# SCONER: Scoring Negative Candidates Before Training Neural Re-Ranker For Question Answering

**Man Luo** [1] **Mihir Parmar** [1] **Jayasurya Sevalur Mahendran** [1] **Sahit Jain** [1] **Chitta Baral** [1]

## Abstract

A neural re-ranker aims to re-scores a set of candidates given by a search engine. It is crucial to obtain good performance on many down-stream tasks such as retrieval-based question answering (ReQA). In this work, we introduce a scoring function for negative candidates to train a neural re-ranker and compare models trained by our approach with three baselines on a range of ReQA tasks. We term our approach as SCONER—scoring negative candidates before training neural re-ranker, which includes 1) a scoring function based on the concept of Semantic Textual Similarity (STS) and data augmentation; and 2) a neural re-ranker trained on data using generated negativeness scores as labels. Experimental results show that SCONER outperforms three baselines by up to 13% absolute improvement on the SearchQA dataset and 5.5% on average across all datasets in terms of P@1. SCONER demonstrates that using different negativeness scores to train a neural-ranker is better than a single score, and we present a simple yet efficient way to generate the scores.

## 1. Introduction

Due to the wide applications in real-world, Retrieval Based Question Answering (ReQA has gained increasing interest and attention in recent years, and many benchmarks have been proposed for Retrieval Based Question Answering (ReQA) (Cohen et al., 2018; Khot et al., 2020; Ahmad et al., 2019; Guo et al., 2021). A promising approach for solving ReQA involves two stages: (1) retrieve a small set of candidates from a large corpus and (2) re-rank these candidates. The re-ranking stage can significantly improve the initial retrieval performance (Ozyurt et al., 2020), and thus it is crucial for any retrieval system (Ma et al., 2021).

**Question**: The manager who recruited David Beckham managed Manchester United during what time frame?
**Answer**: Sir Alexander Chapman Ferguson, CBE (born 31 December 1941) is a Scottish former football manager and player who managed Manchester United **from 1986 to 2013**.
**S1**: Instead, he had drafted in young players like Nicky Butt, David Beckham, Paul Scholes and the Neville brothers, Gary and Phil.
**S2**: The awards ceremony was held at Earls Court in London for the last time.

*Figure 1.* An example from the HotpotQA dataset. While both S1 and S2 are negative candidates to the question, our approach assigns higher negativeness score to S1 than S2.

Large Pretrained Language Models (PrLMs) have been widely used as neural re-rankers (Yilmaz et al., 2019; Nogueira & Cho, 2019) [1]. In most cases, the negative examples used to train the re-ranker are assigned with the same label. However, we argue that some candidates may be more negative than others and should be treated differently. Figure 1 shows an example from HotpotQA dataset (Yang et al., 2018) to illustrate this argument. In this example, neither S1 nor S2 contains the correct answer; yet S1 mentions a key entity in the question (David Beckham), while S2 has no common entity with the question. From the human perspective, S1 should have a higher score than S2.

It leads us to ask a question - "is having different levels of negativeness beneficial for training neural re-rankers?" Driven by this question, we propose an approach for scoring negative candidates (§2). Our approach has two stages. First, we train a model on STS benchmark (Conneau & Kiela, 2018). This model generates a high score for a two-sentence pair if they are semantically similar; otherwise, a low score. Second, we use this STS model to generate scores for the question and negative candidate pairs. In this way, we obtain a set of question-candidate pairs with labels in the continuous range of $[0, 5]$ as opposed to previous works where labels are binary. Furthermore, we want the generated score for a negative candidate to be higher than others if the first candidate has more relevant information

[1]School of Computing AI, Arizona State University, Tempe, AZ. Correspondence to: Man Luo <mluo26@asu.edu>.

ICML workshop on Knowledge Retrieval and Language Models, Baltimore, Maryland, USA, 2022. Copyright 2022 by the author(s).

[1]More related work can be found in Appendix A

to the answer. To achieve this goal, we explore three data augmentation techniques (§2.2). Such scoring approach allows: 1) a good candidate that is not annotated as an answer to have a high score, 2) more negative samples to be used to train a neural re-ranker, and 3) negative candidates to be differentiated using "negativeness" scores. In this paper, negativeness score means the score for a question and a negative candidate; and a higher score means the negative candidate contains more information to answer the question.

We compare three standard training strategies and our proposed method on the MultiReQA (Guo et al., 2021) benchmark, which includes five training datasets across different domains. Based on our experiments, we observe that 1) our proposed approach outperforms three baselines by up to 13% absolute improvement on the SearchQA dataset and 5.5% on average across all datasets in terms of P@1; 2) use of a different negativeness score achieves better performance than the same score even when fewer negative candidates are used; and 3) our proposed method has a significant advantage in a low resource setting. These lead to the answer to the question that use of a negativeness score is an efficient way to train a neural re-ranker.

## 2. Negative Candidate Scoring Approach

The key idea of the negative candidate scoring approach is to utilize a Semantic Textual Similarity (STS) model and the motivation is that STS score determines how close two sentences are in terms of semantic meaning (Conneau & Kiela, 2018) (see Appendix B for details). In the following, we describe the two stages of our scoring approach: (1) training an STS model, and (2) using it to generate negativeness scores for the question and negative candidate pairs.

### 2.1. Training an STS-model

We train an STS model on the STS benchmark, which is a regression model 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 layer which predicts a score (see the figure of the model's input and output in Appendix C). Mean Squared Error (MSE) loss is taken as the training objective to minimize the gap between the predicted score with the ground truth STS score.

### 2.2. Negativeness Score Generation

We use the STS model to generate scores for the question and negative candidate pairs. Due to the fact that sometimes, the important information is only presented in the answer but not in the question, even though a candidate is relevant to a question, the STS model might not produce a high score. To overcome this issue, we introduce three ways to augment a question to consider the answer in the scoring process. We expect that if two candidates have similar information regarding a question, but one has more similar information to the answer than the other, then the first one should obtain a higher score. Next, we present each augmentation approach (examples can be found in Table 5 in Appendix D).

**Question + Answer (Q+A)** The first approach is to simply 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 use Rapid Automatic Keyword Extraction (RAKE) (Rose et al., 2010) to extract the keywords. We believe that answer might include irrelevant information and it can be removed By extracting keywords. Neglecting irrelevant information can help the STS model generate a reasonable negative score.

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

## 3. SCONER: A Neural re-ranker

Generated negative scores are used in training our model, i.e., SCONER. The model has the identical model architecture as the STS model (§2.1) 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 SCONER are question-candidate pairs.

### 3.1. Training Pipeline

The pipeline to train SCONER consists of three steps (the left part in Figure 2). *Step1 (the green block)*: we use BM25 to retrieve the top-100 candidates for a question. From these candidates, we further randomly sample 10 negative candidates. *Step2 (the blue block)*: we augment the question by one of the approaches described in §2.2 and use it as `sentence1` and 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 SCONER 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 step 1. We use MSE loss to optimize our model. Since the upper bound of a negative score is 5 given by the STS model, we assign 5 to all positive candidates as a positive candidate should have a higher score than any negative candidate.

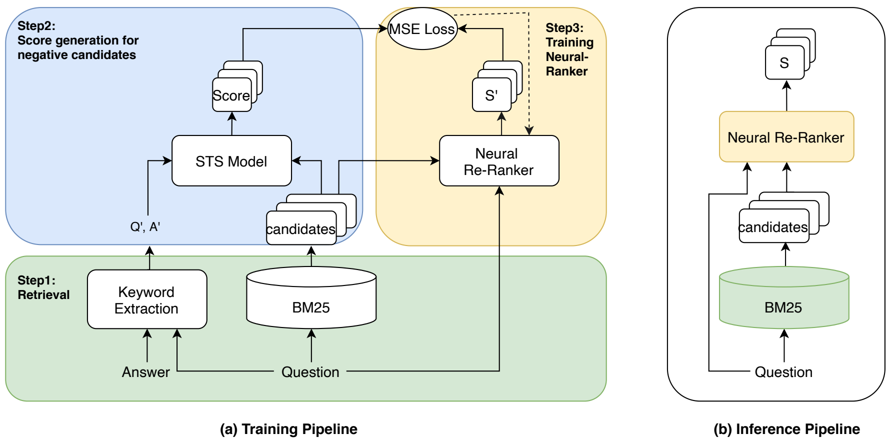

*Figure 2.* (a) Training Pipeline: Step1–retrieve negative candidates for a question using BM25; Step2–use a frozen STS model to generate negativeness scores for a question and candidate pair; and Step3–train a neural re-ranker using the generated scores given by the STS model. (b) Inference Pipeline: retrieve the top-100 candidates using BM25 and re-rank them using neural re-ranker. Q' and A' means augmented questions and answers, S' means predicted scores of neural re-ranker.

### 3.2. 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 SCONER to predict a score for each candidate. Finally, we re-rank the top-100 candidates based on the predicted scores and select top-k candidates as the final answer.

## 4. Experiments and Results

### 4.1. Dataset and Baselines

We conduct experiments on MultiReQA benchmark (Guo et al., 2021) which includes five training datasets: SearchQA (SQA) (Dunn et al., 2017), TriviaQA (TQA) (Joshi et al., 2017), HotpotQA (HQA) (Yang et al., 2018), SQuAD(Rajpurkar et al., 2016), and NaturalQuestions(NQ)(Kwiatkowski et al., 2019). More details and the statistics of each dataset are given in Appendix E.

We compare our proposed approach with three commonly used neural model baselines: Binary Classification Model (BCM), Regression Model (RM), and Triplet Model (TM). More details about baselines can be found in Appendix F. All baseline models and our proposed models have the same size and are based on the RoBERTa base pretrained model. Experimental setup can be found in Appendix G.

### 4.2. Results and Analysis

We use two standard metrics to evaluate each model defined by MultiReQA, P@1 and MRR (see Appendix H). In the following, we mainly describe P@1, however, it is easy to see the same trend extended to MRR.

**Comparison with Baselines** Table 1 shows the SCONER outperforms all baselines across all datasets. The largest gain SCONER achieved is ∼13%, compared to BCM on SearchQA, and the largest average gain is ∼5.5%, compared to RM. While compare to the strongest baseline, i.e., TM (since TM outperforms other two baselines), SCONER achives ∼2.5%, ∼4%, ∼3%, and ∼5% improvement in terms of P@1 on NQ, SQuAD, HotpotQA, and SearchQA, respectively, and outperforms TM on TriviaQA by a small margin. This shows that using more negative candidates and differentiating the negative candidates are important to boost the models' performance.

**Comparison with Existing Methods** The existing methods on MultiReQA directly retrieve answers from the entire corpus without re-ranking. We present one term-matching (e.g. BM25) and two neural-retrieval based methods from Guo et al. (2021), which are fine-tuned BERT dual encoder and USE-QA (Yang et al., 2020) on each in-domain dataset. Other baselines and our model re-rank candidates after BM25 retrieval. From the results, we see that the re-ranking phase improves the performance significantly. For instance, re-ranking improve P@1 at least ∼20%, ∼13%, ∼42%,∼38%, and ∼20% on NQ, SQuAD, HotpotQA, SearchQA and TriviaQA, respectively, compared to BM25.

**Effect of Data Augmentation** We also train neural re-rankers with scores generated by the STS model without augmentation (Q model) to see the effect of augmentation. From Table 1, we can see that baselines outperform Q model in most cases. For example, the performance of Q models is worse than TM on NQ, SearchQA and TriviaQA. Moreover, BCM and TM outperforms the Q model on average. On the other hand, using augmentation methods, Q+A, Q+KA,

| Metric | Model | MultiReQA | | | | | |
|---|---|---|---|---|---|---|---|
| | | NQ | SQuAD | HQA | SQA | TQA | Avg. |
| | *Existing Approach (without re-ranking)* | | | | | | |
| | BM25 | 25.54 | 69.37 | 28.33 | 37.39 | 42.97 | 40.72 |
| | USE-QA | 38.00 | 66.83 | 31.71 | 31.45 | 32.58 | 40.11 |
| | BERT | 36.22 | 55.13 | 32.05 | 30.20 | 29.11 | 36.54 |
| | *Baselines* | | | | | | |
| | 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 |
| P@1 | TM | 50.33 | 85.65 | 70.00 | 73.03 | 65.43 | 68.89 |
| | *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)* | | | | | | |
| | BM25 | 37.66 | 75.95 | 49.99 | 55.62 | 55.19 | 54.88 |
| | USE-QA | 52.27 | 75.86 | 43.77 | 50.70 | 42.39 | 53.00 |
| | BERT | 52.02 | 64.74 | 46.21 | 47.08 | 41.34 | 50.28 |
| | *Baselines* | | | | | | |
| | 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 1.* **Bold** number means the best performance in the column of each block. SCONER outperform all baselines. In addition, generating negativeness score using data augmentation is important to yield good performance.

and KQ+KA are better than Q models and outperform all baselines on average. While three proposed data augmentation all outperform baselines, KQ+KA is the best technique not only because it achieves the best average score but also because it performs more stable across all datasets. This demonstrates that it is important to incorporate the answer to the question in the negative score generation process.

**Effect of Size of Negative Candidates** Here, we compare using 1/3/5/7/10 negative candidates per question to train MultiReQA-SQuAD SCONER. Figure 3 shows the trend of P@1 score with different numbers of negative candidates for each method. We see that for three out of four methods, 5 or 7 already yields best performance, which suggests that SCONER does not need to be trained with many negative candidates. In addition, we have two observations: 1) compared to 1 negative candidate per question, 5/7/10 always yield better performance, and this suggests that using one negative candidate is not enough to train SCONER; 2) compared to the TM baseline, which uses 10 negative candidates per question, all four models perform better than TM even though using less negative candidates (e.g. 3 and 5), this demonstrates that using different negativeness scores is an effective way to train a neural re-ranker.

**Effect of Size of Training Data** We use 5/10/15/18 thousands (K) 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 4 illustrates

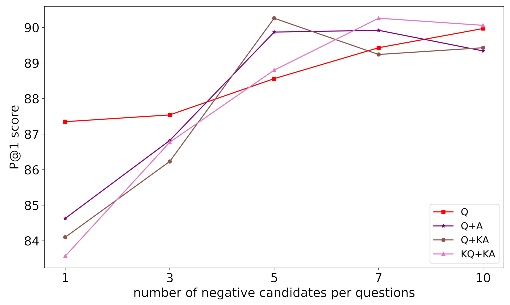

*Figure 3.* P@1 score regarding to the number of negative candidates per question used in the training. Each model is initialized with the STS model.

P@1 w.r.t different numbers of training questions. From Figure 4, we see that (1) except for TM, the other two baselines and our SCONER get improvement as the training size increases, (2) SCONER and TM perform well even with small amount of training data (e.g. 5K), but BCM and RM are much worse, and (3) SCONER always performs better than BCM and RM, and the advantage of SCONER is more significant in low resource settings.

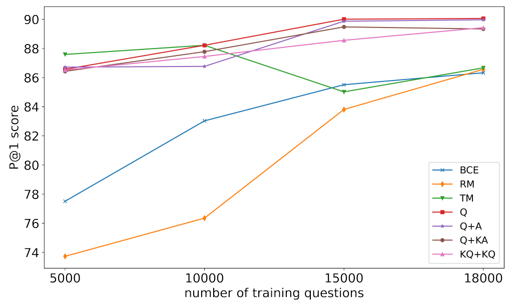

*Figure 4.* P@1 score regarding different training sizes of questions used in the training.

## 5. Ablation Study

We present most insightful studies in this section and other ablation studies can be found in Appendix I.

### 5.1. What are the Effects of STS Model?

STS scores can be used to approximate the scores for question and candidates because STS and question-candidate ranking are related so that these two tasks require similar knowledge or skill to solve. To further justify this intuition, we use the STS model to initialize a re-ranker rather than using the RoBERTa weights. We expect to see that the STS model will be better than a RoBERTa

| Metric | Model | MultiReQA | | | | | |
|---|---|---|---|---|---|---|---|
| | | NQ | SQuAD | HQA | SQA | TQA | Avg. |
| | | *Baselines* | | | | | |
| P@1 | 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 |
| | | *SCONER (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* | | | | | |
| MRR | 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 |
| | | *SCONER (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 2.* We repeat the experiments in Table 1 but initialize each model using the STS model. We use green color to indicate increasement compared to the corresponding result in Table 1, and red for decrease. In most cases, the STS model is better than RoBERTa.

model. We repeat the experiments in Table 1 but use the STS model rather than RoBERTa model to initialize each neural re-ranker and present the results in Table 2. We use green/red color to represent improvements/decrements compared to Table 1 (deeper color means more significant improvements/decrements). From Table 2, we can see that the STS model is better than the RoBERTa model in most cases, which justifies our intuition and to some extent explains why the proposed score generation approach can improve the model performance.

### 5.2. Can SCONER be Applied to Positive Candidates?

In our previous pipeline, SCONER only uses the generated scores for negative candidates. Here, we also use the STS model to generate scores for positive candidates and use them to train re-ranker instead of using fixed score 5 as other experiments mentioned previously. We test on the SQuAD dataset.

Table 3 shows the performance of each SCONER trained with generated scores or fix score 5, and the best baselines model on SQuAD dataset which is RM. We see that except for Q model, other three SCONERs are all better than the best baseline model, this suggests that the score for the positive candidates can be used in the training time. However, we also see that using score 5 is better than the generated scores in all cases. We further find that for the best SCONER model, Q+KA, 98% of the negative candidates have lower scores than the corresponding positive candidates, but the average generated score for the positive candidates is 3.64, which is less than 5. This suggests that a larger gap between scores of positive and negative candidates helps the model to differentiate the positive and negative candidates better. In addition, we also observe that the performance of the Q model, which does not use any augmentation in the generation, is much worse than the score 5 while the other three methods are similar. This suggests that the scores generated by augmentation are more reliable.

| Label | Model | | | | |
|---|---|---|---|---|---|
| | Q | Q+A | Q+KA | KQ+KA | RM |
| fix | 92.51 | 92.36 | 92.71 | 91.92 | 90.58 |
| generated | 73.42↓ | 91.57↓ | 91.78↓ | 90.92↓ | - |

*Table 3.* Each model is initialized with RoBERTa model. Three SCONER using generated scores beat best baseline. Using score 5 is better than generated scores. ↓ means decrease compared to fix score.

## 6. Limitations

In this paper, we study the retrieval-based question answering task and propose a new training strategy for a cross-attention re-ranker model. While we compare with three standard baselines and two simple retrievers, recently, there are many interesting neural retrievers have been proposed such as DPR (Karpukhin et al., 2020), ACNE (Xiong et al., 2020), SPARTA (Zhao et al., 2021), ColBERT-QA (Khattab et al., 2021), and Poly-DPR (Luo et al., 2022). Comparing SCONER with these latest neural retrievers will be an interesting future work.

## 7. Conclusion

Most previous training approaches for ReQA use the same labels for all negative candidates, we argue that different candidates should have different negativeness scores based on their semantic relevance to the question. Motivated by this, we ask the question - "can a neural re-ranker yield better performance trained on different negativeness scores?". To answer this question, we present SCONER, a new pipeline to train neural re-rankers by generating scores for negative candidates which is based on the semantic meaning between question-candidate pairs. Our experimental results show that SCONER outperforms all standard training methods across five datasets and demonstrate that using negativeness scores to train a neural re-ranker is better than using the same labels. Our proposed method makes negative candidates differentiable which further allows us to use more negative samples to train neural re-ranker.

## References

Ahmad, A., Constant, N., Yang, Y., and Cer, D. M. Reqa: An evaluation for end-to-end answer retrieval models.

*ArXiv*, abs/1907.04780:137–146, 2019.

Bilotti, M. W., Ogilvie, P., Callan, J., and Nyberg, E. Structured retrieval for question answering. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 351–358, Amsterdam, 2007. ACM.

Cakaloglu, T., Szegedy, C., and Xu, X. Text embeddings for retrieval from a large knowledge base. In *International Conference on Research Challenges in Information Science*, pp. 338–351, Limassol, 2020. Springer.

Chen, D., Fisch, A., Weston, J., and Bordes, A. Reading Wikipedia to answer open-domain questions. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1870–1879, Vancouver, Canada, July 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-1171. URL https://aclanthology.org/P17-1171.

Chen, T. and Van Durme, B. 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*, pp. 719–725, Valencia, Spain, April 2017. Association for Computational Linguistics. URL https://aclanthology.org/E17-2114.

Cohen, D., Yang, L., and Croft, W. Wikipassageqa: A benchmark collection for research on non-factoid answer passage retrieval. *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, abs/1805.03797, 2018.

Conneau, A. and Kiela, D. Senteval: An evaluation toolkit for universal sentence representations. *ArXiv*, abs/1803.05449:1–6, 2018.

Dai, Z., Xiong, C., Callan, J., and Liu, Z. Convolutional neural networks for soft-matching n-grams in adhoc search. *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, 10.1145/3159652.3159659:126–134, 2018.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL https://aclanthology.org/N19-1423.

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

Fisch, A., Talmor, A., Jia, R., Seo, M., Choi, E., and Chen, D. Mrqa 2019 shared task: Evaluating generalization in reading comprehension. *ArXiv*, abs/1910.09753, 2019.

Guo, J., Fan, Y., Ai, Q., and Croft, W. A deep relevance matching model for ad-hoc retrieval. *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, -:55–64, 2016.

Guo, M., Yang, Y., Cer, D. M., Shen, Q., and Constant, N. Multireqa: A cross-domain evaluation for retrieval question answering models. *ArXiv*, abs/2005.02507, 2020.

Guo, M., Yang, Y., Cer, D., Shen, Q., and Constant, N. MultiReQA: A cross-domain evaluation forRetrieval question answering models. In *Proceedings of the Second Workshop on Domain Adaptation for NLP*, pp. 94–104, Kyiv, Ukraine, April 2021. Association for Computational Linguistics. URL https://aclanthology.org/2021.adaptnlp-1.10.

Hui, K., Yates, A., Berberich, K., and de Melo, G. PACRR: A position-aware neural IR model for relevance matching. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pp. 1049–1058, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. doi: 10.18653/v1/D17-1110. URL https://aclanthology.org/D17-1110.

Joshi, M., Choi, E., Weld, D. S., and Zettlemoyer, L. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *The 55th annual meeting of the Association for Computational Linguistics (ACL)*, Vancouver, 2017. ACL.

Karpukhin, V., Oğuz, B., Min, S., Lewis, P., Wu, L. Y., Edunov, S., Chen, D., and tau Yih, W. Dense passage retrieval for open-domain question answering. *ArXiv*, abs/2010.08191, 2020.

Khattab, O., Potts, C., and Zaharia, M. Relevance-guided supervision for openqa with colbert. *Transactions of the Association for Computational Linguistics*, 9:929–944, 2021.

Khot, T., Clark, P., Guerquin, M., Jansen, P., and Sabharwal, A. Qasc: A dataset for question answering via sentence composition. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 8082–8090, New York, 2020. AAAI Press.

Kwiatkowski, T., Palomaki, J., Redfield, O., Collins, M., Parikh, A. P., Alberti, C., Epstein, D., Polosukhin, I., Devlin, J., Lee, K., Toutanova, K., Jones, L., Kelcey, M., Chang, M.-W., Dai, A. M., Uszkoreit, J., Le, Q., and Petrov, S. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466, 2019.

Laskar, M. T. R., Huang, J. X., and Hoque, E. Contextualized embeddings based transformer encoder for sentence similarity modeling in answer selection task. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pp. 5505–5514, Marseille, France, May 2020. European Language Resources Association. ISBN 979-10-95546-34-4. URL `https://aclanthology.org/2020.lrec-1.676`.

Liu, T.-Y. Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval*, 3(3):225–331, 2009.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. Roberta: A robustly optimized bert pretraining approach. *ArXiv*, abs/1907.11692, 2019.

Luo, M., Mitra, A., Gokhale, T., and Baral, C. Improving biomedical information retrieval with neural retrievers. *AAAI*, 2022.

Ma, J., Korotkov, I., Yang, Y., Hall, K., and McDonald, R. Zero-shot neural passage retrieval via domain-targeted synthetic question generation. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pp. 1075–1088, Online, 2021. ACL.

MacAvaney, S., Yates, A., Cohan, A., and Goharian, N. Cedr: Contextualized embeddings for document ranking. *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 36, 2019.

Makino, T. and Iwakura, T. A boosted supervised semantic indexing for reranking. In *AIRS*, Jeju Island, South Korea, 2017. Springer International Publishing.

McDonald, R., Brokos, G., and Androutsopoulos, I. Deep relevance ranking using enhanced document-query interactions. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 1849–1860, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1211. URL `https://aclanthology.org/D18-1211`.

Min, S., Chen, D., Zettlemoyer, L., and Hajishirzi, H. Knowledge guided text retrieval and reading for open domain question answering. *ArXiv*, abs/1911.03868, 2019.

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

Ozyurt, I. B., Bandrowski, A., and Grethe, J. S. Bio-answerfinder: a system to find answers to questions from biomedical texts. *Database*, 2020, 2020.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. Pytorch: An imperative style, high-performance deep learning library. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 32*, pp. 8024–8035. Curran Associates, Inc., Vancouver, 2019.

Rajpurkar, P., Zhang, J., Lopyrev, K., and Liang, P. Squad: 100, 000+ questions for machine comprehension of text. In *EMNLP*, Austin, 2016. Association for Computational Linguistics.

Rao, J., He, H., and Lin, J. Noise-contrastive estimation for answer selection with deep neural networks. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, pp. 1913–1916, New York, 2016. Association for Computing Machinery.

Rao, J., Liu, L., Tay, Y., Yang, W., Shi, P., and Lin, J. 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)*, pp. 5370–5381, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1540. URL `https://aclanthology.org/D19-1540`.

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

Rose, S., Engel, D., Cramer, N., and Cowley, W. Automatic keyword extraction from individual documents. *Text mining: applications and theory*, 1:1–20, 2010.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. Attention is all you need. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in*

*Neural Information Processing Systems*, volume 30, New York, 2017. Curran Associates, Inc. URL https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., von Platen, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Le Scao, T., Gugger, S., Drame, M., Lhoest, Q., and Rush, A. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 38–45, Online, October 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-demos.6. URL https://aclanthology.org/2020.emnlp-demos.6.

Xiong, C., Dai, Z., Callan, J., Liu, Z., and Power, R. 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*, -, 2017.

Xiong, L., Xiong, C., Li, Y., Tang, K.-F., Liu, J., Bennett, P. N., Ahmed, J., and Overwijk, A. Approximate nearest neighbor negative contrastive learning for dense text retrieval. In *International Conference on Learning Representations*, 2020.

Yang, Y., Yih, W.-t., and Meek, C. WikiQA: A challenge dataset for open-domain question answering. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 2013–2018, Lisbon, Portugal, September 2015. Association for Computational Linguistics. doi: 10.18653/v1/D15-1237. URL https://aclanthology.org/D15-1237.

Yang, Y., Cer, D., Ahmad, A., Guo, M., Law, J., Constant, N., Hernandez Abrego, G., Yuan, S., Tar, C., Sung, Y.-h., Strope, B., and Kurzweil, R. Multilingual universal sentence encoder for semantic retrieval. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pp. 87–94, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-demos.12. URL https://aclanthology.org/2020.acl-demos.12.

Yang, Z., Qi, P., Zhang, S., Bengio, Y., Cohen, W. W., Salakhutdinov, R., and Manning, C. D. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *EMNLP*, Brussels, Belgium, 2018. Association for Computational Linguistics.

Yilmaz, Z. A., Wang, S., Yang, W., Zhang, H., and Lin, J. Applying bert to document retrieval with birch. In *EMNLP/IJCNLP*, Hong Kong, 2019. Association for Computational Linguistics.

Zhang, Z., Vu, T., and Moschitti, A. Joint models for answer verification in question answering systems. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 3252–3262, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.252. URL https://aclanthology.org/2021.acl-long.252.

Zhao, T., Lu, X., and Lee, K. Sparta: Efficient open-domain question answering via sparse transformer matching retrieval. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 565–575, 2021.

## A. Related Work

### A.1. 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., 2021). 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 & 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.

### A.2. Neural Re-Ranker

Bag-of-words ranking models such as BM25 (Robertson & Zaragoza, 2009) have a long history in information retrieval. Although being efficient, these methods depend on handcrafted features and can not be optimized for a specific task such as ReQA. Therefore, a re-ranker is trained on a down-stream task to re-score the candidates after the first step retrieval. Neural networks have been applied as re-rankers (Guo et al., 2016; Hui et al., 2017; Xiong et al., 2017; Dai et al., 2018; McDonald et al., 2018), also called as answer selection models in some work (Rao et al., 2016; Yang et al., 2015; Rao et al., 2019; Laskar et al., 2020). Boosting technique has been proposed to train a neural re-ranker where the training samples are assigned with different weights(Makino & Iwakura, 2017). Recently, large language models like BERT (Devlin et al., 2019) and RoBERTa (Liu et al., 2019) are widely used as re-rankers (Nogueira & Cho, 2019; Yilmaz et al., 2019; MacAvaney et al., 2019). Such re-rankers take the concatenation of a query and a candidate as input and apply attention technique(Vaswani et al., 2017) to allow rich interaction between the question and the candidate. Then a classification or regression module (scoring layer) is added on top to compute a score. Binary classification entropy (BCE) is usually used to train a re-ranker, but BCE has limitations such as a large number of negative candidates being unused to create balanced training samples. Triplet loss addresses this issue by the idea of learning to rank (Liu, 2009). However, none of these methods addresses the concern whether we can use different negativeness scores to train a neural re-ranker. Similar to previous work, we use large PrLMs as re-rankers but different from theirs, we train a model using different scores for negative candidates. Re-ranking using ensemble models have been explored recently (Zhang et al., 2021), but since their systems are more complex than ours in terms of model size, we don't compare with them.

## B. Sentence Textual Similarity (STS)

**Review STS** STS determines how close two sentences are in terms of semantic meaning (Conneau & Kiela, 2018). Specifically, given two sentences, a high STS score indicates that they present similar meanings; while a low score implies that they have different meanings. The STS score is in the range of [0, 5].

Table 4 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.

| **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 4.* 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.

**Motivation of Using STS to Generate Scores** STS lays the foundation of our scoring approach because there is a relation between the STS task and the question-candidate ranking task. Considering a question and a candidate pair, if the candidate has similar information regarding the question, then it is likely to be a relevant candidate (corresponding to a high STS score); on the contrary, if it has less similar information, then it is likely to be irrelevant (corresponding to a lower STS score).

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

Meanwhile, STS is better than other methods of finding similar information because it considers the semantic meaning of two sentences such as synonyms of words.

## C. Figure of STS-Model and Neural Re-ranker

Figure 5 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.

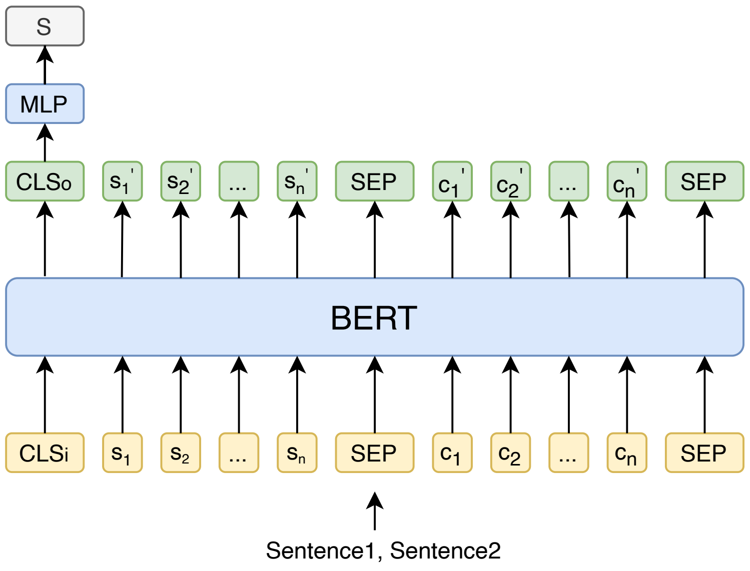

*Figure 5.* The structure of the STS-model, where $S_i$ are tokens from Sentence1, and $C_i$ are tokens from Sentence2.

## D. Data Augmentation

While STS and our question-answer sentence are similar task in some sense, 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 understanding. 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 5.

| 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 5.* Examples of augmented questions, 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.*"

| Dataset | Train | | | Test | | |
|---|---|---|---|---|---|---|
| | Ques. | Cand. | Avg. ans. per ques. | Ques. | Cand. | Avg. ans. per ques. |
| SearchQA | 24793 | 3163801 | 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 |

*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 |

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

## E. Data Processing and Statistics

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. Notice that the number of questions in the training time might be different than the MRQA dataset since we only use those samples where the answer sentences are not empty.

**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.

## F. Baselines

**Binary Classification Model (BCM)** We use the 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 outputs: 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 with label 1, and we randomly selected the same amount of negative samples from the top-100 candidates given by BM25 with label 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. We also use 1 as the label for positive samples but find that 5 yields better performance, thus we use label 5 to train all RM baselines. Appendix I.1 shows the results comparison between labels as 1 and 5.

**Triplet Model (TM)** This baseline has the identical model architecture as the RM baseline, but we use the triplet loss to train the model and in this way, 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 to train TM and SCONER, but SCONER use generated score as labels.

| 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.

## G. Experiment Setup

We use Huggingface (Wolf et al., 2020) and Pytorch (Paszke et al., 2019) implementation for training each model. For each dataset, we use up to 25 thousands questions to train a neural re-ranker. 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 re-ranker (including the baseline), we use four GTX1080 GPUs with MaxL 368, bs 16, lr 2e-5, 5 epochs, and gradient accumulation steps 2.

## H. 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.

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

**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.

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

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

**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.

## I. Ablation Study

### I.1. Ablation Study for 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.

### I.2. How Many Candidates are Needed for Re-ranking?

To answer this question, we use 50/100/150/200 candidates in the re-ranking time. We test on three datasets using the MRR metric and consider the best model of each dataset given in Table 1 (P@1 in Table 10), and they are KQ+KA, Q+KA, and Q+A models for NQ, SQuAD, and HQA, respectively.

From Table 9, we find that the performance gap of SQuAD and NQ are more noticeable than of HotpotQA. For SQuAD, re-ranking 200 candidates yields $\sim 1\%$ improvement compared to 50, and for NQ, re-ranking 100 candidates yields $\sim 1\%$ improvement compared to 50. But for HotpotQA, re-ranking 50 candidates surprisingly yields the best performance, $\sim 0.5\%$ better than 200. Further investigation reveals that the recall of BM25 on HotpotQA is already $99\%$ for 50 candidates and increasing the size of candidates rather introduces more distracting candidates in the re-ranking time; but for SQuAD and NQ, the recall of BM25 increases. On the other hand, re-ranking more candidates causes longer inference time, i.e. the inference time of re-ranking 50/100/150/200 candidates is 0.49/0.85/1.24/1.63 seconds. This suggests that if the recall of less candidates is already high enough (e.g. $99\%$), then using less candidates in re-ranking is time efficient and gets best performance.

| # | Model MRR | | | BM25 Recall | | |
|---|---|---|---|---|---|---|
| | SQuAD | NQ | HQA | SQuAD | NQ | HQA |
| 50 | 91.77 | 60.50 | **86.30** | 94.85 | 73.65 | 99.29 |
| 100 | 91.77 | **61.66** | 86.21 | 96.55 | 77.19 | 99.29 |
| 150 | 92.54 | 61.55 | 86.03 | 97.56 | 79.09 | 99.68 |
| 200 | **92.71** | 61.50 | 85.88 | **98.04** | **80.20** | **99.80** |

*Table 9.* # means the number of re-ranking candidates and HQA means HotpotQA dataset. When the recall of small size of candidate is high (e.g. $99\%$), using small size of candidates in re-ranking is better.

| # | Model MRR | | | BM25 Recall | | |
|---|---|---|---|---|---|---|
| | SQuAD | NQ | HQA | SQuAD | NQ | HQA |
| 50 | 88.61 | 53.39 | **79.83** | 94.85 | 73.65 | 99.29 |
| 100 | 88.61 | **53.75** | 79.83 | 96.55 | 77.19 | 99.29 |
| 150 | 89.29 | 53.75 | 79.83 | 97.56 | 79.09 | 99.68 |
| 200 | **89.48** | 52.80 | 79.80 | 98.04 | 80.20 | 99.80 |

*Table 10.* Compare the re-ranking size. # means the number of re-ranking candidates and HQA means HotpotQA dataset. When the recall of small size of candidate is high (e.g. $99\%$), using small size of candidates in re-ranking is better.