Do We Need to Differentiate Negative Candidates Before Training a Neural Ranker?

Anonymous ACL submission

Abstract

Retrieval-based Question Answering (ReQA) requires a system to find candidates (e.g., sentences or short passages) containing the answer to a given question from a large corpus. A promising way to solve this task is a two-stage pipeline, where the first stage retrieves a set of candidates, and the second stage uses a neural network to rank the retrieved candidates. There are three standard methods to train neural rankers, Binary Cross Entropy loss, Mean Square Error loss, and Hinge loss. While all these training strategies assign the same label for all the negative candidates, we argue that negativeness is not binary but exists as a spectrum, i.e., some candidates may be more negative than the others, and thus should be treated differently. We present SCONER—scoring negative candidates before training neural ranker—a model trained to differentiate negative candidates. Our approach includes 1) semantic textual similarity-based scoring together with data augmentation for score generation of negative candidates; and 2) a neural ranker trained on data using generated scores as labels. Together, we systematically compare three standard training methods and our proposed method on a range of ReQA datasets under multiple settings (i.e., single-domain and multi-domain). Our finding suggests that using more negative candidates to train neural rankers are better than less in both single- and multi-domain settings, where SCONER is the best in the single-domain settings and Hinge loss is the best in multi-domain settings.

1 Introduction

Retrieval Based Question Answering (ReQA) has gained increasing interest and attention in recent years, and many benchmarks have been proposed (Cohen et al., 2018; Khot et al., 2020; Ahmad et al., 2019; Guo et al., 2020). The target of such a task is to retrieve candidates (e.g., sentences or snippets) containing the answer to a question from a large corpus. ReQA is different from "reading comprehension" which aims to extract answer span(s) from a given passage. ReQA is closer to real-world applications where the relevant passage to a question is usually unknown and needs to be retrieved from a large corpus.

A promising approach for solving ReQA involves two stages (coarse-to-fine): first, retrieve a small set of candidates from a large corpus and second re-rank these candidates. The re-ranking stage usually involves neural models to capture the interactive contextual information (Yilmaz et al., 2019; Nogueira and Cho, 2019) which can significantly improve the initial retrieval performance (Ozyurt et al., 2020), and thus it is crucial for any retrieval system (Ma et al., 2020). The focus of this work is to study and improve the neural ranker.

A common strategy to train a neural ranker is to treat it as a binary classification (or regression) model which is trained on balanced positive examples (i.e. question and relevant candidate) and negative examples (i.e. question and irrelevant candidate). The positive examples are human-annotated answers, and negative examples are usually randomly selected from the corpus. However, this strategy results in three issues that potentially lead a model to underperform. *Issue 1:* Some semantic relevant candidates which are not annotated as answers might be selected as negative samples, which injects noise into the training data. *Issue 2*: Many negative samples are unused to keep a balanced training set due to a small set of positive samples. This issue is amplified in the low resource scenario since the size of the training data is not enough to train a neural model. *Issue 3:* All negative examples are equally treated, even though they are not semantically equivalent to a question.

Apart from binary classification (or regression), another training strategy is to use Hing Loss (explained in §5.2) so that more negative candidates can be utilized. This strategy can resolve *Issue 2*,

however, *issue1* and *issue3* remain. These issues originate the question, *"Do we need to differentiate negative candidates before training a neural ranker?"*. Driven by this question, we propose a scoring approach for negative candidates (§3), which has two stages. First, we train a model on Semantic Textual Similarity (STS) benchmark (Conneau and Kiela, 2018). This model generates a high score for a two-sentences pair if they are semantically similar; otherwise, a low score. Second, we use this STS model to generate a score for a question-negative candidate pair. Instead of using straightforward questions and negative candidates as input, we explore different data augmentation approaches to ensure that the more information the negative candidate has to answer the question, the higher score the STS model generates for this pair. This scoring approach allows: 1) a good candidate that is not annotated as an answer to have a high score–addresses *Issue 1*; 2) more negative samples can be used to train a neural ranker–addresses *Issue 2*, and 3) each negative candidate to receive a "negativeness" score which can be helpful in differentiating candidates–addresses *Issue 3*.

Figure 1(a) shows the pipeline of utilizing the scoring approach to train a neural ranker. First, a set of negative candidates are retrieved by BM25. Second, the score generation module produces scores for the question and negative candidate pairs. Last, a neural ranker is trained on both negative samples and positive samples. Figure 1(b) shows the inference pipeline: given a question, BM25 retrieves a set of candidates, then, the neural ranker predicts a score for each candidate, and finally, the top-k[1] candidates are considered as the final prediction.

We compare three standard training strategies and our proposed method on MultiReQA (Guo et al., 2020) benchmark, which includes 5 in-domain datasets and 4 out-of-domain datasets. We observe that 1) using more negative candidates to train a neural ranker is better in both single- and multi-domains settings; 2) such advantage is more obvious in low resource training data; 3) our proposed score generation approach improves the performance compared to three standard methods in the single-domain and two in multi-domain setting; 4) the proposed data augmentation methods are effective in generating better negativeness scores; and 5) the knowledge of an STS model can be transferred to an answer selection model in two ways, generating negativeness scores and initializing a neural ranker with the STS model.

In summary, our contributions are in three folds. First, we systematically study three standard training strategies of neural rankers. Second, we propose a scoring approach to differentiate negative candidates to train a neural ranker. We refer to our approach as SCONER, scoring negative candidates before training neural ranker. Third, our experiments conclude that differentiating the negative candidate is helpful in the single-domain setting but not in the multi-domain setting. To the best of our knowledge, this is the first study that seeks to incorporate the "negativeness" of negative samples to train a neural ranker for ReQA tasks.

2 Related Work

Retrieval Based Question Answering ReQA is to identify sentences from large corpus that contain the answer to a question (Yang et al., 2015; Cakaloglu et al., 2020; Ahmad et al., 2019; Guo et al., 2020). It has practical applications such as s Googles Talk to Books[2]. ReQA is similar to Open Domain Question Answering (ODQA) but different in the following aspect, ReQA aims to build an efficient retrieval system, and the answer is a sentence or a short passage (Ahmad et al., 2019); while ODQA requires a retrieval system to find relevant documents at a large scale and a machine reading comprehension model to predict short answer span from documents (Bilotti et al., 2007; Chen and Van Durme, 2017; Chen et al., 2017; Min et al., 2019; Karpukhin et al., 2020). In this paper, we focus on the ReQA task and believe that building an efficient system for ReQA is also beneficial for the ODQA task. For example, QASC (Khot et al., 2020) requires retrieving sentences from a large corpus and composing them to answer a multiple-choice question, and a good ReQA system can be used to retrieve sentences in the first stage.

Neural Ranker Bag-of-words ranking models like BM25 (Robertson and Zaragoza, 2009) have been widely used for information retrieval for a long time. Although efficient, such methods depend on handcrafted features and can not be optimized on a specific task such as question answering, therefore, neural networks have been applied as re-rankers (Guo et al., 2016; Hui et al., 2017; Xiong et al., 2017; Dai et al., 2018; Mc-

[1] k value depends on evaluation methods or tasks.

[2] https://books.google.com/talktobooks/

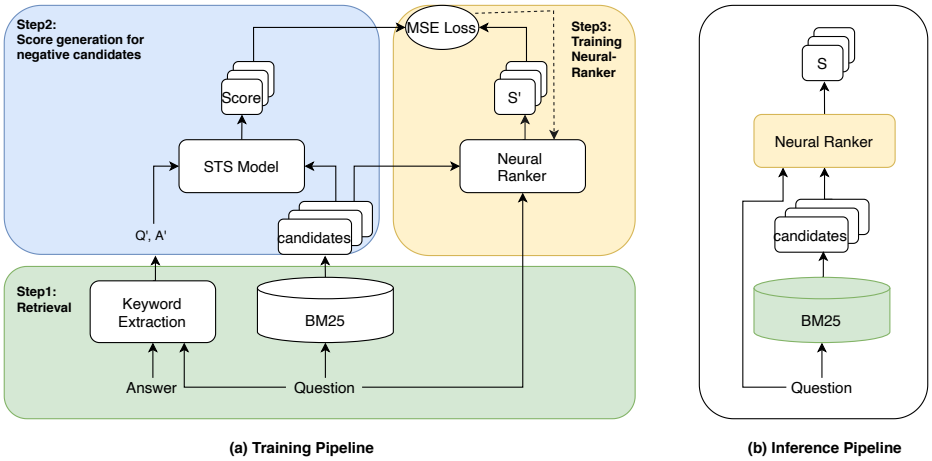

Figure 1: (a) Training Pipeline: Step1–retrieve negative candidates using BM25; Step2–use STS model to generate negativeness scores together with data augmentation; and Step3–train a neural ranker with the generated scores as labels. (b) Inference Pipeline: retrieve the top-200 candidates using BM25 and re-rank them using neural ranker. Q' and A' means modified questions and answers, S' means predicted scores of neural ranker.

Donald et al., 2018), also called as answer selection model in some work (Rao et al., 2016, 2019; Laskar et al., 2020). Recently, transformer-based models like BERT (Devlin et al., 2019) and RoBERTa (Liu et al., 2019) are widely used as re-rankers (Nogueira and Cho, 2019; Yilmaz et al., 2019; MacAvaney et al., 2019). Such re-rankers belong to the cross-attention architecture, which allows the rich interaction between the question and the candidate. Our work belongs to this style but has major difference with previous work, where neural rankers are trained using the same label (or score) for all negative candidates previously, in our work, we train a model using different scores for negative candidates.

3 Negative Candidate Scoring Approach

In this section, we review the concept of Semantic Textual Similarity (STS). Then, we describe the two stages of our scoring approach: (1) training an STS model, and (2) using it to generate negativeness scores for question-negative candidate pairs.

3.1 Review: Semantic Textual Similarity

STS determines how close two sentences are in terms of semantic meaning (Conneau and Kiela, 2018). Specifically, given two sentences, a high STS score indicates that they present a similar meaning, while a low score implies that they have different meaning. The score is a float number range in [0, 5]. Examples from STS-benchmark are given in Appendix A.

3.2 Training an STS-model

The goal of the first stage is to build a model that can generate a high score for a question-candidate pair if the candidate has similar information with the question; otherwise a low score. Furthermore, the model should understand the semantic meaning of the input rather than simply applying the word matching technique (e.g. BM25).

To achieve this goal, we train a regression model on the STS-benchmark, consisting of a RoBERTa model (Liu et al., 2019) and a Multi-Layer Perceptron (MLP) layer. In particular, the input to the RoBERTa model is `[CLS] sentence1 [SEP] sentence2 [SEP]`. Then we feed the representation of the `[CLS]` token to the MLP which predicts a score. We apply the Mean Squared Error (MSE) loss as the training objective to minimize the gap between the predicted score with the ground truth STS score. The figure of the model's input and output is given in Appendix B.

3.3 Negativeness Score Generation via Data Augmentation

In the second stage, we use the STS model to generate scores, where the `sentence1` is an augmented question and `sentence2` is a negative candidate. The intuition behind using augmented questions rather than the original question is that there is a difference between identifying semantic similarity and answer ranking. The former focuses on detecting meaning equivalence and the latter emphasizes keyword matching and semantic under-

standing. Thus, even though a candidate is relevant to a question, the STS model might not produce a high score due to the semantic meaning gaps between the candidate and the question. To illustrate this, consider a question "Beyonce has a fan base that is referred to as what?" and a good candidate "The name Bey Hive derives from the word beehive, purposely misspelt to resemble her first name, and was penned by fans after petitions on the online social networking service Twitter and online news reports during competitions," (the answer is *Bey Hive*), the STS model generate a low score for this pair as shown in the first row (Q) of Table 1.

To overcome this issue, we propose three ways to augment a question by using its answer. If any candidate has information semantically similar to the answer, the STS model is expected to generate a higher score for this candidate than other candidates which do not have any match (or less match) with the answer. Next, we present each augmentation approach and Table 1 shows examples of each approach and the corresponding generated score.

Question + Answer (Q+A) The first approach is to concatenate the answer to the original question.

Question + Keywords of Answer (Q+KA) The second approach is to extract the keywords from the answer and concatenate the keywords to the original question. We used Rapid Automatic Keyword Extraction (RAKE) (Rose et al., 2010) to extract the keywords. The intuition is that the answer might include irrelevant information. By extracting keywords, some irrelevant information can be removed, and we hypothesized that neglecting this distracting information can help the STS model generate a more reasonable negative score.

Keywords of Question and Answer (KQ+KA) This method extracts the keywords not only for the answer but also the question. Then we concatenate the keywords sequentially. The intuition is the same as the second approach and extends to the question as well.

4 Neural Ranker

Our neural ranker has the identical model architecture as the STS model (§3.2) but with different inputs i.e., the inputs of the STS model are a pair of sentences from the STS-benchmark, while the inputs of the neural ranker are question-candidate pairs. Next, we describe the training and inference pipelines of a neural-ranker.

4.1 Training Pipeline

The pipeline to train a neural ranker consists of three steps (the left part in Figure 1). *Step1 (the green block)*: we use BM25 to retrieve the top100 candidates for a question. From these candidates, we further randomly sample 10 negative candidates from the top-100. *Step2 (the blue block)*: we augment the question in one of the approaches described in §3.3 and use it as `sentence1`. Each negative candidate from step1 is used as `sentence2`. We feed the `sentence1` and `sentence2` to the STS model and obtain a score for the question and negative candidate pair. *Step3 (the yellow block)*: we train the neural ranker using the question and positive candidate pairs which are given in the training set and the question and negative candidate pairs which are given in step1. The neural ranker is trained by MSE loss as follows:

$$MSE = \frac{1}{n}\sum_{c^- \in Neg}^{n}(\hat{y_{c^-}} - neg_{c^-})^2 + \frac{1}{m}\sum_{c^+ \in Pos}(\hat{y_{c^+}} - pos_{c^+})^2, \quad (311)$$

where c^- is a negative candidate for a question q and neg_{c^-} is the generated score given in step2, c^+ is a positive candidate and pos_{c^+} is 5, $\hat{y_{c^-}}$ and $\hat{y_{c^+}}$ are the predicted scores given by the neural ranker for negative and positive samples respectively.

4.2 Inference Pipeline

During the inference time, for any given question, we retrieve the top-100 candidates using BM25. We then concatenate the question with every candidate and ask the neural ranker to predict a score for each candidate. Finally, we re-rank the top-100 candidates based on the neural ranker scores and select Top-K candidates as the final answer.

5 Experiments

5.1 Dataset

MultiReQA (Guo et al., 2020) has five different in-domain datasets and four out-of-domain datasets. The in-domain datasets include training and testing sets, there are SearchQA (Dunn et al., 2017), TriviaQA (Joshi et al., 2017), HotpotQA (Yang et al., 2018), SQuAD (Rajpurkar et al., 2016), and NaturalQuestions(NQ) (Kwiatkowski et al., 2019). The out-domain datasets includes DuoRC (Saha

Approach	Augmented Question	Score
Q	Beyoncé's has a fan base that is referred to as what?	2.20
Q+A	Beyoncé's has a fan base that is referred to as what? The Bey Hive is the name given to Beyoncé's fan base.	3.08
Q+KA	Beyoncé's has a fan base that is referred to as what? name given fan base bey hive beyoncé	2.86
KQ+KA	Fan base referred Beyoncé's name given fan base bey hive beyoncé	2.95

Table 1: The example of augmented question, where the original question is *Beyoncé's has a fan base that is referred to as what?*, and the given answer is *The Bey Hive is the name given to Beyoncé's fan base*. Each score is generated by the STS model given each Augmented Question and sentence "*The name Bey Hive derives from the word beehive, purposely misspelled to resemble her first name, and was penned by fans after petitions on the online social networking service Twitter and online news reports during competitions.*"

et al., 2018), RelationExtraction (Levy et al., 2017), TextbookQA (Kembhavi et al., 2017) and BioASQ (Tsatsaronis et al., 2015). The datasets come from a different domain like Wikipedia and Science, and require different reasoning skills like multihop reasoning and numerical reasoning. The domains and type of questions of each dataset are given in Appendix D. In MultiReQA, the corpus consists of sentences from supporting documents of the original dataset. The statistic of training and testing data are given in Appendix C.

5.2 Baselines

BM25 Following (Guo et al., 2020), we use the implementation of BM25 from (Rehurek and Sojka, 2010) with the default parameters and BERT tokenizer to create indexing for search.

Binary Classification Model (BCM) We use RoBERTa model as the encoder, which takes input as `[CLS] question [SEP] candidate [SEP]`. Then, we feed the vector representation of `[CLS]` to a linear layer with two logits as output: one represents the probability of candidates being irrelevant and the other represents it being relevant. We apply binary cross entropy loss to train this model. The training data is constructed by using the positive samples for each question, and we randomly selected the same amount of negative samples from the top-100 candidates given by BM25, where the label for the positive samples are 1, and negative samples are 0.

Regression Model (RM) This baseline is similar to the BCM baseline, but the linear layer only outputs one logit instead of two, thus it is a regression model rather than a binary classification model. We use MSE loss to train this model. The positive and negative samples are the same as BCM, but the positive samples have label 5, and the negative samples have label 0. We also use 1 as the label for positive samples but find that 5 yields better performance, thus we use label 5 to train RM baselines. Appendix F shows the results comparison between label as 1 and 5 for three datasets.

Triplet Model (TM) This baseline has the identical model architecture as the RM baseline, but we use the hinge loss to train the model in which more negative candidates can be used. Specifically, each training sample is a triplet, i.e., $\langle q, c^+, c^- \rangle$, where q is a question, c^+ is a positive candidate, and c^- is a negative candidate. Let $S(q, c)$ denote the score given by the model for question q and candidate c. The model is trained such that $S(q, c^+)$ is higher than $S(q, c^-)$. We use the same negative candidates as our SCONER to train the TM model.

5.3 Experiment Setup

We use Huggingface (Wolf et al., 2020) and Pytorch (Paszke et al., 2019) implementation for training each model. We initialize each model with pretrained RoBERTa-base parameters. To train the STS model, we use one GTX1080 GPU with maximum length (MaxL) 128, batch size (bs) 32, learning rate (lr) 2e-5, and 6 training epochs (epochs). For each neural ranker (including the baseline), we use four GTX1080 GPUs with MaxL 368, bs 16, lr 2e-5, 5 epochs, and gradient accumulation steps 2.

5.4 Results and Analysis

In all tables, NQ, HQA, SQA, and TQA stand for NaturalQuestions, HotpotQA, SearchQA and TriviaQA, respectively. We use two metric to evaluate each models, P@1 and MRR. The equations of calculating these two metric are given in Appendix E. In this section, we mainly compare P@1, but it is easy to see the same trend extended to MRR.

Comparison with Existing Method and Effect of Re-ranking In Table 2, we present two existing methods for MultiReQA. Guo et al. (2020) fine-tune BERT dual encoder or USE-QA (Yang et al., 2020a) on each in-domain dataset, where USE-QA

Metric	Model	MultiReQA					
		NQ	SQuAD	HQA	SQA	TQA	Avg.
	Existing Approach (without re-ranking)						
	Guo et al. (2020)	38.00	66.83	32.05	31.45	32.58	40.18
	Baselines						
	BM25	25.54	69.37	28.33	37.39	42.97	40.72
	BCM	46.07	83.71	76.60	65.48	62.05	66.78
	RM	44.76	85.36	70.61	69.79	60.41	66.19
	TM	50.33	85.65	70.00	73.03	65.43	68.89
P@1	*SCONER (Ours)*						
	Q	48.64	89.09	64.76	68.64	62.20	66.67
	Q+A	49.97	89.14	**79.80**	70.27	64.73	70.78
	Q+KA	50.87	**89.48**	71.71	**78.26**	65.16	71.10
	KQ+KA	52.80	88.37	76.28	75.64	**65.45**	**71.71**
	Existing Approach (without re-ranking)						
	Guo et al. (2020)	52.27	75.86	46.21	50.70	42.39	53.49
	Yang et al. (2020b)	65.90	73.70	-	-	-	-
	Baselines						
	BM25	37.66	75.95	49.99	55.62	55.19	54.88
	BCM	58.03	89.72	84.73	73.94	71.97	75.68
	RM	57.02	90.58	80.45	78.81	70.67	75.51
MRR	TM	60.87	90.27	81.00	82.22	**75.30**	77.93
	SCONER (Ours)						
	Q	58.46	92.51	70.73	76.64	68.94	73.46
	Q+A	60.14	92.36	**85.88**	78.62	72.48	77.90
	Q+KA	60.16	**92.71**	80.08	84.72	72.51	78.04
	KQ+KA	**61.50**	91.92	82.87	**83.02**	72.54	**78.37**

Table 2: Performance comparison of SCONER with existing models and standard baselines, evaluated in terms of P@1 and MRR on five benchmark datasets. **Bold** number means the best performance in the column of each block.

Metric	Model	MultiReQA					
		NQ	SQuAD	HQA	SQA	TQA	Avg.
	Baselines						
	BCM	46.38	86.33	77.49	70.41	61.35	68.39
	RM	47.15	86.57	74.71	70.15	61.34	67.98
	TM	51.64	86.67	68.57	69.37	63.64	67.98
P@1	*Neg-Ranker (Ours)*						
	Q	50.44	90.06	71.54	71.26	65.22	69.70
	Q+A	50.54	**89.97**	**77.63**	77.74	66.52	**72.48**
	Q+KA	51.72	89.34	74.51	**77.75**	**66.97**	72.06
	KQ+KA	**53.44**	89.43	77.25	75.92	66.07	72.42
	Baselines						
	BCM	58.02	91.30	**84.98**	78.70	71.11	76.82
	RM	58.46	91.35	83.16	78.47	71.07	76.50
	TM	61.57	91.10	79.80	79.33	73.99	77.16
MRR	*Neg-Ranker (Ours)*						
	Q	59.71	**92.96**	77.83	78.49	72.13	76.22
	Q+A	60.42	92.92	84.33	84.14	74.06	**79.17**
	Q+KA	61.21	92.45	82.00	**84.38**	**74.09**	78.83
	KQ+KA	**62.31**	92.62	83.77	82.74	73.25	78.94

Table 3: We initialize each model using the STS model. Comparing to Table 2, the performance improve in most cases. Green/red means improvement/decrease.

was pre-trained specifically for retrieval question answering tasks. This method and BM25 directly retrieve answers from the entire corpus without re-ranking. The other models (baselines and ours) re-rank candidates after retrieval. From the results, we see that the re-ranking phase improves the performance significantly, for example, compared to BM25, re-ranking improve P@1 at least ∼20%, ∼13%, ∼42%,∼38%, and ∼20% on NQ, SQuAD, HotpotQA, SearchQA and TriviaQA, respectively. Similarly, the re-ranking significantly improve the neural retrievers compared to retrievers proposed in Guo et al. (2020).

Comparison with Baselines Among the three re-ranking baselines, the TM model achieves the best performance, indicating that using more negative candidates and the ranking loss to train a model can yield better performance than the other two. More importantly, SCONERs achieve the best performance compared to baselines across all datasets in a single-domain (see in Table 2). Specifically, the largest gain SCONERs achieved is ∼13%, compared to BCM on SearchQA, and the largest average gain is ∼5.5%, compared to RM. When compared to the best baseline (TM), SCONERs achieve ∼2.5%, ∼4%, ∼3%, ∼5% P@1 improvement on NQ, SQuAD, HotpotQA, and SearchQA, respectively, and show similar performance for TriviaQA. This shows that while using more negative candidates is important to train neural rankers, differentiating the negative candidates can further improve the models' performance, which demonstrates the usefulness of our proposed scoring method of negative candidates.

Effect of Question Augmentation To show the effectiveness of the data augmentation proposed in §3.3, we also train neural rankers with scores generated by the STS model without augmentation (i.e. using question and negative candidate), termed as Q model. From Table 2, we can see that training neural rankers in such a way can not always beat baselines, for example, on NQ, SearchQA and TriviaQA, the performance of Q models is worse than TM models in terms of P@1 score, and Q models do not beat BCM and TM on an average. On the other hand, using augmentation methods, Q+A, Q+KA, and KQ+KA are better than Q models and beat all baselines on average (see Table 2). This demonstrates that while the STS model can be used to generate the scores of the negative candidate, it is important to incorporate the answer to the question in the generation process, demonstrating the importance of our proposed augmentation method.

Looking at the three data augmentation methods, they achieve similar performance on average, where KQ+KA achieves the best performance with a slight margin.

Effects of STS Model The intuition of using the STS Model to generate scores is that semantic similarity and answer selection are related because a model needs to understand the semantic meaning of the question and candidate in the later task. Thus, the knowledge of STS can be beneficial for

Metric	Model	MultiReQA (IID)						MultiReQA (OOD)				
		NQ	SQuAD	HQA	SQA	TQA	Avg.	DuoRC	R.E.	TextbookQA	BioASQ	Avg.
						Baselines						
	BCM	35.74	80.80	**73.99**	68.61	58.02	63.43	46.00	48.93	7.20	11.78	28.48
	RM	39.59	81.68	68.69	69.31	58.80	63.61	45.50	58.88	7.99	11.44	30.95
	TM	**55.14**	**91.66**	73.38	74.54	**66.87**	**72.32**	**49.50**	**67.06**	**10.26**	11.58	**34.60**
						SCONER (Ours)						
P@1	Q	45.17	87.45	69.93	72.18	63.79	67.70	43.00	53.96	9.86	**11.78**	29.65
	Q+A	43.63	83.47	67.24	71.50	63.33	65.83	41.50	60.03	9.33	11.51	30.59
	Q+KA	47.43	85.75	66.72	**75.01**	64.10	67.80	41.50	57.72	9.33	11.71	30.06
	KQ+KA	44.35	84.73	65.19	73.36	62.77	66.08	37.50	59.08	9.13	11.31	29.25
						Baselines						
	BCM	49.46	87.61	83.15	77.21	69.15	73.32	55.46	67.41	15.32	22.80	40.25
	RM	52.22	88.09	79.67	77.55	69.37	73.38	54.18	75.34	16.82	22.59	42.23
	TM	**64.29**	**94.25**	**83.67**	**83.53**	**76.38**	**80.42**	**59.04**	**81.82**	**20.79**	**23.25**	**46.23**
						SCONER (Ours)						
MRR	Q	55.35	91.25	76.99	79.75	71.92	75.05	50.64	72.81	19.22	22.03	41.18
	Q+A	54.51	88.16	75.83	79.09	71.27	73.77	46.89	75.39	17.76	21.26	40.32
	Q+KA	57.88	90.08	74.99	82.69	72.31	75.59	49.34	74.70	19.18	22.03	41.31
	KQ+KA	55.43	89.00	72.50	80.89	70.27	73.62	45.39	75.07	17.61	21.31	39.84

Table 4: Each model is trained with five in-domain datasets and tested on five IID datasets and four OOD datasets.

the answer ranking task. To further justify this intuition, instead of initializing a neural ranker with RoBERTa, we use the STS model. We expect to see that the STS model will be better than a RoBERTa model. Table 3 shows the results of fine-tuning an STS model by each training method, where the training data is the same as those in Table 2. We use green/red color to represent improvements/decrements compared to Table 2 (deeper color means more significant improvements/decrements). From Table 3, we can see that the STS model is better than the RoBERTa model in most cases, which justifies our intuition and to some extent explain why the proposed score generation approach can improve the model performance.

Multi-Domain Model and Generalization To see how well each model learns from multi-domain and how well they generalize to unseen data, we train different models using all in-domain datasets. We test the model on each in-domain data as well as out-of-domain datasets. Table 4 shows the result. For multi-domain learning, we see that our methods are better than two baselines, BCM and RM, by $\sim 4\%$ in terms of P@1 score. For out-of-domain performance, our models are better than BCM. TM achieves the best performance in both IID and OOD. One potential reason why TM is better than SCONERs is that TM has more training samples. Although both SCONER and TM use 10 negative candidates per question, in TM we pair the positive answer with each negative answer, i.e. if there are $\mathbf{x}$ number of positive candidate for one question, TM has $10\,\mathbf{x}$ training samples but SCONER has $10 + \mathbf{x}$ training samples. On the other hand, because of this, the training time of SCONER is much less than the TM model.

6 Ablation Study

Size of Negative Candidate The proposed scoring function allows us to use a different number of negative candidates for training. Here, we investigate the effect of the number of negative candidates per question used in training. We study 1/3/5/7/10 negative candidates per question on the MultiReQA-SQuAD dataset. Figure 2 shows the P@1 score associated with each method when the number of negative candidates increases. We have three observations: (1) compared to one negative candidate per question, more negative candidates are better, and this demonstrates that using more negative candidates indeed help the model to select the best answer; (2) when using only one negative candidate and comparing to the BCE baseline which also uses one negative candidate, all of our models perform better than the BCE baseline, this demonstrates that a regression model with differentiable values for negative candidates is better than a binary classification model; and (3) compared to the TM baseline, which uses 10 negative candidates per question, all four models performs better than the baseline even though using less negative candidates (e.g. 3, 5, and 7) for the SQuAD dataset (see Figure 2), this again demonstrates that using different labels for the negative candidate is an effective way to train a neural ranker.

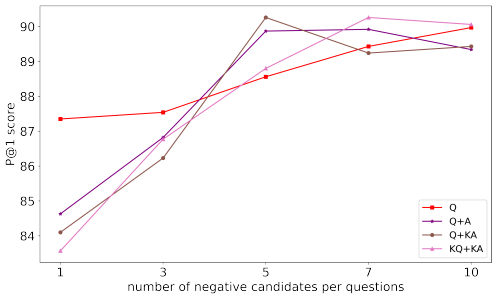

Figure 2: P@1 score regarding to the number of negative candidates per question used in the training.

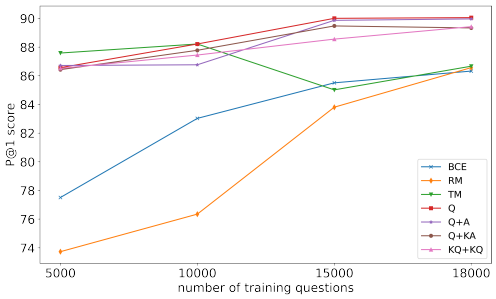

Figure 3: P@1 score regarding to different training size of questions used in the training.

Size of Training Samples We are interested in the question: does our method helps in low resource scenario? To answer this question, we use 5/10/15/18K questions to train our models and the baselines on the SQuAD dataset. For TM and our models, each question is paired with 10 negative candidates. Figure 3 illustrates the P@1 w.r.t different numbers of training questions. From Figure 3, we see that TM and our models are much better than BCE and RM when the training data only includes 5000 questions. In general, the improvement gap is wider as the training size is reduced, this suggests that it is useful in improving performance to use more negative candidates. However, it is especially important in low-resource scenarios.

Time Efficiency As mentioned before, our neural rankers belong to cross-attention model architecture. Although it has shown better performance, since it has to compute the attention score of every combination of input words at every layer, the inference time is long. Re-ranking fewer candidates can speed up the inference time. Here, we study the time efficiency by comparing the inference time of re-ranking 50/100/150/200 candidates. We use batch size 16 in inference time, which is the largest size that our machine allows. For each question, the inference time of re-ranking 50/100/150/200 candidates is 0.49/0.85/1.24/1.63 seconds. We also investigate whether re-ranking more candidates can yield better performance than less. Surprisingly, the performance of re-ranking 50/100/150/200 candidates does not exhibit a noticeable difference. We further find that the reason for this is that the recall of the top-50 given by BM25 is already as good as the top 100/150/200. This suggests that rather than using a large size of re-ranking, the recall of the initial retrieval module can be a good indicator of how many candidates should be re-ranked so that we can achieve the best trade-off between time efficiency and performance.

7 Future Directions

Currently, our pipeline relies on BM25 to retrieve initial candidates, in recent days, there has been a growing interest in building a neural model as a direct retriever (Zamani et al., 2018; Dai and Callan, 2019; Lee et al., 2019; Chang et al., 2020). DPR (Karpukhin et al., 2020) is one of such neural retriever which can be trained on down-stream tasks like question answering and thus can be optimized in a specific domain. DPR use in-batch negative candidates to train a neural retriever. One future work is to apply our negative scoring approach to score negative candidates and use them as the labels to train DPR.

8 Conclusion

While standard training methods take all negative candidates equivalent to train a neural ranker, we argue that different candidates should have different negativeness scores based on their semantic relevancy to the question. Motivated by this, we present SCONER, a new pipeline to train neural rankers by generating scores for negative candidates which is based on the semantic meaning between question-candidate pairs. This proposed method gives an advantage in terms of using more negative samples and making them differentiable. Our experimental results show that SCONER outperforms all standard training methods in single-domain setting, and most methods in multi-domain setting. Also, the ablation study demonstrates the usefulness of SCONER in both low and high resource scenario. Our detailed analysis demonstrates the efficacy of SCONER.

References

Amin Ahmad, Noah Constant, Yinfei Yang, and Daniel Cer. 2019. ReQA: An evaluation for end-to-end answer retrieval models. In *Proceedings of the 2nd Workshop on Machine Reading for Question Answering*, pages 137–146, Hong Kong, China. Association for Computational Linguistics.

Matthew W Bilotti, Paul Ogilvie, Jamie Callan, and Eric Nyberg. 2007. Structured retrieval for question answering. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 351–358.

Tolgahan Cakaloglu, Christian Szegedy, and Xiaowei Xu. 2020. Text embeddings for retrieval from a large knowledge base. In *International Conference on Research Challenges in Information Science*, pages 338–351. Springer.

Wei-Cheng Chang, F. Yu, Yin-Wen Chang, Yiming Yang, and S. Kumar. 2020. Pre-training tasks for embedding-based large-scale retrieval. *ArXiv*, abs/2002.03932.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading Wikipedia to answer open-domain questions. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1870–1879, Vancouver, Canada. Association for Computational Linguistics.

Tongfei Chen and Benjamin Van Durme. 2017. Discriminative information retrieval for question answering sentence selection. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 719–725, Valencia, Spain. Association for Computational Linguistics.

D. Cohen, Liu Yang, and W. Croft. 2018. Wikipassageqa: A benchmark collection for research on non-factoid answer passage retrieval. *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*.

Alexis Conneau and Douwe Kiela. 2018. Senteval: An evaluation toolkit for universal sentence representations. *ArXiv*, abs/1803.05449.

Zhuyun Dai and J. Callan. 2019. Context-aware sentence/passage term importance estimation for first stage retrieval. *ArXiv*, abs/1910.10687.

Zhuyun Dai, Chenyan Xiong, J. Callan, and Zhiyuan Liu. 2018. Convolutional neural networks for soft-matching n-grams in ad-hoc search. *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*.

J. Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*.

M. Dunn, Levent Sagun, M. Higgins, V. U. Güney, Volkan Cirik, and Kyunghyun Cho. 2017. Searchqa: A new q&a dataset augmented with context from a search engine. *ArXiv*, abs/1704.05179.

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*, pages 1–13, Hong Kong, China. Association for Computational Linguistics.

J. Guo, Y. Fan, Qingyao Ai, and W. Croft. 2016. A deep relevance matching model for ad-hoc retrieval. *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*.

Mandy Guo, Yinfei Yang, Daniel Matthew Cer, Qinlan Shen, and Noah Constant. 2020. Multireqa: A cross-domain evaluation for retrieval question answering models. *ArXiv*, abs/2005.02507.

Kai Hui, Andrew Yates, Klaus Berberich, and Gerard de Melo. 2017. PACRR: A position-aware neural IR model for relevance matching. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1049–1058, Copenhagen, Denmark. Association for Computational Linguistics.

Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *ACL*.

V. Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Yu Wu, Sergey Edunov, Danqi Chen, and Wen tau Yih. 2020. Dense passage retrieval for open-domain question answering. *ArXiv*, abs/2010.08191.

Aniruddha Kembhavi, Minjoon Seo, Dustin Schwenk, Jonghyun Choi, Ali Farhadi, and Hannaneh Hajishirzi. 2017. Are you smarter than a sixth grader? textbook question answering for multimodal machine comprehension. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5376–5384.

Tushar Khot, Peter Clark, Michal Guerquin, P. Jansen, and A. Sabharwal. 2020. Qasc: A dataset for question answering via sentence composition. In *AAAI*.

T. Kwiatkowski, J. Palomaki, Olivia Redfield, Michael Collins, Ankur P. Parikh, C. Alberti, D. Epstein, Illia Polosukhin, J. Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Q. Le, and Slav Petrov. 2019. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466.

Md Tahmid Rahman Laskar, Jimmy Xiangji Huang, and Enamul Hoque. 2020. Contextualized embeddings based transformer encoder for sentence similarity modeling in answer selection task. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 5505–5514, Marseille, France. European Language Resources Association.

Kenton Lee, Ming-Wei Chang, and Kristina Toutanova. 2019. Latent retrieval for weakly supervised open domain question answering. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6086–6096, Florence, Italy. Association for Computational Linguistics.

Omer Levy, Minjoon Seo, Eunsol Choi, and Luke Zettlemoyer. 2017. Zero-shot relation extraction via reading comprehension. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 333–342, Vancouver, Canada. Association for Computational Linguistics.

Y. Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, M. Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *ArXiv*, abs/1907.11692.

Ji Ma, I. Korotkov, Yinfei Yang, K. Hall, and R. McDonald. 2020. Zero-shot neural passage retrieval via domain-targeted synthetic question generation.

Sean MacAvaney, Andrew Yates, Arman Cohan, and Nazli Goharian. 2019. Cedr: Contextualized embeddings for document ranking. *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*.

Ryan McDonald, George Brokos, and Ion Androutsopoulos. 2018. Deep relevance ranking using enhanced document-query interactions. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1849–1860, Brussels, Belgium. Association for Computational Linguistics.

Sewon Min, Danqi Chen, Luke Zettlemoyer, and Hannaneh Hajishirzi. 2019. Knowledge guided text retrieval and reading for open domain question answering. *ArXiv*, abs/1911.03868.

Rodrigo Nogueira and Kyunghyun Cho. 2019. Passage re-ranking with bert. *ArXiv*, abs/1901.04085.

Ibrahim Burak Ozyurt, Anita Bandrowski, and Jeffrey S Grethe. 2020. Bio-answerfinder: a system to find answers to questions from biomedical texts. *Database*, 2020.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

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

Jinfeng Rao, Hua He, and Jimmy Lin. 2016. Noise-contrastive estimation for answer selection with deep neural networks. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, pages 1913–1916.

Jinfeng Rao, Linqing Liu, Yi Tay, Wei Yang, Peng Shi, and Jimmy Lin. 2019. Bridging the gap between relevance matching and semantic matching for short text similarity modeling. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5370–5381, Hong Kong, China. Association for Computational Linguistics.

Radim Rehurek and P. Sojka. 2010. Software framework for topic modelling with large corpora.

S. Robertson and H. Zaragoza. 2009. The probabilistic relevance framework: Bm25 and beyond. *Found. Trends Inf. Retr.*, 3:333–389.

Stuart Rose, Dave Engel, Nick Cramer, and Wendy Cowley. 2010. Automatic keyword extraction from individual documents. *Text mining: applications and theory*, 1:1–20.

Amrita Saha, Rahul Aralikatte, Mitesh M. Khapra, and Karthik Sankaranarayanan. 2018. DuoRC: Towards complex language understanding with paraphrased reading comprehension. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1683–1693, Melbourne, Australia. Association for Computational Linguistics.

George Tsatsaronis, Georgios Balikas, Prodromos Malakasiotis, Ioannis Partalas, Matthias Zschunke, Michael R Alvers, Dirk Weissenborn, Anastasia Krithara, Sergios Petridis, Dimitris Polychronopoulos, et al. 2015. An overview of the bioasq large-scale biomedical semantic indexing and question answering competition. *BMC bioinformatics*, 16(1):1–28.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame,

Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Chenyan Xiong, Zhuyun Dai, J. Callan, Zhiyuan Liu, and R. Power. 2017. End-to-end neural ad-hoc ranking with kernel pooling. *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*.

Yi Yang, Wen-tau Yih, and Christopher Meek. 2015. WikiQA: A challenge dataset for open-domain question answering. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2013–2018, Lisbon, Portugal. Association for Computational Linguistics.

Yinfei Yang, Daniel Cer, Amin Ahmad, Mandy Guo, Jax Law, Noah Constant, Gustavo Hernandez Abrego, Steve Yuan, Chris Tar, Yun-hsuan Sung, Brian Strope, and Ray Kurzweil. 2020a. Multilingual universal sentence encoder for semantic retrieval. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 87–94, Online. Association for Computational Linguistics.

Yinfei Yang, Ning Jin, Kuo Lin, Mandy Guo, and Daniel Cer. 2020b. Neural retrieval for question answering with cross-attention supervised data augmentation. *arXiv preprint arXiv:2009.13815*.

Z. Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, R. Salakhutdinov, and Christopher D. Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *EMNLP*.

Zeynep Akkalyoncu Yilmaz, S. Wang, W. Yang, Haotian Zhang, and Jimmy Lin. 2019. Applying bert to document retrieval with birch. In *EMNLP/IJCNLP*.

Hamed Zamani, M. Dehghani, W. Croft, E. Learned-Miller, and J. Kamps. 2018. From neural re-ranking to neural ranking: Learning a sparse representation for inverted indexing. *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*.

A Examples of STS-B

Table 5 shows two pairs of sentences with score 0 and 5 from the STS-B dataset. Score 5 means two sentences are semantically equivalent and score 0 means semantically irrelevant. In STS-B dataset, the scores are range from [0, 5].

Sentence 1	Sentence 1	Score
A man is playing a guitar.	A man plays the guitar.	5.0
A young man is playing the piano.	A woman is peeling a prawn.	0.0

Table 5: Two examples from the STS-benchmark, the first pair of sentences have highest score since they are highly similar, while the second pair have lowest score because they have totally different meaning.

B Figure of STS-Model

Figure 4 shows the architecture of the models used in all experiments, except for binary classification model where the output of MLP is two logits. Our model consists of two parts. A RoBERTa model takes the concatenation of two sentences as input and output the contextual representation of `[CLS]` token. An MLP layer takes this representation as input and output a score.

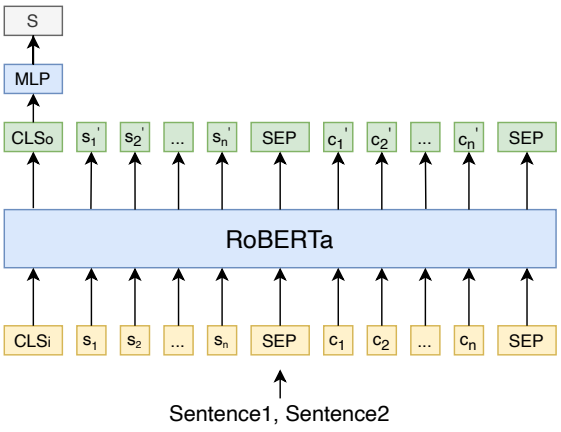

Figure 4: The structure of the STS-model, where S_i are tokens from Sentence1, and C_i are tokens from Sentence2.

C Data Processing and Statistic

The dataset includes two parts, question-answer pairs and a corpus. The question-answer pairs are from MRQA (Fisch et al., 2019). MRQA is a collection of extractive QA task where the goal is to extract an answer span given a question and a context. The corpus is given by MultiReQA. Particularly, to convert extractive QA task to ReQA task, where the context is not given, Guo et al. (2020) divide the context into single sentences and combine all sentence to construct a corpus. The goal is to retrieve the answer to a question from the corpus. We refer reader to see the details of processing of the corpus in Guo et al. (2020). We remove all questions that do not have any answer in the corpus in both training and testing sets. Table 6 shows the number of questions, the number of candidates (the size of the corpus) and the average number of answers per questions of each dataset. The average number of answers for SearchQA and TriviaQA are more than others.

D Questions Domain and Reasoning Type

Table 7 shows the domains and reasoning type of each dataset. We see that the MultiReQA include datasets from different domains and different types of reasoning skill required to answer the question.

E Evaluation Metrics

We present two evaluation metrics as follows.

Precision@K P@K reveals the proportion of top-K retrieved candidates that are relevant. R@K reveals the proportion of relevant documents are in the top-K retrieved candidates. In Eq 1, N is the number of questions, A_K are the top-K retrieved answer, A^* is correct answers.

$$P@K = \frac{1}{N}\sum_{i}^{N}\frac{|A_K \cap A^*|}{K} \quad (1)$$

Mean Average Precision P@K does not take the position of relevant candidates into account, which means a system that ranks the relevant answer higher than another system can not be identified as better. MAP address this issue, computed as follows, where in Eq 2, $Rel@i$ is 1 if the i^{th} answer is correct, 0 otherwise.

$$AveP@K = \frac{1}{|A^*|}\sum_{i=1}^{i=K} P@i \times Rel@i, \quad (2)$$

$$MAP@K = \frac{1}{N}\sum_{q=1}^{q=N} AveP@K(q) \quad (3)$$

MRR The MRR score is computed as follows,

$$MRR = \frac{1}{N}\sum_{i}^{N}\frac{1}{rank_i},$$

where $rank_i$ is the rank of the first relevant answer.

Dataset	Train			Test		
	Ques.	Cand.	Avg. ans. per ques.	Ques.	Cand.	Avg. ans. per ques.
SearchQA	24793	Cand.	6.00	16883	454836	6.66
TriviaQA	61688	1893674	6.00	7776	238339	6.0
HotpotQA	72519	508879	1.50	5860	52191	1.74
SQuAD	18768	95659	1.04	2063	10642	3.44
NQ	102577	71147	1.21	3892	22118	1.31
DuoRC	-	-	-	200	5525	3.47
R.E.	-	-	-	3301	2945	1.00
TextbookQA	-	-	-	1501	71147	3.32
BioASQ	-	-	-	1503	14158	2.91

Table 6: Statistic of MultiReQA datasets

Dataset	Domain	Type
NQ	Wikipedia	single-hop
SQuAD	Wikipedia	single-hop
HQA	Wikipedia	multi-hop
TQA	Trivia and quiz-league websites	single-hop
SQA	Jeopardy! TV show	single-hop
DuoRC	wikipedia	numerical reasoning
RE	Wikiread	
TbQA	Lessons from middle school Life Science, Earth Science, and Physical Science textbooks	single-hop
BioASQ	Science (PubMed) articles	single-hop

Table 7: The domain and reasoning type of each dataset.

F Regression Model Baseline

Here, we explore two labels for regression models, 0 and 5 on three datasets. We train each model by initializing it with RoBERTa. Table 8 shows the results, and we found that label 5 is better than 0 in three cases.

Label	Dataset		
	SQuAD	HotpotQA	NQ
1	64.84	43.50	70.00
5	85.36	44.76	70.61

Table 8: Comparison of two regression models with label 1 and 5 in terms of P@1.