

# Look at the First Sentence: Position Bias in Question Answering

Miyoung Ko Jinhyuk Lee<sup>†</sup> Hyunjae Kim Gangwoo Kim Jaewoo Kang<sup>†</sup>  
Korea University

{miyoungko, jinhyuk\_lee, hyunjae-kim}@korea.ac.kr  
{gangwoo\_kim, kangj}@korea.ac.kr

## Abstract

Many extractive question answering models are trained to predict start and end positions of answers. The choice of predicting answers as positions is mainly due to its simplicity and effectiveness. In this study, we hypothesize that when the distribution of the answer positions is highly skewed in the training set (e.g., answers lie only in the  $k$ -th sentence of each passage), QA models predicting answers as positions can learn spurious positional cues and fail to give answers in different positions. We first illustrate this *position bias* in popular extractive QA models such as BiDAF and BERT and thoroughly examine how position bias propagates through each layer of BERT. To safely deliver position information without position bias, we train models with various de-biasing methods including entropy regularization and bias ensembling. Among them, we found that using the prior distribution of answer positions as a bias model is very effective at reducing position bias, recovering the performance of BERT from 37.48% to 81.64% when trained on a biased SQuAD dataset.

## 1 Introduction

Question answering (QA) is a task of answering questions given a passage. Large-scale QA datasets have attracted many researchers to build effective QA models, and with the advent of deep learning, recent QA models are known to outperform humans in some datasets (Rajpurkar et al., 2016; Devlin et al., 2019; Yang et al., 2019). Extractive QA is the task that assumes that answers always lie in the passage. Based on this task assumption, various QA models are trained to predict the start and end positions as the answers. Following the

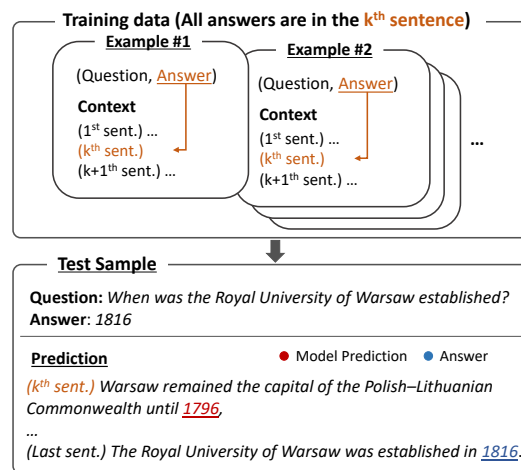


Figure 1: Example of position bias. BERT trained on the dataset with a skewed answer position distribution, provides wrong predictions, biased to the specific sentence position.

structure of earlier deep learning-based QA models (Wang and Jiang, 2016; Seo et al., 2017; Xiong et al., 2017), recent QA models provide positions of answers without much consideration (Yu et al., 2018; Devlin et al., 2019; Yang et al., 2019).

The popularity of predicting the answer positions is credited to the fact that it reduces the prediction space to  $\mathcal{O}(n)$  where  $n$  is the length of an input document. It is more efficient and effective than directly generating answers from a large vocabulary space. Furthermore, it reduces the QA task to a classification task which is convenient to model. Nevertheless, very few studies have discussed the side effects of predicting the answer positions. Could there be any unwanted biases when using answer positions as prediction targets?

In this paper, we demonstrate that the models predicting the position can be severely biased when trained on datasets that have a very skewed answer position distribution. We define this as position bias as shown in Figure 1. Models trained on a biased dataset where answers always lie in the same

<sup>†</sup>Corresponding authors

Training Data	BiDAF			BERT			XLNet		
	EM	F1	$\Delta$	EM	F1	$\Delta$	EM	F1	$\Delta$
SQuAD <sub>train</sub>	66.51	76.46		81.32	88.63		80.69	89.24	
SQuAD <sub>train</sub> (Sampled)	58.76	70.52	-5.94	76.48	85.06	-3.57	80.07	88.32	-0.92
SQuAD <sub>train</sub> <sup>k=1</sup>	21.44	27.92	-48.54	31.20	37.48	-51.15	38.59	45.27	-43.97
SQuAD <sub>train</sub> <sup>k=1</sup> + First Sentence	53.16	63.21	-13.25	72.75	81.18	-7.45	74.85	82.84	-6.40
SQuAD <sub>train</sub> <sup>k=1</sup> + Sentence Shuffle	54.40	65.20	-11.26	73.37	81.90	-6.73	77.83	86.18	-3.06

Table 1: Performance of QA models trained on the biased SQuAD dataset (SQuAD<sub>train</sub><sup>k=1</sup>), and tested on SQuAD<sub>dev</sub>.  $\Delta$  denotes the difference in F1 score with SQuAD<sub>train</sub>. We use exact match (EM) and F1 score for evaluation.<sup>1</sup>

sentence position mostly give predictions on the corresponding sentence. As a result, BERT (Devlin et al., 2019) trained on a biased training set where every answer appear in the first sentence only achieves 37.48% F1 score in the SQuAD development set whereas the same model trained on the same amount of randomly sampled examples achieves 85.06% F1 score.

To examine the cause of the problem, we thoroughly analyze the learning process of QA models trained on the biased training sets, especially focusing on BERT. Our analysis shows that hidden representations of BERT preserve a different amount of word information according to the word position when trained on the biased training set. The predictions of biased models also become more dependent on the first few words when the training set has answers only in the first sentences.

To tackle the problem, we test various options, ranging from relative position encodings (Yang et al., 2019) to ensemble-based de-biasing methods (Clark et al., 2019; He et al., 2019). While simple baselines motivated by our analysis improve the test performance, our ensemble-based de-biasing method largely improves the performance of most models. Specifically, we use the prior distribution of answer positions as an additional bias model and train models to learn reasoning ability beyond the positional cues.

Contributions of our paper are in threefold; First, we define position bias in extractive question answering and illustrate that common extractive QA models suffer from it. Second, we examine the reason for the failure of the biased models and show that positions can act as spurious biases. Third, we show that the prior distribution of answer positions helps us to build positionally de-biased models, recovering the performance of BERT from 37.48% to 81.64%. We also generalize our findings in many different positions and datasets.<sup>2</sup>

<sup>2</sup><https://github.com/dmis-lab/position-bias>

## 2 Analysis

We first demonstrate the presence of position bias using biased training sets sampled from SQuAD (Rajpurkar et al., 2016) and visualize how position bias propagates in BERT.

### 2.1 Position Bias on Synthetic Datasets

From the original training set  $\mathcal{D}_{\text{train}}$ , we subsample a biased training set  $\mathcal{D}_{\text{train}}^k$  whose answers lie in the  $k$ -th sentence.<sup>3</sup> We conduct experiments on SQuAD ( $\mathcal{D} = \text{SQuAD}$ ) as most examples in SQuAD are answerable with a single sentence (Min et al., 2018). Our analysis mainly focuses on SQuAD<sub>train</sub><sup>k=1</sup> (i.e., all answers are in the first sentence), which has the largest proportion of samples compared to other sentence positions in SQuAD (28,263 out of 87,599). The proportion in the development set (SQuAD<sub>dev</sub>) is similar, having 3,637 out of 10,570 answers in the first sentence. Note that while our analysis is based on SQuAD<sub>train</sub><sup>k=1</sup>, we also test various sentence positions in our main experiments (Section 4.2). We experiment with three popular QA models that provide positions as answers: BiDAF (Seo et al., 2017), BERT (Devlin et al., 2019), and XLNet (Yang et al., 2019). All three models are trained on SQuAD<sub>train</sub><sup>k=1</sup> and are evaluated on SQuAD<sub>dev</sub>. For a fair comparison, we also randomly sample examples from the original training set and make SQuAD<sub>train</sub> (Sampled) which has the same number of examples with SQuAD<sub>train</sub><sup>k=1</sup>.

Table 1 shows the performance of the three models trained on SQuAD<sub>train</sub><sup>k=1</sup>. The performances of all models drop significantly compared to the models trained on SQuAD<sub>train</sub> or SQuAD<sub>train</sub> (Sampled). The relative position encodings in XLNet mitigate position bias to some extent, but its performance

<sup>2</sup>Evaluation code is provided by <https://rajpurkar.github.io/SQuAD-explorer/>

<sup>3</sup>We use Spacy Sentencizer (<https://spacy.io/api/sentencizer>) for the sentence split.

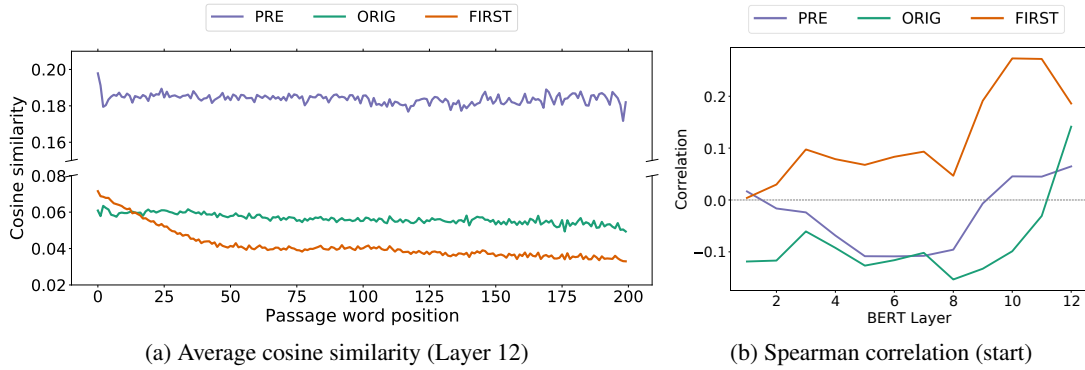


Figure 2: Visualization of position bias with BERT trained on  $\text{SQuAD}_{\text{train}}$  (ORIG),  $\text{SQuAD}_{\text{train}}^{k=1}$  (FIRST), and BERT without fine-tuning (PRE). See Section 2.2 for more details.

still degrades significantly.

To better understand the cause of position bias, we additionally perform two pre-processing methods on  $\text{SQuAD}_{\text{train}}^{k=1}$ . First, we truncate each passage up to the first sentence ( $\text{SQuAD}_{\text{train}}^{k=1}$  + First Sentence). In this case, most performance is recovered, which indicates that the distributions of answer positions are relatively defined with respect to the maximum sequence length. Shuffling the sentence order of  $\text{SQuAD}_{\text{train}}^{k=1}$  ( $\text{SQuAD}_{\text{train}}^{k=1}$  + Sentence Shuffle) also recovers most performance, showing that the spreadness of answers matters. However, these pre-processing methods cannot be a solution as more fine-grained biases (e.g., word level positions) could cause the problem again and models cannot learn proper multi-sentence reasoning from a corrupted context.

## 2.2 Visualization of Position Bias

To visualize how position bias propagates throughout the layers, we compare BERT models, each trained on  $\text{SQuAD}_{\text{train}}^{k=1}$  and  $\text{SQuAD}_{\text{train}}$  respectively and BERT without any fine-tuning. The uncased version of BERT-base is used for the analysis.

Figure 2 (a) shows the amount of word information preserved in the hidden representations at the last layer of BERT. We define the amount of word information for each word position as the cosine similarity between the word embedding and its hidden representation at each layer. The similarities are averaged over the passage-side hidden representations in  $\text{SQuAD}_{\text{dev}}$ .

BERT trained on  $\text{SQuAD}_{\text{train}}^{k=1}$  (FIRST) has higher similarities at the front of the passages compared with BERT trained on  $\text{SQuAD}_{\text{train}}$  (ORIG). In the biased model, the similarity becomes smaller after the first few tokens, which shows position bias of

BERT.

Figure 2 (b) shows the Spearman’s rank correlation coefficient between the final output logits<sup>4</sup> and the amount of word information at each layer defined by the cosine similarity. A higher correlation means that the model is more dependent on the word information kept in that layer. The correlation coefficient is much higher in the biased model (FIRST), especially in the last few layers. Combined with the observation from Figure 2 (a), this indicates that the predictions of the biased model are heavily relying on the information of the first few words.

## 2.3 Why is Position Bias Bad?

Our analysis shows that it is very easy for neural QA models to exploit positional cues whenever possible. While it is natural for neural models to learn the strong but spurious correlation present in the dataset (McCoy et al., 2019; Niven and Kao, 2019), we argue that reading ability should be cultivated independent of such positional correlation. Our study aims to learn proper reading ability even in extreme cases where all answers are in the  $k$ -th sentence. Although exploiting the position distribution within the dataset could help the model improve performance on its corresponding test set, position bias should not be learned since we cannot guarantee realistic test environments to follow similar distribution.

## 3 Method

To prevent models from learning a direct correlation between word positions and answers, we introduce simple remedies for BERT and a bias

<sup>4</sup>We show the results with start position logits and the same pattern is observed with end position logits.

ensemble method with answer prior distributions that can be applied to any QA models.

### 3.1 Baselines

**Randomized Position** To avoid learning the direct correlation between word positions and answers, we randomly perturb input positions. We first randomly sample  $t$  indices from a range of 1 to maximum sequence length of BERT. We sample  $t = 384$  when the maximum sequence length is 512. Then, we sort the indices in an ascending order to preserve the ordering of input words. Perturbed indices then generate position embedding at each token position, which replaces the original position embedding.

**Entropy Regularization** Inspired by the observation in Section 2.2, we force our model to preserve a constant amount of word information regardless of the word positions. Maximizing the entropy of normalized cosine similarity between the word embeddings and their hidden representations encourages models to maintain a uniform amount of information. As the cosine similarities are not probabilities, we normalize them to be summed to 1. We compute the entropy regularization term from the last layer and add it to the start/end prediction loss with a scaling factor  $\lambda$ .

### 3.2 Bias Ensemble with Answer Prior

Bias ensemble methods (Clark et al., 2019; He et al., 2019) combine the log probabilities from a pre-defined bias model and a target model to de-bias. Ensembling makes the target model to learn different probabilities other than the bias probabilities. In our case, we define the prior distribution of the answer positions as our bias model. Specifically, we introduce the sentence-level answer prior and the word-level answer prior.

**Bias Ensemble Method** Given a passage and question pair, a model has to find the optimal start and end positions of the answer in the passage, denoted as  $y_s, y_e$ . Typically, the model outputs two probability distributions  $p^s$  and  $p^e$  for the start and end positions. As our method is applied in the same manner for both start and end predictions, we drop the superscript from  $p^s, p^e$  and subscript from  $y_s, y_e$  whenever possible.

For ensembling two different log probabilities from the bias model and the target model, we use a product of experts (Hinton, 2002). Using the

product of experts, a probability at the  $i$ -th position is calculated as:

$$\hat{p}_i = \text{softmax}(\log(p_i) + \log(b_i)) \quad (1)$$

where  $\log(p_i)$  is a log probability from the target model and  $\log(b_i)$  is a log probability from the bias model. The ensembled probability  $\hat{p}$  is used for the training.

To dynamically choose the amount of bias for each sample, Clark et al. (2019) introduce a learned mixing ensemble with a trainable parameter. Probabilities in the training phase are now defined as:

$$\hat{p}_i = \text{softmax}(\log(p_i) + g(X)\log(b_i)) \quad (2)$$

We use hidden representations before the softmax layer as  $X$ .  $g(X)$  then applies affine transformation on the representations to obtain a scalar value. Softplus activation followed by max pooling is used to obtain positive values. As BiDAF has separate hidden representations for the start and end predictions, we separately define  $g(X)$  for each start and end representation.

As models often learn to ignore the biases and make  $g(X)$  to 0, Clark et al. (2019) suggest adding an entropy penalty term to the loss function. However, the entropy penalty did not make much difference in our case as  $g(X)$  was already large enough. Note that we only use  $\log(b_i)$  during training, and the predictions are solely based on the predicted log probability  $\log(p_i)$  from the model.

We define bias log probability as pre-calculated answer priors. Using prior distributions in machine learning has a long history such as using class frequency in the class imbalance problem (Domingos, 1999; Japkowicz and Stephen, 2002; Zhou and Liu, 2006; Huang et al., 2016). In our case, the class prior corresponds to the prior distribution of answer positions.

**Word-level Answer Prior** First, we consider the word-level answer prior. Given the training set having  $N$  examples having  $N$  answers  $\{y^{(1)}, y^{(2)}, \dots, y^{(N)}\}$ , we compute the word-level answer prior at position  $i$  over the training set. In this case, our bias log probability at  $i$ -th position is:

$$\log(b_i) := \frac{1}{N} \sum_{j=1}^N \mathbf{1}[y^{(j)} = i] \quad (3)$$

where we use the indicator function  $\mathbf{1}[cond]$ . Bias log probabilities for the end position prediction are

calculated in a similar manner. Note that the word-level answer prior gives an equal bias distribution for each passage while the distribution is more fine-grained than the sentence-level prior described in the next section.

**Sentence-level Answer Prior** We also use the sentence-level answer prior which dynamically changes depending on the sentence boundaries of each sample. First, we define a set of sentences  $\{S_1^{(j)}, \dots, S_L^{(j)}\}$  for the  $j$ -th training passage, where  $L$  is the maximum number of sentence in whole training passages. Then, the sentence-level answer prior of the  $i$ -th word position (for the start prediction) for the  $j$ -th sample, is derived from the frequency of answers appearing in the  $l$ -th sentence:

$$\log(b_i^{(j)}) := \frac{1}{N} \sum_{k=1}^N \mathbf{1}[y^{(k)} \in S_i^{(k)}], \quad i \in S_i^{(j)} \quad (4)$$

Note that as boundaries of sentences in each sample are different, bias log probabilities should be defined in every sample. Again, bias log probabilities for the end positions are calculated similarly.

It is very convenient to calculate the answer priors for any datasets. For instance, on  $\mathcal{D}_{\text{train}}^{k=1}$ , we use the first sentence indicator as the sentence-level answer prior as all answers are in the first sentence. More formally, the sentence-level answer prior for  $\mathcal{D}_{\text{train}}^{k=1}$  is 1 for  $l = 1$ , and 0 when  $l > 1$ :

$$\log(b_i^{(j)}) := \begin{cases} 1 & i \in S_1^{(j)}, \\ 0 & i \notin S_1^{(j)} \end{cases} \quad (5)$$

which is a special case of the sentence-level answer prior. For general datasets where the distributions of answer positions are less skewed, the answer priors are more softly distributed. See Appendix B for a better understanding of the answer priors.

Both word-level and sentence-level answer priors are experimented with two bias ensemble methods: product of experts with bias (**Bias Product**, Equation 1) and learned mixing of two log probabilities (**Learned-Mixin**, Equation 2).

## 4 Experiments

We first experiment the effects of various de-biasing methods on three different QA models using both biased and full training sets. Our next experiments generalize our findings in different sentence positions and different datasets such as NewsQA (Trischler et al., 2017) and NaturalQuestions (Kwiatkowski et al., 2019).

### 4.1 Effect of De-biasing Methods

We first train all three models (BiDAF, BERT, and XLNet) on  $\text{SQuAD}_{\text{train}}^{k=1}$  with our de-biasing methods and evaluate them on  $\text{SQuAD}_{\text{dev}}$  (original development set),  $\text{SQuAD}_{\text{dev}}^{k=1}$ , and  $\text{SQuAD}_{\text{dev}}^{k=2,3,\dots}$ . Note that  $\text{SQuAD}_{\text{dev}}^{k=2,3,\dots}$  is another subset of  $\text{SQuAD}_{\text{dev}}$ , whose answers do not appear in the first sentence, but in other sentences. We also experiment with BERT trained on the full training set,  $\text{SQuAD}_{\text{train}}$ .

For all models, we use the same hyperparameters and training procedures as suggested in their original papers (Seo et al., 2017; Devlin et al., 2019; Yang et al., 2019), except for batch sizes and training epochs (See Appendix A).  $\lambda$  for the entropy regularization is set to 5. Most of our implementation is based on the PyTorch library.

**Results with  $\text{SQuAD}_{\text{train}}^{k=1}$**  The results of applying various de-biasing methods on three models with  $\text{SQuAD}_{\text{train}}^{k=1}$  are in Table 2. Performance of all models without any de-biasing methods (denoted as ‘None’) is very low on  $\text{SQuAD}_{\text{dev}}^{k=2,3,\dots}$ , but fairly high on  $\text{SQuAD}_{\text{dev}}^{k=1}$ . This means that their predictions are highly biased towards the first sentences. In the case of BERT, F1 score on  $\text{SQuAD}_{\text{dev}}^{k=1}$  is 85.81%, while F1 score on  $\text{SQuAD}_{\text{dev}}^{k=2,3,\dots}$  is merely 12.12%. Our simple baseline approaches used in BERT improve the performance up to 34.63% F1 score (Random Position) while the entropy regularization is not significantly effective.

Bias ensemble methods using answer priors consistently improve the performance of all models. The sentence-level answer prior works the best, which obtains a significant gain after applying the Learned-Mixin method. We found that the coefficient  $g(X)$  in Equation 2 averages to 7.42. during training for BERT + Learned-Mixin, which demonstrates a need of proper balancing between the probabilities. The word-level answer prior does not seem to provide strong position bias signals as its distribution is much softer than the sentence-level answer prior.

**Results with  $\text{SQuAD}_{\text{train}}$**  The results of training BERT with our de-biasing methods on the full training set  $\text{SQuAD}_{\text{train}}$  are in the bottom of Table 2. Note that the answer prior is more softened than the answer prior used in  $\text{SQuAD}_{\text{train}}^{k=1}$  as answers are now spread in all sentence positions. While exploiting the positional distribution of the training set

De-biasing Method		SQuAD <sub>dev</sub> <sup>k=1</sup>		SQuAD <sub>dev</sub> <sup>k=2,3,...</sup>		SQuAD <sub>dev</sub>	
		EM	F1	EM	F1	EM	F1
<b>BERT trained on SQuAD<sub>train</sub><sup>k=1</sup></b>							
Baseline	None	77.07	85.81	7.14	12.12	31.20	37.48
	Random Position	69.95	80.73	27.32	34.63	41.99	50.49
	Entropy Regularization	77.40	86.17	10.50	15.72	33.52	39.96
Word-Level	Bias Product	<b>78.61</b>	<b>87.08</b>	7.85	12.88	32.19	38.41
	Learned-Mixin	78.17	86.56	8.55	13.43	32.51	38.59
Sentence-Level	Bias Product	78.39	87.06	13.33	18.73	35.71	42.24
	Learned-Mixin	77.18	85.15	<b>71.31</b>	<b>79.79</b>	<b>73.33</b>	<b>81.64</b>
<b>BiDAF trained on SQuAD<sub>train</sub><sup>k=1</sup></b>							
Baseline	None	61.04	72.91	0.66	4.34	21.44	27.92
Sentence-Level	Bias Product	<b>62.00</b>	<b>73.87</b>	0.78	4.48	21.84	28.36
	Learned-Mixin	56.53	66.79	<b>50.28</b>	<b>60.77</b>	<b>52.43</b>	<b>62.84</b>
<b>XLNet trained on SQuAD<sub>train</sub><sup>k=1</sup></b>							
Baseline	None	78.99	87.24	11.52	16.77	38.59	45.27
Sentence-Level	Bias Product	<b>79.24</b>	<b>87.88</b>	33.28	39.93	49.09	56.43
	Learned-Mixin	68.82	82.05	<b>64.63</b>	<b>77.65</b>	<b>66.07</b>	<b>79.16</b>
<b>BERT trained on SQuAD<sub>train</sub></b>							
Baseline	None	81.55	88.68	81.21	88.61	81.32	88.63
Sentence-Level	Bias Product	<b>81.88</b>	<b>88.87</b>	<b>81.29</b>	<b>88.87</b>	<b>81.49</b>	<b>88.87</b>
	Learned-Mixin	81.58	88.38	80.87	88.47	81.12	88.44

Table 2: Results of applying de-biasing methods. Each model is evaluated on SQuAD<sub>dev</sub> and two subsets: SQuAD<sub>dev</sub><sup>k=1</sup> and SQuAD<sub>dev</sub><sup>k=2,3,...</sup>.

could be more helpful when evaluating on the development set that has a similar positional distribution, our method maintains the original performance. It shows that our method works safely when the positional distribution doesn't change much.

**Visualization** To investigate the effect of de-biasing methods, we visualize the word information in each layer as done in Section 2.2. We visualize the BERT trained on SQuAD<sub>train</sub><sup>k=1</sup> ensemble with sentence-level answer prior in Figure 3. The bias product method (**PRODUCT**) and the model without any de-biasing methods (**NONE**) are similar, showing that it still has position bias. The learned-mixin method (**MIXIN**), on the other hand, safely delivers the word information across different positions.

#### 4.2 Generalizing to Different Positions

As the SQuAD training set has many answers in the first sentence, we mainly test our methods on SQuAD<sub>train</sub><sup>k=1</sup>. However, does our method gener-

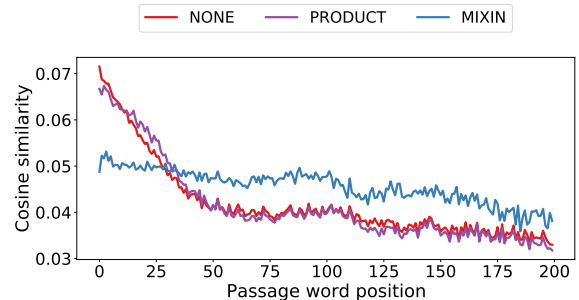


Figure 3: Visualization of BERT models trained on SQuAD<sub>train</sub><sup>k=1</sup> with and without de-biasing method.

alize to different sentence positions? To answer this question, we construct four SQuAD<sub>train</sub><sup>k</sup> datasets based on the sentence positions of answers. Note that unlike SQuAD<sub>train</sub><sup>k=1</sup>, the number of samples becomes smaller and the sentence boundaries are more blurry when  $k > 1$ , making answer priors much softer. We train three QA models on different biased datasets and evaluate them on SQuAD<sub>dev</sub> with and without de-biasing methods.

	SQuAD <sub>dev</sub>							
	EM	F1	EM	F1	EM	F1	EM	F1
SQuAD <sub>train</sub> <sup>k</sup>	<i>k</i> = 2 (20,593 samples)		<i>k</i> = 3 (15,567 samples)		<i>k</i> = 4 (10,379 samples)		<i>k</i> = 5, 6, ... (12,610 samples)	
<b>BiDAF</b>	18.43	25.74	12.26	19.04	9.96	16.50	12.34	19.65
+Bias Product	21.51	28.67	11.19	18.39	11.20	17.78	10.09	16.78
+Learned-Mixin	<b>47.49</b>	<b>58.36</b>	<b>43.57</b>	<b>53.80</b>	<b>30.18</b>	<b>39.51</b>	<b>18.51</b>	<b>27.30</b>
<b>BERT</b>	36.16	43.14	44.76	52.89	49.13	58.01	57.95	66.69
+Bias Product	52.89	50.38	52.42	60.99	53.39	62.69	58.75	67.67
+Learned-Mixin	<b>71.61</b>	<b>80.36</b>	<b>69.04</b>	<b>77.91</b>	<b>64.31</b>	<b>73.72</b>	<b>62.82</b>	<b>72.30</b>
<b>XLNet</b>	47.55	55.01	46.67	54.56	50.49	58.74	58.29	66.67
+Bias Product	59.49	67.35	61.99	70.89	67.26	76.55	72.44	81.85
+Learned-Mixin	<b>68.34</b>	<b>80.35</b>	<b>69.28</b>	<b>79.99</b>	<b>70.07</b>	<b>80.12</b>	<b>73.33</b>	<b>82.79</b>

Table 3: Position bias in different positions. Each model is trained on a biased SQuAD dataset (SQuAD<sub>train</sub><sup>k</sup>) and evaluated on SQuAD<sub>dev</sub>.

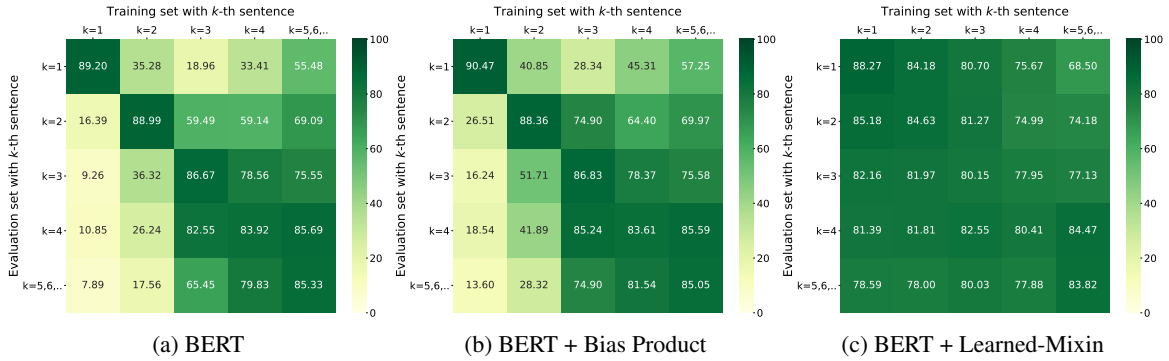


Figure 4: Sentence-wise position bias in SQuAD. Models are trained on SQuAD<sub>train</sub><sup>k</sup> and evaluated on SQuAD<sub>dev</sub><sup>k</sup>. (a) Standard BERT suffers from position bias as the off-diagonal performance is significantly lower. (b), (c) Our de-biasing method successfully handles the bias and provides consistently higher performance.

**Results** As shown in Table 3, all three models suffer from position bias in every sentence position while the learned-mixin method (+Learned-Mixin) successfully resolves the bias. Due to the blurred sentence boundaries, position bias is less problematic when *k* is large. We observe a similar trend in BERT and XLNet while a huge performance drop is observed in BiDAF even with a large *k*.

**Visualization** Figure 4 visualizes the sentence-wise position biases. We train BERT, BERT + Bias Product and BERT + Learned Mixin on different subsets of SQuAD training set (SQuAD<sub>train</sub><sup>k</sup>) and evaluated on every SQuAD<sub>dev</sub><sup>k</sup> whose answers lie only in the *k*-th sentence. As a result, the low performance in the off-diagonal represent the presence of position bias. The figure shows that the biased model fails to predict the answers in different sentence positions (Figure 4 (a)) while our de-biased model achieves high performance regardless of the

sentence position (Figure 4 (c)). Again, as the value of *k* increases, the boundary of the *k*-th sentence varies a lot in each sample, which makes the visualization of sentence-wise bias difficult.

### 4.3 NewsQA and NaturalQuestions

We test the effect of de-biasing methods on datasets having different domains and different degrees of position bias. NewsQA (Trischler et al., 2017) is an extractive QA dataset that includes passages from CNN news articles. NaturalQuestions (Kwiatkowski et al., 2019) is a dataset containing queries and passages collected from the Google search engine. We use the pre-processed dataset provided by the MRQA shared task (Fisch et al., 2019).<sup>5</sup>

For each dataset, we construct two sub-training datasets; one contains samples with answers in the first sentence (*k* = 1), and the other contains

<sup>5</sup><https://github.com/mrqa/MRQA-Shared-Task-2019>

NewsQA <sub>train</sub> <sup>k</sup>	NewsQA <sub>dev</sub>		
	k = All	k = 1	k = 2, 3, ...
<b>BERT</b>	<b>69.94</b>	27.99	56.15
+Bias Product	69.46	28.81	56.86
+Learned-Mixin	69.42	<b>44.50</b>	<b>58.22</b>

Table 4: F1 scores on NewsQA. Models are evaluated on the original development dataset (NewsQA<sub>dev</sub>).

NQ <sub>train</sub> <sup>k</sup>	NQ <sub>dev</sub>		
	k = All	k = 1	k = 2, 3, ...
<b>BERT</b>	78.79	56.79	49.59
+Bias Product	78.84	56.77	53.34
+Learned-Mixin	<b>79.04</b>	<b>72.83</b>	<b>60.63</b>

Table 5: F1 scores on NaturalQuestions. Models are evaluated on the original development dataset (NQ<sub>dev</sub>).

the remaining samples ( $k = 2, 3, \dots$ ). Models are trained on the original dataset and two sub-training datasets and evaluated on the original development set.

**Implementation Details** For NewsQA, we truncate each paragraph so that the length of each context is less than 300 words. We eliminate training and development samples that become unanswerable due to the truncation. For NaturalQuestions, we choose firstly occurring answers for training extractive QA models, which is a common approach in weakly supervised setting (Joshi et al., 2017; Talmor and Berant, 2019).

From NewsQA and NaturalQuestions, we construct two sub-training datasets having only the first annotated samples ( $\mathcal{D}_{\text{train}}^{k=1}$ ) and the remaining samples ( $\mathcal{D}_{\text{train}}^{k=2,3,\dots}$ ). For a fair comparison, we fix the size of two sub-training sets to have 17,000 (NewsQA) and 40,000 samples (NaturalQuestions).

**Results** In Table 4 and Table 5, we show results of applying our methods. In both datasets, BERT, trained on biased datasets ( $k = 1$  and  $k = 2, 3, \dots$ ), significantly suffers from position bias. Position bias is generally more problematic in the  $k = 1$  datasets while for NaturalQuestions,  $k = 2, 3, \dots$  is also problematic. Our de-biasing methods prevent performance drops in all cases without sacrificing the performance on the full training set ( $k = \text{All}$ ).

## 5 Related Work

Various question answering datasets have been introduced with diverse challenges including reasoning over multiple sentences (Joshi et al., 2017),

answering multi-hop questions (Yang et al., 2018), and more (Trischler et al., 2017; Welbl et al., 2018; Kwiatkowski et al., 2019; Dua et al., 2019). Introduction of these datasets rapidly progressed the development of effective QA models (Wang and Jiang, 2016; Seo et al., 2017; Xiong et al., 2017; Wang et al., 2017; Yu et al., 2018; Devlin et al., 2019; Yang et al., 2019), but most models predict the answer as positions without much discussion on it.

Our work builds on the analyses of dataset biases in machine learning models and ways to tackle them. For instance, sentence classification models in natural language inference and argument reasoning comprehension suffer from word statistics bias (Poliak et al., 2018; Minervini and Riedel, 2018; Kang et al., 2018; Belinkov et al., 2019; Niven and Kao, 2019). On visual question answering, models often ignore visual information due to the language prior bias (Agrawal et al., 2016; Zhang et al., 2016; Goyal et al., 2017; Johnson et al., 2017; Agrawal et al., 2018). Several studies in QA also found that QA models do not leverage the full information in the given passage (Chen et al., 2016; Min et al., 2018; Chen and Durrett, 2019; Min et al., 2019). Adversarial datasets have been also proposed to deal with this type of problem (Jia and Liang, 2017; Rajpurkar et al., 2018). In this study, we define position bias coming from the prediction structure of QA models and show that positionally biased models can ignore information in different positions.

Our proposed methods are based on the bias ensemble method (Clark et al., 2019; He et al., 2019). Ensembling with the bias model encourages the model to solve tasks without converging to bias shortcuts. Clark et al. (2019) conducted de-biasing experiments on various tasks including two QA tasks while they use tf-idf and the named entities as the bias models.

It is worth noting that several models incorporate the pointer network to predict the answer positions in QA (Vinyals et al., 2015; Wang and Jiang, 2016; Wang et al., 2017). Also, instead of predicting positions, some models predict the n-grams as answers (Lee et al., 2016; Seo et al., 2019), generate answers in a vocabulary space (Raffel et al., 2019), or use a generative model (Lewis and Fan, 2019). We expect that these approaches suffer less from position bias and leave the evaluation of position bias in these models as our future work.



## 6 Conclusion

Most QA studies frequently utilize start and end positions of answers as training targets without much considerations. Our study shows that most QA models fail to generalize over different positions when trained on datasets having answers in a specific position. Our findings show that position can work as a spurious bias and alert researchers when building QA models and datasets. We introduce several de-biasing methods to make models to ignore the spurious positional cues, and find out that the sentence-level answer prior is very useful. Our findings also generalize to different positions and different datasets. One limitation of our approach is that our method and analysis are based on a single paragraph setting which should be extended to a multiple paragraph setting to be more practically useful.

## Acknowledgments

This research was supported by a grant of the Korea Health Technology R&D Project through the Korea Health Industry Development Institute (KHIDI), funded by the Ministry of Health & Welfare, Republic of Korea (grant number: HR20C0021). This research was also supported by National Research Foundation of Korea (NRF-2017R1A2A1A17069645, NRF-2017M3C4A7065887).

## References

- Aishwarya Agrawal, Dhruv Batra, and Devi Parikh. 2016. Analyzing the behavior of visual question answering models. In *EMNLP*.
- Aishwarya Agrawal, Dhruv Batra, Devi Parikh, and Aniruddha Kembhavi. 2018. Don't just assume; look and answer: Overcoming priors for visual question answering. In *CVPR*.
- Yonatan Belinkov, Adam Poliak, Stuart M Shieber, Benjamin Van Durme, and Alexander Rush. 2019. On adversarial removal of hypothesis-only bias in natural language inference. *NAACL HLT*.
- Danqi Chen, Jason Bolton, and Christopher D Manning. 2016. A thorough examination of the cnn/daily mail reading comprehension task. In *ACL*.
- Jifan Chen and Greg Durrett. 2019. Understanding dataset design choices for multi-hop reasoning. In *NAACL-HLT*.
- Christopher Clark, Mark Yatskar, and Luke Zettlemoyer. 2019. Don't take the easy way out: Ensemble based methods for avoiding known dataset biases. In *EMNLP-IJCNLP*.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*.
- Pedro Domingos. 1999. Metacost: a general method for making classifiers cost-sensitive. In *SIGKDD*.
- Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. 2019. Drop: A reading comprehension benchmark requiring discrete reasoning over paragraphs. In *NAACL-HLT*.
- Adam Fisch, Alon Talmor, Robin Jia, Minjoon Seo, Eunsol Choi, and Danqi Chen. 2019. Mrqa 2019 shared task: Evaluating generalization in reading comprehension. In *Proceedings of the 2nd Workshop on Machine Reading for Question Answering*.
- Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. 2017. Making the v in vqa matter: Elevating the role of image understanding in visual question answering. In *CVPR*.
- He He, Sheng Zha, and Haohan Wang. 2019. Unlearn dataset bias in natural language inference by fitting the residual. *EMNLP-IJCNLP 2019*.
- Geoffrey E Hinton. 2002. Training products of experts by minimizing contrastive divergence. *Neural computation*.
- Chen Huang, Yining Li, Chen Change Loy, and Xiaoou Tang. 2016. Learning deep representation for imbalanced classification. In *CVPR*.
- Nathalie Japkowicz and Shaju Stephen. 2002. The class imbalance problem: A systematic study. *Intelligent data analysis*.
- Robin Jia and Percy Liang. 2017. Adversarial examples for evaluating reading comprehension systems. In *EMNLP*.
- Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. 2017. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *CVPR*.
- Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *ACL*.
- Dongyeop Kang, Tushar Khot, Ashish Sabharwal, and Eduard Hovy. 2018. Adventure: Adversarial training for textual entailment with knowledge-guided examples. In *ACL*.
- Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Matthew Kelcey, Jacob Devlin, Kenton Lee, Kristina N. Toutanova, Llion Jones, Ming-Wei Chang, Andrew Dai, Jakob

- Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural questions: a benchmark for question answering research. *TACL*.
- Kenton Lee, Shimi Salant, Tom Kwiatkowski, Ankur Parikh, Dipanjan Das, and Jonathan Berant. 2016. Learning recurrent span representations for extractive question answering. *arXiv preprint arXiv:1611.01436*.
- M Lewis and A Fan. 2019. Generative question answering: Learning to answer the whole question. In *ICLR*.
- Tom McCoy, Ellie Pavlick, and Tal Linzen. 2019. Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference. In *ACL*.
- Sewon Min, Eric Wallace, Sameer Singh, Matt Gardner, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2019. Compositional questions do not necessitate multi-hop reasoning. In *ACL*.
- Sewon Min, Victor Zhong, Richard Socher, and Caiming Xiong. 2018. Efficient and robust question answering from minimal context over documents. In *ACL*.
- Pasquale Minervini and Sebastian Riedel. 2018. Adversarially regularising neural nli models to integrate logical background knowledge. *CoNLL*, page 65.
- Timothy Niven and Hung-Yu Kao. 2019. Probing neural network comprehension of natural language arguments. In *ACL*.
- Adam Poliak, Jason Naradowsky, Aparajita Haldar, Rachel Rudinger, and Benjamin Van Durme. 2018. Hypothesis only baselines in natural language inference. In *Proceedings of the Seventh Joint Conference on Lexical and Computational Semantics*.
- Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*.
- Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for squad. In *ACL*.
- Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*.
- Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2017. Bidirectional attention flow for machine comprehension. In *ICLR*.
- Minjoon Seo, Jinhyuk Lee, Tom Kwiatkowski, Ankur Parikh, Ali Farhadi, and Hannaneh Hajishirzi. 2019. Real-time open-domain question answering with dense-sparse phrase index. In *ACL*.
- Alon Talmor and Jonathan Berant. 2019. Multiqa: An empirical investigation of generalization and transfer in reading comprehension. In *ACL*.
- Adam Trischler, Tong Wang, Xingdi Yuan, Justin Harris, Alessandro Sordani, Philip Bachman, and Kaheer Suleman. 2017. Newsqa: A machine comprehension dataset. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*.
- Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *NIPS*.
- Shuohang Wang and Jing Jiang. 2016. Machine comprehension using match-lstm and answer pointer. *arXiv preprint arXiv:1608.07905*.
- Wenhui Wang, Nan Yang, Furu Wei, Baobao Chang, and Ming Zhou. 2017. Gated self-matching networks for reading comprehension and question answering. In *ACL*.
- Johannes Welbl, Pontus Stenetorp, and Sebastian Riedel. 2018. Constructing datasets for multi-hop reading comprehension across documents. *TACL*.
- Caiming Xiong, Victor Zhong, and Richard Socher. 2017. Dcn+: Mixed objective and deep residual coattention for question answering. In *ICLR*.
- Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *arXiv preprint arXiv:1906.08237*.
- Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *EMNLP*.
- Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V Le. 2018. Qanet: Combining local convolution with global self-attention for reading comprehension. In *ICLR*.
- Peng Zhang, Yash Goyal, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. 2016. Yin and yang: Balancing and answering binary visual questions. In *CVPR*.
- Zhi-Hua Zhou and Xu-Ying Liu. 2006. On multi-class cost-sensitive learning. In *AAAI*.

## A Implementation Details

**Details of Training** For all experiments, we use uncased BERT-base and cased XLNet-base. We modify the open-sourced Pytorch implementation of models.<sup>6</sup> BiDAF is trained with the batch size of 64 for 30 epochs and BERT and XLNet are trained for 2 epochs with batch sizes 12 and 10, respectively. The choice of hyperparameters mainly comes from the limitation of our computational resources and mostly follows the default setting used in their original works. Note that our de-biasing methods do not require additional hyperparameters.

For all three models, the number of parameters remains the same as default settings with bias product and increases by a single linear layer with learned-mixin. We trained models on a single Titan X GPU. The average training time of the bias ensemble method is similar to the original models.

## B Examples of Answer Prior

To provide a better understanding of our methods, Figure B.1 shows examples of answer priors, which are used as bias models. See section 3 for detail.

## C Visualization of Position Bias

In Figure C.1, we plot the preserved amount of word information in the middle layers of BERT. Figure C.2 shows the effect of applying the de-biasing methods in each layer of BERT. See Section 2 and 4.1 for more detail. We plot the results of layers 1, 4, 7, 10, and 11.

---

<sup>6</sup><https://github.com/allenai/allennlp>, <https://github.com/huggingface/transformers>

**Word-Level Answer Prior Distribution**

$\mathcal{D}_{\text{train}}$  : [0.15, 0.10, 0.12, 0.10, 0.05, 0.08, 0.05, 0.03, 0.02, 0.04, ...]

$\mathcal{D}_{\text{train}}^{k=1}$  : [0.30, 0.20, 0.20, 0.15, 0.09, 0.01, 0.01, 0.004, 0.002, 0.001, ...]

**Sentence-Level Answer Prior Distribution**

$\mathcal{D}_{\text{train}}$  : [0.80, 0.20]

$\mathcal{D}_{\text{train}}^{k=1}$  : [1, 0]

Answer Prior	Dataset	Sentence 1					Sentence 2				
		Word1	Word2	Word3	Word4	.	Word5	Word6	Word7	Word8	.
Word Level	$\mathcal{D}_{\text{train}}$	0.15	0.10	0.12	0.10	0.05	0.08	0.05	0.03	0.02	0.04
	$\mathcal{D}_{\text{train}}^{k=1}$	0.30	0.20	0.20	0.15	0.09	0.01	0.01	0.004	0.002	0.001
Sentence Level	$\mathcal{D}_{\text{train}}$	0.8	0.8	0.8	0.8	0.8	0.2	0.2	0.2	0.2	0.2
	$\mathcal{D}_{\text{train}}^{k=1}$	1	1	1	1	1	0	0	0	0	0

Figure B.1: Example of three types of answer priors, word-level answer prior (Word-Level), sentence-level answer prior (Sentence-Level) and sentence-level answer prior on  $\mathcal{D}_{\text{train}}^{k=1}$  (Sentence-Level (First)).

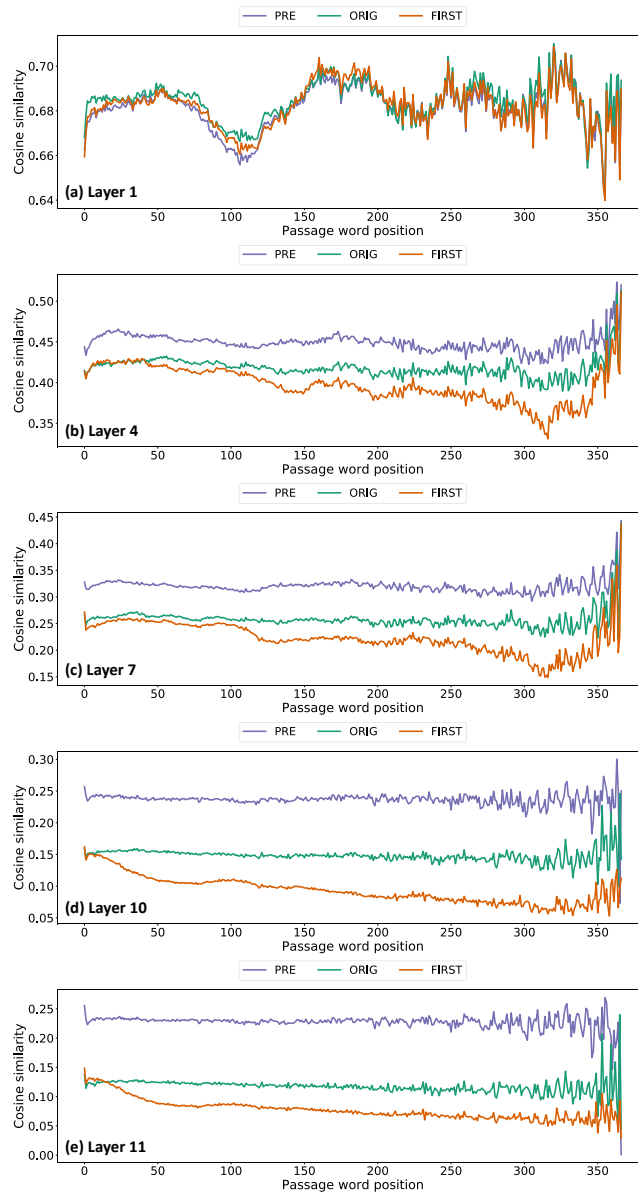


Figure C.1: Visualization of each layer of BERT trained on SQuAD<sub>train</sub> (ORIG), SQuAD<sub>train</sub><sup>k=1</sup> (FIRST), and BERT without fine-tuning (PRE). As the input passes each layer, position bias becomes more problematic.

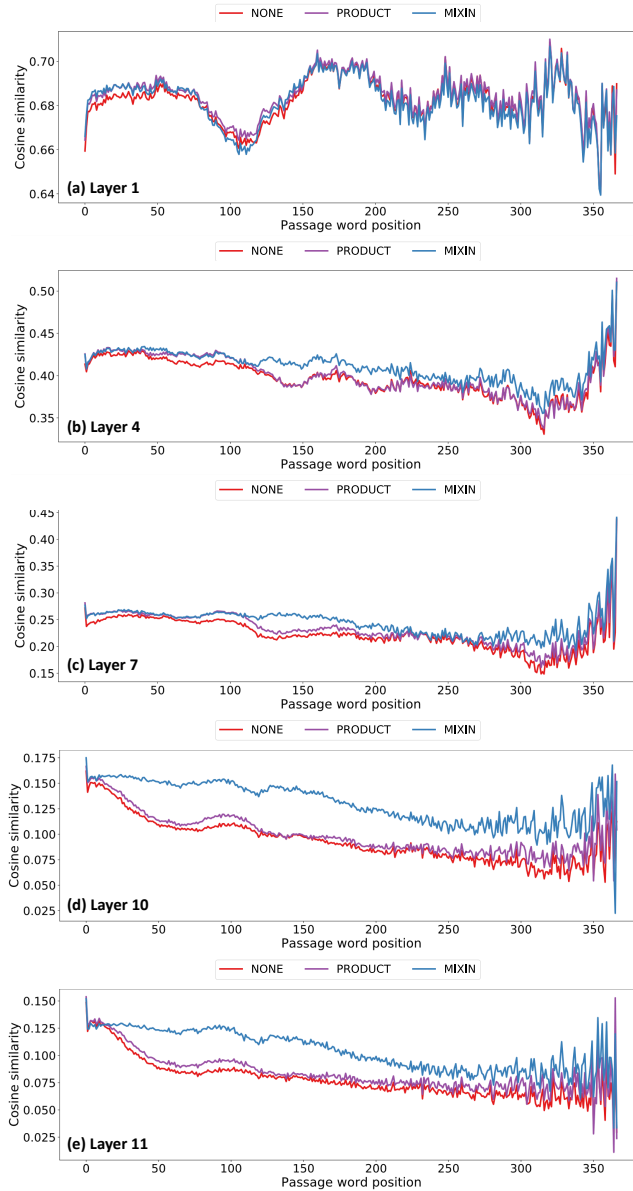


Figure C.2: Visualization of each layer of de-biased BERT. BERT trained on  $\text{SQuAD}_{\text{train}}^{k=1}$  without any de-biasing methods (**NONE**), with sentence-level prior bias product (**PRODUCT**), with learned-mixin (**MIXIN**). **MIXIN** preserves consistent information compared with **NONE** and prevents the bias propagation.