i,1}, d_{i,2}, \dots d_{i,u_i} >$ for $i = \{1, 2, 3\}$ where u_i is the length of i^{th} distractor. Our goal is to generate D_1, D_2 and D_3 given the triplet $<S, Q, A>$.

$$D_i = \arg\max_{\bar{D}_i} logP(\bar{D}_i | S, Q, A; \theta_i) \tag{1}$$

$P(\bar{D}_i | S, Q, A; \theta_i)$ is conditional log-likelihood of i^{th} distractor.

3.2 Model Overview

The standard sequence-to-sequence architecture for automated distractor generation suffers due to the large size of input article (RACE datasets have 342 tokens/article on average). On the other hand, through experimentations, we have seen that existing hierarchical

[1]the data and code of proposed model will be made publicly available for community

sequence-to-sequence models ([5, 31]) from literature are suitable for generating a single distractor and failed to generate multiple distractors correctly. In this paper, we propose an advancement over hierarchical sequence-to-sequence model and call it a Hierarchical Multi-Decoder Network (HMD-Net) that overcomes the limitations of the existing models.

As our goal is to generate three distractors, we view this problem as a one-to-many mapping problem ,i.e., single encoder for input triplet and three decoders for three distractors. At the **Encoder Side**, we first obtain the contextual word-level and sentence-level representations of the input triplet by the hierarchical encoder. Then we use *SoftSel* operation and *Gated Mechanism* to capture rich semantic relations among the components of the triplet. This semantic information is exploited to find relevance scores for article sentences. The scoring favors sentences that are in-context with the question but are not semantically equivalent to the correct answer. At the **Decoder Side**, we employed a multi-decoder model with a combination of cross-entropy and dis-similarity loss to generate three distractors. This novel architecture is trained in an end-to-end manner to generate high-quality distractors. The datasets used in this work contain less than three distractors for many questions, as it can be seen in Table 1. Moreover, any reference distractor is a ground truth for all the decoders. Hence, the training instances for the model are 4-tuples, each containing an article, a question, a correct answer and a distractor. We now present a detailed description of the components of our proposed HMD-Net architecture.

3.3 Distractor Hierarchical Encoder

In this section, we describe the different components of the encoder architecture of the proposed HMD-Net model. Figure 1 presents an architectural diagram of the model.

3.3.1 *Input Word Embedding.*

We obtain word embedding for each components of triplet in two different ways.

(1) We use pre-trained GloVe embedding to map tokens to vector representations. In addition, four linguistic feature ($f_1, \cdots f_4$) representations are concatenated. These features are: Parts of Speech tags, Named Entities, root form (lemma) of the word and Dependency Parsing Labels extracted from *Stanford CoreNLP* package.

- Sentence token embedding $se_{i,j} = [GloVe(w_{i,j}); f_{1_{i,j}}; f_{2_{i,j}}; f_{3_{i,j}}; f_{4_{i,j}}]$
- Question token embedding $qe_i = [GloVe(q_i); f_{1_i}; f_{2_i}; f_{3_i}; f_{4_i}]$
- Answer token embedding $ae_i = [GloVe(a_i); f_{1_i}; f_{2_i}; f_{3_i}; f_{4_i}]$

(2) BERT feature extraction method is used to obtain representation for each token. The output from BERT model is a representation of word-pieces which further aggregated (average pooling) to produce final token representation.

$se_{i,j} = BERT(w_{i,j})$, $qe_i = BERT(q_i)$ and $ae_i = BERT(a_i)$

3.3.2 *Article Word Encoder and Sentence Encoder.*

The initial token embeddings $se_{i,1}, se_{i,1}, \ldots, se_{i,k}$ of article sentence s_i are fed through a bidirectional Long Sort Term Memory (BiLSTM) network referred to as $LSTM^w$ to generate contextual representations of the tokens.

$$\overrightarrow{h^{enc}_{i,j}} = \overrightarrow{LSTM^w}(\overrightarrow{h^{enc}_{i,j-1}}, se_{i,j}) \quad \overleftarrow{h^{enc}_{i,j}} = \overleftarrow{LSTM^w}(\overleftarrow{h^{enc}_{i,j+1}}, se_{i,j}) \quad (2,3)$$

$\overrightarrow{h^{enc}_{i,j}}$ and $\overleftarrow{h^{enc}_{i,j}}$ are forward and backward hidden representations of $LSTM^w$. The final hidden state is $h^{enc}_{i,j} = [\overrightarrow{h^{enc}_{i,j}}; \overleftarrow{h^{enc}_{i,j}}]$. We denote h_p as the sequence of hidden states ($h^{enc}_{i,j}$) of all tokens of the article.

In order to represent each sentence s_i in the article we employed another bidirectional LSTM layer ($LSTM^s$) on top of word encoding layer. The inputs for $LSTM^s$ are the last token hidden representation of sentence s_i from $LSTM^w$ i.e., $h^{enc}_{i,k} = [\overrightarrow{h^{enc}_{i,k-1}}; \overleftarrow{h^{enc}_{i,k-1}}]$ and the first token hidden representation of sentence s_i from $LSTM^w$ i.e., $h^{enc}_{i,1} = [\overrightarrow{h^{enc}_{i,1}}; \overleftarrow{h^{enc}_{i,1}}]$. The final encoded contextual representation of a sentence is denoted as y_i. Now, the article can be represented as $y =< y_1, y_2, \ldots y_p >$. This completes the hierarchical structure of the encoder.

3.3.3 *Question Encoder and Answer Encoder.*

To determine contextual representations of the question and answer tokens, the initial token embeddings are fed through a bidirectional LSTM. This LSTM network is shared with article word-level LSTM.

$$h^q_i = [\overrightarrow{LSTM^w}(\overrightarrow{h^q_{i-1}}, qe_i); \overleftarrow{LSTM^w}(\overleftarrow{h^q_{i+1}}, qe_i)] \quad (4)$$

$$h^a_i = [\overrightarrow{LSTM^w}(\overrightarrow{h^a_{i-1}}, ae_i); \overleftarrow{LSTM^w}(\overleftarrow{h^a_{i+1}}, ae_i)] \quad (5)$$

The question and answer contextual representations are $h_q =< h^q_1, h^q_2, \ldots h^q_n >$ and $h_a =< h^a_1, h^a_2, \ldots h^a_l >$ respectively.

3.3.4 *SoftSel Operation.*

We investigate the effect of exploiting rich interactions among the word-level representations of the components of the triplet. It turns out that these interactions are helpful in finding potential candidate sentences of the article for DG. Such interactions can be achieved using SoftSel operation [27], which encodes the most relevant aspects of a sequence to another sequence. The input to *SoftSel* operation are two sequences, and output is an encoded sequence. The operation has three steps:

(1) **Cartesian Similarity:** For given two input sequences $h_1 \in \mathbb{R}^{r \times l_1}$ and $h_2 \in \mathbb{R}^{r \times l_2}$, a cartesian similarity $L \in \mathbb{R}^{l_1 \times l_2}$ is obtained across all possible states or words in h_1 and h_2.

$$L = h_1^T W^L h_2 \quad (6)$$

(2) **Row-wise Softmax:** To obtain distribution $\bar{L} \in \mathbb{R}^{l_2 \times 1}$ over cartesian similarity scores $softmax$ is applied on each row of L separately.

$$\bar{L} = \text{row-wise } softmax(L) \quad (7)$$

(3) **Weighted Sum :** Finally, a weighted sum of second sequence h_2 is encoded at given state j of first sequence h_1 and denoted as $\bar{h_{1j}} \in \mathbb{R}^{r \times 1}$. $\bar{h_{1j}}$ may be considered as the representation of j^{th} state of first sequence determined from the most influential parts of the second sequence for that state.

$$\bar{h_1} = h_2 \bar{L}^T \quad (8)$$

$W^L \in \mathbb{R}^{r \times r}$ is a weight matrix learned during the training process.

3.3.5 *Evidence Encoder.*

We leverage softsel operation to encode relatedness/similarity among the components of triplet, and term it as *evidence.*

Question-Evidence Encoder: First, we extract *evidence* between question and passage. More specifically, each state of question sequence is represented as the weighted sum of state representations of the article sequence h_p. The final encoded question

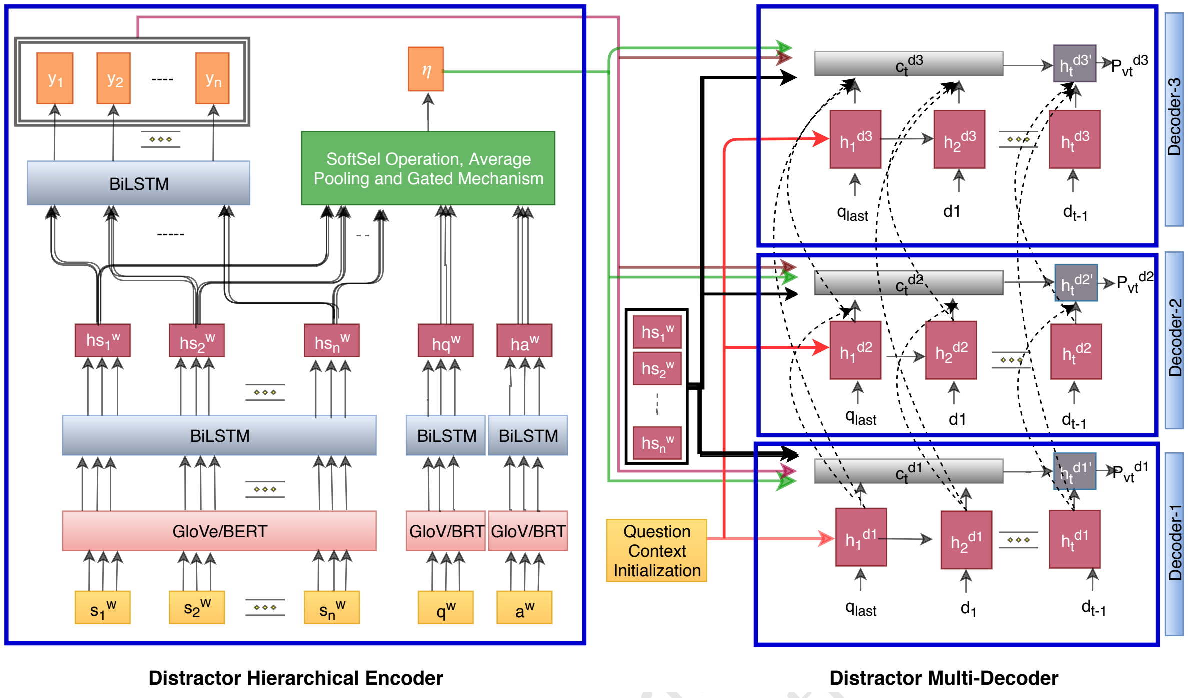


Figure 1: Architectural diagram of the proposed Hierarchical Multi-Decoder Model (*better viewed in color*)

sequence $\bar{h}_q$ can be viewed as a synthesized evidence vector known as the question-evidence encoder.

$$\bar{h}_q = SoftSel(h_q, h_p) \tag{9}$$

Answer-Evidence Encoder: Similar to question-evidence encoder another softsel operation is performed to obtain answer-evidence encoder $\bar{h}_a$.

$$\bar{h}_a = SoftSel(h_a, h_p) \tag{10}$$

Question-Answer-Evidence Encoder: It is two step process. We first apply softsel operation between question sequence h_q and answer sequence h_a to encode most relevant aspects of answer in $\bar{h}'_q$. Than, another softsel operation is applied between $\bar{h}'_q$ and article hidden sequence h_p to obtain question-answer-evidence representation $\bar{h}_{qa}$.

$$\bar{h}'_q = SoftSel(h_q, h_a) \qquad \bar{h}_{qa} = SoftSel(\bar{h}'_q, h_p) \tag{11, 12}$$

Answer-Question-Evidence Encoder: Similar to question-answer-evidence encoder another set of softsel operations are performed to obtain question-answer-evidence representation $\bar{h}_{aq}$.

$$\bar{h}'_a = SoftSel(h_a, h_q) \qquad \bar{h}_{aq} = SoftSel(\bar{h}'_a, h_p) \tag{13, 14}$$

Note that softsel operation is not symmetric. So the representations $\bar{h}_{qa}$ and $\bar{h}_{aq}$ are different. Computation of $\bar{h}_q$ detects question-relevant sentences from the article. Whereas, the other three evidence encoders are majorly oriented towards the answer and detect answer-relevant sentences in the article. This information is used in subsequent stages to ensure that the candidate sentences for distractor generation are not semantically equivalent to the answer.

3.3.6 *Gated Contextual and Evidence Encoding Layer.*

Inspired from the work on sequential variant of highway network [27], we adopted gated mechanism (*Gtd_Mchsm*) to control and encode information flow between contextual representation (i.e. from LSTM network) and evidence representation (i.e. from softsel operation). Let's define the gated mechanism which will result in a contextual-evidence representation $\bar{K} \in \mathbb{R}^{rxm}$ given contextual representation $C \in \mathbb{R}^{rxm}$ and evidence representation $E \in \mathbb{R}^{rxm}$ as given below:

$$z = \sigma(W^C C + W^E E + b) \tag{15}$$

$$\bar{K} = C * z + E * (1 - z) \tag{16}$$

Functionality of z is similar to that of the *reset gate* of RNNs. It determines what fraction of past knowledge (C) to forget, and what fraction to retain. $W^C, W^E \in \mathbb{R}^{rxr}$, $b \in \mathbb{R}^r$ are parameters.

Average pooling: We applied average-pooling operation to transform contextual representations (article sentence (s_i), question (q) and answer (a)) and evidence representations ($\bar{h_q}$, $\bar{h_a}$, $\bar{h}_{qa}$ and $\bar{h}_{aq}$) to fixed length vector representations.

$$\mathbf{s_i} = \frac{1}{k}\sum_{t=1}^{k} h^{enc}_{i,t}, \mathbf{q} = \frac{1}{n}\sum_{t=1}^{n} h^q_t, \mathbf{a} = \frac{1}{l}\sum_{t=1}^{l} h^a_t \tag{17}$$

$$\bar{\mathbf{h}}_\mathbf{q} = \frac{1}{n}\sum_{t=1}^{n} \bar{h}_{q_t}, \bar{\mathbf{h}}_\mathbf{a} = \frac{1}{l}\sum_{t=1}^{l} \bar{h}_{a_t}, \bar{\mathbf{h}}_\mathbf{qa} = \frac{1}{n}\sum_{t=1}^{n} \bar{h}_{qa_t}, \bar{\mathbf{h}}_\mathbf{aq} = \frac{1}{l}\sum_{t=1}^{l} \bar{h}_{aq_t}, \tag{18}$$

Notice that for article sentence representation, we do not use the contextual sentence representation y_i from the hierarchical model. Because, the idea is to exploit information of word-level representation $h^{enc}_{i,j}$ of sentences. The word-level representation contains more fine-grained information. The decoder of the model utilizes sentence-level representation. Finally, the gated mechanism is applied on different pairs of fixed vector representations:

$$q_F = Gtd\_Mchsm(\mathbf{q}, \bar{\mathbf{h}}_{\mathbf{q}}) \qquad a_F = Gtd\_Mchsm(\mathbf{a}, \bar{\mathbf{h}}_{\mathbf{a}}) \qquad (19, 20)$$

$$qa_F = Gtd\_Mchsm(\mathbf{a}, \bar{\mathbf{h}}_{\mathbf{qa}}) \quad aq_F = Gtd\_Mchsm(\mathbf{a}, \bar{\mathbf{h}}_{\mathbf{aq}}) \quad (21, 22)$$

Softsel Matching Score and Normalization Layer

Inspired from the way the humans extract distractors (see section 1), we derive a function to score the sentences according to their fitness for generating distractors.

$$m_i = \lambda_q s_i^T W_z q_F - \lambda_a (s_i^T W_z a_F + s_i^T W_z qa_F + s_i^T W_z aq_F) + b_z \quad (23)$$

Here, λ_q and λ_a are hyperparameters. $W_z \in \mathbb{R}^{rxr}$ and $b_z \in \mathbb{R}^r$ are learnable parameters. Similar to the approach of [5] we applied temperature τ to find final softsel matching score η. Intuitively, softsel matching score should be learned based on question property. If question can be answered using few sentences than distribution of scores for those sentences should be high. In other case, if question requires reasoning or summery of article then distribution should be inclined towards uniformity.

$$\tau = \sigma(W_q^T h_q + b_q) \qquad \eta_i = m_i / \tau \qquad (24, 25)$$

where σ is sigmoid activation function, h_q is contextual representation from word-level LSTM, W_q and b_q are parameters.

3.4 Distractor Hierarchical Multi-Decoder

We now discuss the different stages of our decoder network. Unlike previous works where a single decoder and Jaccard similarity over beam samples were used to generate three different distractors, we propose a novel multi-decoder model to generate distractors. We used three separate decoders (i.e., uni-directional LSTM networks) to generate three different distractors. Ideally, multiple decoders should not focus on the same word of a sentence to generate distractors, and at the same time, they should not consider non-relevant sentences of the article. We achieve this goal in two ways: a) during training, we learn the parameters of the second and third decoder in such a way that the generated distributions over candidate words should not be exactly similar to the distributions generated by the earlier decoders and neither be very different and b) as the ground truth for each decoder output is the same, we proposed a dis-similarity based loss function (we will provide mode detail in section 3.5) to learn appropriate distributions.

3.4.1 ***Question Context Initialization.*** To ensure that the decoders start with the context of the question, we use a separate uni-directional LSTM layer ($LSTM^{init}$) to encode the question and used the last hidden state of LSTM (as also done in [5]) in two ways. a) For each decoder, we use the last token of question q_{last} and b) employed final cell state c_n^{init} and hidden state h_n^{init} of LSTM to initialize each decoder. We further examined *whether using more than one last tokens of question improve the question context or not.* In our limited experiments, we found that in this setting, the generated distractors were biased towards the question. So there should be a trade-off between question context and the quality of generated distractor. Experimental results gave evidence that using the last token of questions works well.

3.4.2 ***Multi-Decoder Model.*** For a given decoder, at every decoding time step t we obtain two attention scores i.e., sentence-attention score $\beta_i^{d_k}$ and word-attention score $\alpha_{i,j}^{d_k}$ for article sentences and article words respectively. The superscript k indicates k^{th} decoder. The word-attention score is further used with softsel matching score η_i to obtain final attention distribution $\bar{\alpha}_{i,j}^{d_k}$ as we describe in Equation (32). Combining a softsel matching score is necessary because we need to de-emphasize the sentences and words that are not in the question-context or are answer-revealing. The idea of sentence and word-level attention is well studied as hierarchical attention model for text summarization task and is proven to be efficient [25]. We utilized article sentence representation y_i to compute sentence-attention score.

h_t^{d1}, h_t^{d2} and h_t^{d3} are states from three decoder LSTMs at any given time-step t. As discussed previously the learned distribution of different decoders should not be too same or too different; we achieve this goal in following way. The sentence and word level attentions for the first decoder are computed as:

$$\beta_i^{d_1} = y_i^T W_{d_1} h_t^{d_1} \qquad \alpha_{i,j}^{d_1} = h_{i,j}^{enc\,T} W_{d_1'} h_t^{d_1} \qquad (26, 27)$$

The attention scores for the second decoder are given by:

$$\beta_i^{d_2} = y_i^T W_{d_2} h_t^{d_2} - \lambda_{dist1} * y_i^T W_{d_2} h_t^{d_1}, \qquad (28)$$

$$\alpha_{i,j}^{d_2} = h_{i,j}^{enc\,T} W_{d_2'} h_t^{d_2} - \lambda_{dist1} * h_{i,j}^{enc\,T} W_{d_2'} h_t^{d_1} \qquad (29)$$

The second terms in Equations 28 and 29 try to move the the distributions away from the distributions learned by the first decoder. Similarly, the set of equations for the third decoder are given by:

$$\beta_i^{d_3} = y_i^T W_{d_3} h_t^{d_3} - \lambda_{dist1} * y_i^T W_{d_3} h_t^{d_1} - \lambda_{dist2} * y_i^T W_{d_3} h_t^{d_2}, \quad (30)$$

$$\alpha_{i,j}^{d_3} = h_{i,j}^{enc\,T} W_{d_3'} h_t^{d_3} - \lambda_{dist1} * h_{i,j}^{enc\,T} W_{d_3'} h_t^{d_1} - \lambda_{dist2} * h_{i,j}^{enc\,T} W_{d_3'} h_t^{d_2} \qquad (31)$$

Where W_{d_1}, $W_{d_1'}$, W_{d_2}, $W_{d_2'}$, W_{d_3} and $W_{d_3'}$ are learnable parameters. λ_{dist1} and λ_{dist1} are hyper-parameters which are fine-tuned. Finally, we combine the $\beta_i^{d_k}$ and $\alpha_{i,j}^{d_k}$ and softsel matching score η_i. We further normalize the score to obtain final word-level attention distribution $\bar{\alpha}_{i,j}^{d_k}$ across the article.

$$\bar{\alpha}_{i,j}^{d_k} = \frac{\alpha_{i,j}^{d_k} \beta_i^{d_k} \eta_i}{\sum_{i,j} \alpha_{i,j}^{d_k} \beta_i^{d_k} \eta_i} \qquad (32)$$

Then the context vector c_t^k is derived using attention-weighted additive operation over the word-level context representations of article words.

$$c_t^k = \sum_{i,j} \bar{\alpha}_{i,j}^{d_k} h_{i,j}^{enc} \qquad (33)$$

The distribution over given vocabulary words V at a time step t and for k^{th} decoder is computed as:

$$Pv_t^k = softmax(W_v tanh(W_{\bar{h}}[h_t^{d_k}; c_t^k]) + b_v) \qquad (34)$$

Where W_v, $W_{\bar{h}}$ and b_v are learnable parameters.

3.5 Training and Dis-Similarity Loss

As the ground truth distractor for each decoder is same, we need to be cautious that all decoders do not try to generate the same (ground truth) distractor. In an attempt to achieve this, the parameters of the model are already computed in such a way that the parameters (α, β values) of any decoder D_k are dependent on previous decoders $D_{1 \dots k-1}$. Towards this cause, additionally, we give an incentive to the model to learn slightly different distributions from the ground truth distractor during the learning process. We add a dis-similarity loss in the loss function to achieve this. The dis-similarity loss measures the distance between the ground truth distractor and the generated distractor. Hence, the loss function has two components: (a) cross-entropy loss and (b) dis-similarity loss, which are contrasting in nature. The impact of the dis-similarity loss is tuned using a parameter λ_{ds}. The effect of addition of the dis-similarity is analyzed with detailed evaluations in the Results Section. Dis-similarity loss is obtained in the following way:

(1) First, we feed ground truth distractor through uni-directional LSTM network where the last hidden step $h_{t-1}^{d_g}$ encodes the contextual representation of ground truth distractor.
(2) We also collected the last hidden state representation from each decoder i.e., $h_{t-1}^{d_1}$, $h_{t-1}^{d_2}$ and $h_{t-1}^{d_3}$.
(3) Finally, a cosine similarity score is computed across ground truth representation and final state of each decoder.

$$ds_i = cos(h_{t-1}^{d_g}, h_{t-1}^{d_i}) \tag{35}$$

where ds_i indicate similarity score with i^{th} decoder.

The final loss function to be minimized is given by:

$$L = \sum_{k=1}^{3} \left(- \sum_{D_k \in V} logP(D_k|S,Q,A;\theta_k) - \lambda_{ds} * (1 - ds_k) \right) \tag{36}$$

4 EXPERIMENTAL SETTING

4.1 Dataset

We used two distractor generation (DG) datasets to evaluate the performance of the proposed models: 1) RACE DG, and 2) RACE++ DG. RACE [11] is a popular reading comprehension dataset. It was collected from English examinations of the middle (called as RACE-M) and high-school (called as RACE-H) Chinese students. It consists 97,687 questions from 27,933 articles. RACE++ is an extension of the RACE dataset. RACE++ additionally includes 14,122 questions from 4,275 articles collected from college-level English examinations (called as RACE-C). In RACE dataset, each record is a 6-tuple containing article, question, correct answer and three distractors. It was observed that in RACE, there are distractors that do not have any semantic relevance with the article [5]. Then [5] used linguistic features and handcrafted rules to eliminate poor quality distractors. This resultant dataset is referred to as RACE-DG.

Designing an exhaustive set of rules is not a trivial task, and created rules may be biased in nature. Hence we investigate at the semantic level on the RACE++ data to prepare the RACE++ DG dataset containing good quality distractors. To find semantically relevant distractors, we applied the following methodology:

Parameters	RACE DG	RACE++ DG
Total no. of train samples	96501	135321
Total no. of dev samples	12089	16915
Total no. of test samples	12284	16915
Avg. article length (tokens)	342.0	342.3
Avg. question length	9.76	10.9
Avg. answer length	8.63	8.00
Avg. distractor length	8.48	7.68
Avg. sentences length (in article)	19.9	19.6
Avg. no. of distractors per triplet	2.1	2.3

Table 1: Statistics of RACE and RACE++ DG dataset. Average statistics are computed across all three samples

(1) We manually removed those distractors which are dependent on other distractors. For example, distractors like 'all of the above,' 'option a and option b are correct,' etc.

(2) Distractor, question, and answer should have a minimum length of three.

(3) Similar to [5], we also removed those questions that have fill in the blanks at the beginning or in the middle of the question.

(4) For an <article, question, correct answer, distractor> tuple, we found BERT representations for each of these components and also of that of the individual article sentences. We then computed the cosine similarity of the distractor with the question, correct answer, and each sentence of the article. The distractor is retained only if its cosine similarity with the question, correct answer, and the average cosine similarity over the article sentences - all of them were above a certain threshold.

We randomly divide RACE++ DG data into train (80%), test (10%) and validation (10%). For RACE DG data, this 8:1:1 split is already available. Final Statistics of the datasets can be seen in Table 1.

4.2 Methods Compared

We compared our model performance with following baselines:

- **Seq2Seq** [17]: It is standard encoder-decoder model with global attention mechanism. It consists of single LSTM in both the encoder and the decoder side.
- **HieRarchical Encoder-Decoder (HRED)** [25]: This is an advancement over the basic seq2seq model with global attention to handle large input. By construction, it is hierarchical to encode the input at word-level and sentence-level.
- **Hierarchical Static Attention (HSA)** [5]**:** This is similar to HRED but uses static and dynamic attentions instead of single global attention.
- **Hierarchical Co-Attention (HCA)** [31]**:** It is an improvement over the HSA model by exploiting rich interaction between article and question by co-attention model.
- **Static Attn + Multi-Dec (SMD) :** This is a variant of the proposed HMD-Net model. In employs static attention (as used in HSA [5]) instead of softsel and gated mechanism in the encoder side. We define this model to check the effectiveness of the multi-decoder model in the previous setting.
- **Encoder of HMD-Net + Decoder of HSA (EHMD+DHSA)** : We propose this model to verify the effectiveness of encoder of HMD-Net. The encoder of the model is that of the HMD-Net (utilizing softsel and gated contextual ideas) and decoder is similar

to the HSA model (single decoder that generates three distractors using beam search algorithm).

Additionally, three other models are implemented by adding linguistic features (LF) and BERT embeddings for each token in our HMD-Net framework. We term these models as HMD-Net+LF and HMD-Net+BERT, respectively. Further, we also want to observe the behavior of one of the above models with BERT, i.e., EHMD+DHSA+BERT.

4.3 Evaluation Metrics

We evaluated our model performance on seven automatic and three manual evaluation metrics. Previous models reported their scores for BLEU(1-4) [19] and ROUGE-L [15] automatic metrics only. These are word-overlap based metrics and may not reflect actual model performance. Several drawbacks of the BLEU scores have already been discussed in literature [1].

We aim to report the scores based on metrics that reflect the actual performance of the system and correlate with human judgments. To accomplish this, we additionally use lexical similarity-based metric METEOR [12], embedding based metrics [16], and BERT cosine similarity (BERT CS) metric. Unlike BLEU, METEOR leverages linguistic resources like Word-Net and root form of the word to compute the score. The three embedding based metrics are Greedy Matching [22], Embedding Average [29] and Vector Extrema [4]. Finally, we proposed a BERT-CS score, influenced by the recent work on BERT model [3]. We obtained the sentence representation from BERT for both generated and reference distractors and applied cosine similarity to compute the BERT-CS score. For manual evaluation we used *Grammatical correctness* and *Distractablity.* Additionally, we performed another human assessment to identify which method is generating a more confusing distractor. More confusing the distractors are, more better the model is.

4.4 Implementation Details

We modified the implementation of the OpenNMT toolkit [10] for our different models. For BERT based model we extracted the word feature from `bert-base-uncased` of dimension 768. For other models `GloVe.840B.300d` pre-trained word embedding is used. The number of layers for all the word encoders (either BiLSTM or uni-directional LSTM), including question-context initializer and target sentence encoder, is 2 and for sentence encoder is 1. All three decoders also have the number of layers as 2. We set 700 hidden size for both BiLSTM (350 for each direction) and uni-directional LSTMs. After several experiments on the validation set, the hyper-parameters $\lambda_q, \lambda_a, \lambda_{dist1}, \lambda_{dist2}$ and λ_{ds} are set as 0.5, -0.3, 0.5, 0.4 and 0.0001 respectively. Stochastic gradient descent (SGD) optimizer is initialized with learning rate 0.1 for all the models. Mini batch size is set to 16. We run the model for 200k steps. After 150k steps, the learning rate is halved at every 10k steps till the end. Additionally, we employed teacher-forcing. The maximum length of the generated distractor is set to 15. The beam size is set to 10. All the hyper-parameters are searched over validation dataset and results are reported on test dataset[2].

[2]readme file of code will provide more implementation details

5 RESULTS AND ANALYSIS

5.1 Automatic Evaluation Results

The automatic evaluation results of our proposed models are reported in Table-2 and Table-3 on RACE DG and RACE++ DG datasets respectively. The comparison with literature methods is presented for only RACE DG datasets, as the corresponding paper has the results for this dataset. The results are not available for RACE++ DG. It can be observed that HMD-Net outperformed the existing literature models as well our two baseline models (SMD and EHMD+DHSA) across all three distractors. Including linguistic features (LF) and BERT contextual embedding push the model performance even further. HMD-Net+BERT is noted as our best performing model whereas, EHMD+DHSA+BERT model was the second best model. We can observe that there is a significant performance gap between HMD-Net+LF and HMD-Net+BERT, which reveals the importance of contextual embedding. The better score of SMD over the HCA model across all three distractors acknowledges that the generation of distractors is suitable and effective in the multi-decoder setting. A higher score of EHMD+DHSA over the HSA model across all three distractors validates the impact of the stronger encoder. Note that the results for second and third distractors are also improved significantly as compared to previous approaches.

The results across METEOR, embedding based metrics, and BERT-CS, are consistent. These metrics give additional evidence that HMD-Net+BERT consistently outperforms all other models. The scores for embedding metrics are very close across different models. Considering this, we further investigate and obtain statistical significant bounds for each metric, which validated the correctness of results. Additionally, the very high BERT-CS scores (>0.81) confirm that generated distractors are semantically very close to reference distractors. The performance of all the models on the RACE++ DG dataset is better than that on the RACE DG dataset. This improvement in performances can be attributed to the facts that (a) RACE++ DG has more number of examples and (b) it contains quality distractors that help the models to learn better.

5.2 Human Evaluation Results

In human evaluation, we try to find answers to the following questions: a) *Which of the proposed models is performing the best?* and b) *What is the quality of the generated text?* Towards this, we performed two types of assessments: *comparative study* and *quantitative study.*

(1) **Comparative Study**: For this study, we employed 30 annotators (holding at-least master's degree in computer science and fluent in the English language). The annotators were distributed in three annotator-sets, each of size 10. From the RACE-DG test dataset, we randomly selected 120 questions from 40 articles (three questions from each article). To reduce bias and subjective evaluation, we gave all 120 questions to each annotator-set. Every annotator had to annotate 12 questions from 3 passages. To each annotator, along with passage and question, we provided four different distractors (as options) from our four models: SMD, HMD-Net, HMD-Net+LF, and HMD-Net+BERT. We asked annotators to select the closest correct answer. It was mentioned to the annotators that some questions might not have the correct answer; in that case, they had to select

	Models	BLEU-1	BLEU-2	BLEU-3	BLEU-4	ROUGE-L	METEOR	Embd Avg	G. Match	Ext.Score	BERT-CS
1st	Seq2Seq [17]	25.28	12.43	7.12	4.52	13.58	-	-	-	-	-
	HRED* [25]	27.96	14.41	9.05	6.35	14.68	-	-	-	-	-
	HSA* [5]	28.18	14.57	9.19	6.43	14.89	-	-	-	-	-
	HCA [31]	28.65	15.15	9.77	7.01	15.39	-	-	-	-	-
	EHMD+DHSA	28.25	14.52	9.34	6.66	24.03	10.76	0.569 ± 0.00006	2.530 ± 0.0004	0.357 ± 0.00005	0.813
	SMD	28.78	15.60	10.12	7.26	25.59	11.22	0.574 ± 0.0006	2.585 ± 0.0004	0.362 ± 0.00005	0.817
	HMD-Net	29.26	16.16	10.16	**7.66**	25.78	11.58	0.582 ± 0.00006	2.619 ± 0.0004	0.367 ± 0.00005	0.818
	HMD-Net+ LF	29.80	16.31	10.64	7.57	26.31	11.56	0.581 ± 0.00006	2.629 ± 0.0004	0.367 ± 0.00005	0.823
	EHMD+DHSA+BERT	29.44	16.02	10.06	6.6	25.04	11.08	0.586 ± 0.00005	2.610 ± 0.0004	0.364 ± 0.00005	0.823
	HMD-Net+ BERT	**30.99**	**17.30**	**11.09**	7.52	**26.50**	**12.07**	**0.591 ± 0.00005**	**2.667 ± 0.0004**	**0.370 ± 0.00005**	**0.823**
2nd	Seq2Seq [17]	25.13	12.02	6.56	3.93	13.20	-	-	-	-	-
	HRED* [25]	27.85	13.39	7.89	5.22	14.48	-	-	-	-	-
	HSA* [5]	27.85	13.41	7.87	5.17	14.41	-	-	-	-	-
	HCA [31]	27.29	13.57	8.19	5.51	14.85	-	-	-	-	-
	EHMD+DHSA	27.41	13.47	7.96	5.27	22.75	10.41	0.563 ± 0.00006	2.455 ± 0.0004	0.352 ± 0.00005	0.812
	SMD	28.17	14.62	8.96	6.00	24.15	10.82	0.570 ± 0.00006	2.519 ± 0.0004	0.355± 0.00005	0.814
	HMD-Net	28.84	15.06	9.29	6.37	24.79	11.15	0.580± 0.00006	2.591 ± 0.0004	0.364± 0.00005	0.818
	HMD-Net + LF	29.19	15.33	9.34	6.23	24.90	11.27	0.583 ± 0.00006	2.595 ± 0.0004	0.366 ± 0.00005	0.820
	EHMD+DHSA+BERT	30.16	15.9	9.68	6.19	24.05	11.29	0.583 ± 0.00005	2.535 ± 0.0003	0.359 ± 0.00004	0.823
	HMD-Net + BERT	**30.93**	**16.89**	**10.64**	**7.10**	**25.76**	**11.96**	**0.595 ± 0.00005**	**2.646 ± 0.0004**	**0.368 ± 0.00005**	**0.826**
3rd	Seq2Seq [17]	25.34	11.53	5.94	3.33	13.23	-	-	-	-	-
	HRED* [25]	26.73	12.55	7.21	4.58	14.86		-	-	-	-
	HSA* [5]	26.93	12.62	7.25	4.59	14.72	-	-	-	-	-
	HCA [31]	26.64	12.67	7.42	4.88	15.08	-	-	-	-	-
	EHMD+DHSA	26.93	12.97	7.32	4.56	22.31	10.29	0.560 ± 0.00005	2.416 ± 0.0003	0.352 ± 0.00005	0.811
	SMD	27.50	13.69	7.90	5.01	23.38	10.39	0.562 ± 0.00006	2.463 ± 0.0004	0.350 ± 0.00005	0.813
	HMD-Net	27.64	13.98	8.22	5.33	23.42	10.53	0.572 ± 0.00006	2.526 ± 0.0004	0.356 ± 0.00005	0.815
	HMD-Net + LF	29.09	14.64	8.63	5.60	24.63	10.99	0.580 ± 0.00005	2.540 ± 0.0004	0.360 ± 0.00005	0.819
	EHMD+DHSA+BERT	29.62	15.47	9.52	6.18	23.93	11.27	0.585 ± 0.00005	2.513 ± 0.0003	0.359 ± 0.00004	0.823
	HMD-Net + BERT	**29.70**	**15.95**	**9.74**	**6.21**	**24.91**	**11.37**	**0.584 ± 0.00005**	**2.614 ± 0.0004**	**0.363 ± 0.00005**	**0.824**

Table 2: Automatic evaluation results on the RACE-DG dataset. For Seq2Seq and HCA, the results are taken from [5]. For HRED and HSA (rows with *), the results are taken from [31] as these numbers are better than the numbers reported in the original paper [5] for the same dataset. Symbol (-) indicates that results are not available.

	Models	BLEU-1	BLEU-2	BLEU-3	BLEU-4	ROUGE-L	METEOR	Embd Avg	G. Match	Ext.Score	BERT-CS
1st	EHMD+DHSA	32.68	18.62	12.41	8.96	31.23	12.3	0.5713 ± 0.00005	2.4006 ± 0.0003	0.3649 ± 0.00004	0.8395
	SMD	33.18	18.45	11.43	7.36	**32.48**	12.53	0.5854 ± 0.00005	2.62 ± 0.0003	0.3770 ± 0.00004	0.8424
	HMD-Net	33.37	18.61	11.64	7.66	32.29	12.49	0.5877 ± 0.00005	2.6689 ± 0.0003	0.3766 ± 0.00004	0.8419
	HMD-Net+ LF	33.45	18.81	11.87	7.93	32.18	12.54	0.5826 ± 0.00005	2.6641 ± 0.0003	0.3762 ± 0.00004	0.8429
	EHMD+DHSA+BERT	33.57	19.38	12.79	8.96	31.81	12.44	0.5848 ± 0.00004	2.6624 ± 0.0003	0.3710 ± 0.00004	0.8444
	HMD-Net+ BERT	**34.58**	**20.26**	**13.54**	**9.66**	32.24	**12.85**	**0.5939 ± 0.00005**	**2.6904 ± 0.0003**	**0.3782 ± 0.00004**	**0.8452**
2nd	EHMD+DHSA	31.46	16.5	10.1	6.65	28.69	11.58	0.5656 ± 0.00005	2.3023 ± 0.0003	0.3540 ± 0.00004	0.8371
	SMD	32.42	17.29	10.36	6.54	30.41	12.09	0.5774 ± 0.00005	2.5257 ± 0.0003	0.3684 ± 0.00004	0.8422
	HMD-Net	33.99	17.62	10.42	6.45	30.49	12.23	0.5839+/- 0.00005	2.5756 ± 0.0003	0.3709 ± 0.00004	0.8423
	HMD-Net + LF	33.26	18.03	10.79	6.81	**31.01**	12.37	0.5846 ± 0.00005	2.6099 ± 0.0003	0.3763 ± 0.00004	0.8426
	EHMD+DHSA+BERT	33.47	18.83	12.28	8.5	29.51	12.21	0.5838 ± 0.00004	2.6248 ± 0.0002	0.3642 ± 0.00004	0.8425
	HMD-Net + BERT	**34.01**	**19.53**	**12.83**	**9.02**	30.86	**12.51**	**0.5953 ± 0.00004**	**2.6554 ± 0.0003**	**0.3763 ± 0.00004**	**0.8430**
3rd	EHMD+DHSA	31.27	15.85	9.38	6.05	27.67	11.49	0.5698 ± 0.00005	2.2752 ± 0.0002	0.3533 ± 0.00004	0.8362
	SMD	31.73	16.39	9.42	5.72	29.85	11.73	0.5794 ± 0.00005	2.483 ± 0.0003	0.3682 ± 0.00004	0.8376
	HMD-Net	32.14	16.67	9.55	5.69	29.75	11.95	0.5864 ± 0.00004	2.5196 ± 0.0003	0.3683 ± 0.00004	0.8369
	HMD-Net + LF	31.89	16.89	9.85	6.07	29.75	11.95	0.5736 ± 0.00005	2.5819 ± 0.0003	0.3653 ± 0.00004	0.8380
	EHMD+DHSA+BERT	33.26	18.59	12.05	8.32	29.12	12.14	0.5817 ± 0.00004	2.5675 ± 0.0002	0.3635 ± 0.00004	0.8401
	HMD-Net + BERT	**33.29**	**18.84**	**12.28**	**8.52**	**29.87**	**12.17**	**0.5881 ± 0.00005**	**2.6214 ± 0.0002**	**0.3690 ± 0.00004**	**0.8400**

Table 3: Automatic evaluation results of different models on RACE++ DG dataset.

Models	Annot-set1	Annot-set2	Annot-set3
SMD	24	27	25
HMD-Net	25	26	27
HMD-Net + LF	34	30	33
HMD-Net + BERT	37	37	35

Table 4: Comparative study results of human evaluation.

the closest option. We hypothesize that the distractors which are close to correct answer (selected by annotators) are more confusing. More confusing the distractors are, more better the model is. We intentionally did not expose the correct answers to evaluators, so that the evaluation should not be biased. Table 4 includes results of this study. Each entry in the table indicates the number of times the distractor generated by the model (in row) is selected as the correct answer by the annotator set (in column). Comparing across all three annotator-sets, it can be concluded that the HMD-Net+BERT model generated more confusing distractors. The second best-model is HMD-Net+LF. The performance ordering of the models is similar to the one obtained in automatic evaluation results.

	Models	Annot-set1	Annot-set2
GC	EHMD+DHSA	4.007	3.298
	SMD	4.058	3.894
	HMD-Net	3.780	3.747
	HMD-Net+LF	4.061	3.988
	EHMD+DHSA+BERT	4.054	**4.071**
	HMD-Net+BERT	**4.155**	3.982
DA	EHMD+DHSA	2.431	2.557
	SMD	2.567	2.457
	HMD-Net	2.522	2.491
	HMD-Net+LF	2.680	2.560
	EHMD+DHSA+BERT	2.661	**2.752**
	HMD-Net+BERT	**2.752**	2.634

Table 5: Extended quantitative study results of human evaluation. Where GC-Grammatical Correctness and DA- Distractablity

(2) **Quantitative Study**: To have a quantitative idea of the quality of generated distractors, we asked each annotator to rate the generated distractors on a scale of 1-to-5 (1 is very poor and 5 is very good) on two manual evaluation metrics: (a) **Grammatical correctness**- *how grammatically correct the distractors are?* and (b) **Distractability**- *how confusing the distractors are?* As the Quantitative Study will provide absolute evaluation scores, we conducted this on large dataset and 6 models. We randomly selected 350 questions from 117 passages. We employed two sets of annotators (holding at-least master's degree in computer science and fluent in English language). Each set of annotators had to evaluate all 350 questions. Due to more data and more number of models in this task as compared to the comparative study, the workload on the volunteers were more for this task. There were lesser number of volunteers for this evaluation. We divided them into two sets. Each annotator had to annotate 50 questions. In this study we randomly selected one distractor each from EHMD+DHSA, SMD, HMD-Net, HMD-Net+LF, EHMD+DHSA+BERT and HMD-Net+BERT models. Outcomes of the study can be found in Table-5. Grammatical Correctness metric received high score over distractability and HMD-Net+BERT was best performing model. Considering the fact that the current performance ceiling of humans on the RACE dataset is 95% [11] (for identification of correct answer given article, question and four options), confusing humans is a challenging task. Considering these factors, we can conclude that our model performed decently on distractability aspect.

5.3 Ablation Study and Inter-distractor Similarity Test

To verify the effect of each component of the proposed HMD-Net, we performed an ablation study. The results on first distractor can be seen in Table-7. It is observed that the evidence encoding layer, dis-similarity loss and gated contextual representation are key components of the model. The removal of these components resulted in lower performance. Using the last two tokens of question sentence - diverts the model training and model performed worst under this setting. The possible explanation can be - including a larger context may disturb distractor sentence structure and the model gets confused in learning critical patterns. The CER, h_{aq} and h_{qa}) have minor impact on the model.

It is expected that generated distractors should be semantically related to each other. If all the generated distractors are similar on a lexical level, then all the metrics used above will report a high score for all the distractors, but effectively only one distractor is generated. This error is hard to catch until some careful analysis is performed. To nullify this kind of situation and provide the evidence that our generated distractors are different at the lexical level too, we performed an additional experiment on the RACE-DG dataset. For each pair of generated distractors, we computed the Jaccard Similarity (JS). The results are reported in Table-6. Note that the maximum similarity was 0.264 on the BERT model, which is very low. Previously, [5] and [31] used JS threshold as 0.5 while selecting three distractors from the pool of distractors generated by the beam search algorithm. This gives evidence that our model generates lexically distinct distractors.

5.4 Case Study

Figure-2 includes a sample distractor generated from the HMD-Net model. In the middle, we plotted the distribution of the softsel score. We can observe that sentences 4, 7, and 8 are potential candidates for distractor generation and received high scores. Sentence 2 consists correct answer hence received the low score. The three ground truth and generated distractors are shown on the top right side of the figure. In the bottom right, final learned attention ($\bar{\alpha}_{i,j}^{d_k}$) is included for all three decoders at decoding time step t_2 i.e. after generating word *'Because'*. For generating the next word, decoder1 selected sentence4, decoder2 selected sentence6 and decoder3 selected sentence7 at time step t_2. Each decoder obtains word distribution for the selected sentence and accordingly generates the next word.

6 CONCLUSION

In this paper, we have presented a Hierarchical Multi-Decoder Network (HMD-Net) and its variants with linguistic features and BERT contextual embedding. It is a data-driven approach to generate long and high-quality distractors for reading comprehension. We exploited the rich interaction among question, answer and passage using SoftSel operation and Gated Mechanism at the encoder side. At the decoder side, we used three separate decoders to generate three distractors. The distractors should not be exactly same or very different. We also prepared a high-quality new distractor generation dataset. Our model outperformed existing methods from literature and our own baselines on automated and manual evaluation metrics. Extending the proposed model directly to generate arbitrary number of distractors will be costly. So, we would like to develop an approach where any number of in-context and non-answer-revealing distractors can be generated using a single decoder.

REFERENCES

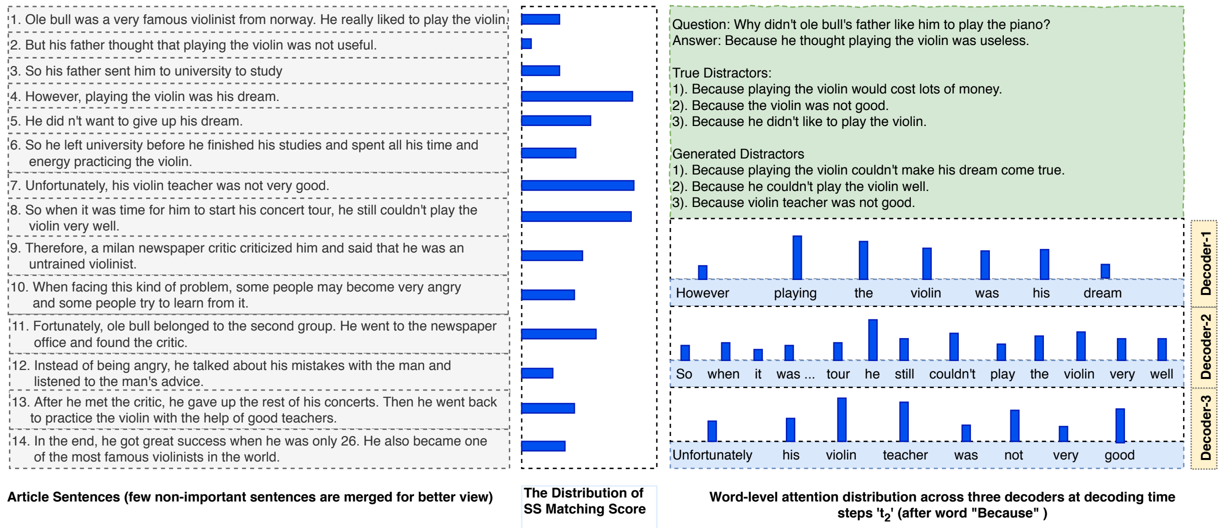
[1] Chris Callison-Burch, Miles Osborne, and Philipp Koehn. 2006. Re-evaluation the role of bleu in machine translation research. In *11th Conference of the European Chapter of the Association for Computational Linguistics.*

Models	Dist1-Dist2	Dist1-Dist3	Dist2-Dist3
SMD	0.200	0.191	0.216
HMD-Net	0.221	0.210	0.236
HMD-Net + LF	0.215	0.219	0.201
HMD-Net + BERT	0.264	0.251	0.246

Table 6: Average jaccard similarity scores across generated three distractors on RACE DG dataset.

Models	BLEU-1	BLEU-2	BLEU-3	BLEU-4	ROUGE-L	METEOR
Full HMD-Net Model	29.26	16.16	10.16	7.66	25.78	11.58
Without EEL (with CR)	28.60	15.17	9.68	6.90	25.15	10.96
Without CR (with EEL)	29.15	15.71	10.20	7.27	25.62	11.40
*Without CER	28.86	15.46	9.96	7.15	25.38	11.11
Without h_aq & h_qa	28.92	15.89	10.39	7.50	25.56	11.44
Without DSL	28.35	15.24	9.91	7.05	25.39	10.96
With last two Question tokens in QCI	20.90	9.38	5.45	3.50	17.45	8.24

Table 7: Ablation study results of the first distractor on the RACE DG dataset. Where EEL: Evidence Encoding Layer, CR: contextual representation, CER: contextual evidence representation (gated mechanism output), DSL: dis-similarity loss and QCI: question Context Initialization. The row with * indicate that -the results are obtained without CER where SS matching score is obtained from the average of CR and EEL scores


1. Ole bull was a very famous violinist from norway. He really liked to play the violin.
2. But his father thought that playing the violin was not useful.
3. So his father sent him to university to study
4. However, playing the violin was his dream.
5. He did n't want to give up his dream.
6. So he left university before he finished his studies and spent all his time and energy practicing the violin.
7. Unfortunately, his violin teacher was not very good.
8. So when it was time for him to start his concert tour, he still couldn't play the violin very well.
9. Therefore, a milan newspaper critic criticized him and said that he was an untrained violinist.
10. When facing this kind of problem, some people may become very angry and some people try to learn from it.
11. Fortunately, ole bull belonged to the second group. He went to the newspaper office and found the critic.
12. Instead of being angry, he talked about his mistakes with the man and listened to the man's advice.
13. After he met the critic, he gave up the rest of his concerts. Then he went back to practice the violin with the help of good teachers.
14. In the end, he got great success when he was only 26. He also became one of the most famous violinists in the world.

Question: Why didn't ole bull's father like him to play the piano?
Answer: Because he thought playing the violin was useless.

True Distractors:
1). Because playing the violin would cost lots of money.
2). Because the violin was not good.
3). Because he didn't like to play the violin.

Generated Distractors
1). Because playing the violin couldn't make his dream come true.
2). Because he couldn't play the violin well.
3). Because violin teacher was not good.

However playing the violin was his dream
So when it was ... tour he still couldn't play the violin very well
Unfortunately his violin teacher was not very good
Decoder-1
Decoder-2
Decoder-3
Article Sentences (few non-important sentences are merged for better view)
The Distribution of SS Matching Score
Word-level attention distribution across three decoders at decoding time steps 't2' (after word "Because")

Figure 2: A sample of generated distractors from HMD-Net model on RACE DG dataset.

[2] Dhawaleswar Rao Ch and Sujan Kumar Saha. 2018. Automatic Multiple Choice Question Generation from Text: A Survey. *IEEE Transactions on Learning Technologies* (2018).
[3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. [n.d.]. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of 2019 Conference of North American Chapter of the Association for Computational Linguistics: Human Language Technologies, year = 2019, pages =*.
[4] Gabriel Forgues, Joelle Pineau, Jean-Marie Larchevêque, and Réal Tremblay. 2014. Bootstrapping dialog systems with word embeddings. In *Nips, modern machine learning and natural language processing workshop*, Vol. 2.
[5] Yifan Gao, Lidong Bing, Piji Li, Irwin King, and Michael R Lyu. 2019. Generating distractors for reading comprehension questions from real examinations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 6423–6430.
[6] Hubbard C Goodrich. 1977. Distractor efficiency in foreign language testing. *Tesol Quarterly* (1977), 69–78.
[7] Qi Guo, Chinmay Kulkarni, Aniket Kittur, Jeffrey P Bigham, and Emma Brunskill. 2016. Questimator: Generating knowledge assessments for arbitrary topics. In *IJCAI-16: Proceedings of the AAAI Twenty-Fifth International Joint Conference on Artificial Intelligence*.
[8] Jennifer Hill and Rahul Simha. 2016. Automatic generation of context-based fill-in-the-blank exercises using co-occurrence likelihoods and Google n-grams. In *Proceedings of the 11th Workshop on Innovative Use of NLP for Building Educational Applications*. 23–30.
[9] Shu Jiang and John Lee. 2017. Distractor generation for chinese fill-in-the-blank items. In *Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications*. 143–148.
[10] Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander Rush. 2017. OpenNMT: Open-Source Toolkit for Neural Machine Translation. In *Proceedings of ACL 2017, System Demonstrations*. Vancouver, Canada, 67–72.
[11] 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*. Copenhagen, Denmark, 785–794.
[12] Alon Lavie and Michael J Denkowski. 2009. The METEOR metric for automatic evaluation of machine translation. *Machine translation* 23, 2-3 (2009), 105–115.
[13] Chen Liang, Xiao Yang, Neisarg Dave, Drew Wham, Bart Pursel, and C Lee Giles. 2018. Distractor generation for multiple choice questions using learning to rank. In *Proceedings of the thirteenth workshop on innovative use of NLP for building educational applications*. 284–290.
[14] Chen Liang, Xiao Yang, Drew Wham, Bart Pursel, Rebecca Passonneaur, and C Lee Giles. 2017. Distractor generation with generative adversarial nets for automatically creating fill-in-the-blank questions. In *Proceedings of the Knowledge Capture Conference*. ACM, 33.
[15] Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*. 74–81.
[16] Chia-Wei Liu, Ryan Lowe, Iulian Serban, Mike Noseworthy, Laurent Charlin, and Joelle Pineau. 2016. How NOT To Evaluate Your Dialogue System: An Empirical Study of Unsupervised Evaluation Metrics for Dialogue Response Generation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Austin, Texas, 2122–2132.
[17] Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective Approaches to Attention-based Neural Machine Translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal, 1412–1421.
[18] Ruslan Mitkov et al. 2003. Computer-aided generation of multiple-choice tests. In *Proceedings of the HLT-NAACL 03 workshop on Building educational applications using natural language processing*. 17–22.
[19] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*. 311–318.
[20] Juan Pino and Maxine Eskenazi. 2009. Semi-automatic generation of cloze question distractors effect of students' L1. In *International Workshop on Speech and Language Technology in Education*.
[21] Juan Pino, Michael Heilman, and Maxine Eskenazi. 2008. A selection strategy to improve cloze question quality. In *Proceedings of the Workshop on Intelligent Tutoring Systems for Ill-Defined Domains. 9th International Conference on Intelligent Tutoring Systems, Montreal, Canada*. 22–32.
[22] Vasile Rus and Mihai Lintean. 2012. A comparison of greedy and optimal assessment of natural language student input using word-to-word similarity metrics. In *Proceedings of the Seventh Workshop on Building Educational Applications Using NLP*. 157–162.

[23] Keisuke Sakaguchi, Yuki Arase, and Mamoru Komachi. 2013. Discriminative approach to fill-in-the-blank quiz generation for language learners. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2)*. 238–242.

[24] Rico Sennrich and Barry Haddow. 2016. Linguistic Input Features Improve Neural Machine Translation. In *Proceedings of the First Conference on Machine Translation: Volume 1, Research Papers*. Berlin, Germany, 83–91.

[25] Alessandro Sordoni, Yoshua Bengio, Hossein Vahabi, Christina Lioma, Jakob Grue Simonsen, and Jian-Yun Nie. 2015. A Hierarchical Recurrent Encoder-Decoder for Generative Context-Aware Query Suggestion *(CIKM '15)*. Association for Computing Machinery, New York, NY, USA, 553–562.

[26] Eiichiro Sumita, Fumiaki Sugaya, and Seiichi Yamamoto. 2005. Measuring non-native speakers' proficiency of English by using a test with automatically-generated fill-in-the-blank questions. In *Proceedings of the second workshop on Building Educational Applications Using NLP*. 61–68.

[27] Min Tang, Jiaran Cai, and Hankz Hankui Zhuo. 2019. Multi-Matching Network for Multiple Choice Reading Comprehension. *Proceddings of the AAAI, Honolulu* (2019).

[28] Johannes Welbl, Nelson F. Liu, and Matt Gardner. 2017. Crowdsourcing Multiple Choice Science Questions. In *Proceedings of the 3rd Workshop on Noisy User-generated Text*. Copenhagen, Denmark, 94–106.

[29] John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2016. Towards Universal Paraphrastic Sentence Embeddings. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, Yoshua Bengio and Yann LeCun (Eds.).

[30] Torsten Zesch and Oren Melamud. 2014. Automatic generation of challenging distractors using context-sensitive inference rules. In *Proceedings of the Ninth Workshop on Innovative Use of NLP for Building Educational Applications*. 143–148.

[31] Xiaorui Zhou, Senlin Luo, and Yunfang Wu. 2019. Co-Attention Hierarchical Network: Generating Coherent Long Distractors for Reading Comprehension. *arXiv preprint arXiv:1911.08648* (2019).