Double Retrieval and Ranking for Accurate Question Answering

Anonymous ACL submission

Abstract

Recent work has shown that an answer verification step introduced in Transformer-based answer selection models can significantly improve the state of the art in Question Answering. This step is performed by aggregating the embeddings of top \( k \) answer candidates to support the verification of a target answer. Although the approach is intuitive and sound, it still shows two limitations: (i) the supporting candidates are ranked only according to the relevancy with the question and not with the answer, and (ii) the support provided by the other answer candidates is suboptimal as these are retrieved independently of the target answer. In this paper, we address both drawbacks by proposing (i) a double reranking model, which, for each target answer, selects the best support; and (ii) a second neural retrieval stage designed to encode question and answer pair as the query, which finds more specific verification information. The results on three well-known datasets for Answer Sentence Selection (AS2) show consistent and significant improvement of the state of the art.

1 Introduction

In recent years, automated Question Answering (QA) research has received a renewed attention thanks to the diffusion of Virtual Assistants. For example, Google Home, Siri and Alexa provide general information inquiry services, while many other systems serve customer requests in different application domains. QA is enabled by two main tasks: (i) Answer Sentence Selection (AS2), which, given a question and a set of answer-sentence candidates, consists in selecting sentences (e.g., retrieved by a search engine) that correctly answer the question; and (ii) Machine Reading (MR), e.g., (Chen et al., 2017), which, given a question and a reference text, finds an exact text span that answers the question. Deploying MR systems in production is challenging for efficiency reasons, while AS2 models can efficiently target large text databases. Indeed, they originated from TREC QA tracks (Voorhees and Tice, 1999), which dealt with real-world retrieval systems since the first edition. Another limitation of MR is the focus on factoid answers: although it can in principle provide longer answers, the datasets developed for the task mainly contains short answers and in particular named entities. In contrast, as AS2 processes entire sentences, its inference steps always involve sentences/paragraphs, which make the approach agnostic to both factoid and non-factoid classes.

Garg et al. (2020) proposed the TANDA approach, which basically uses two stage of fine-tuning on pre-trained Transformer models (using a general dataset, ASNQ, and the target dataset), obtaining impressive improvement over the state of the art for AS2, measured on the two most used datasets, WikiQA (Yang et al., 2015) and TRECQA (Wang et al., 2007). The approach above, based on pointwise rerankers, was significantly improved by the Answer Support-based Reranker (ASR) (Zhang et al., 2021), which adds an answer verification step similar to the one operated by fact checking systems, e.g., see the FEVER challenge (Thorne et al., 2018).

More specifically, given a question \( q \), and a target, \( t \), answer to be verified, which is taken from a set of answer candidates \( C_k = \{c_1, ..., c_k\} \), ASR concatenates transformer-based embeddings of \((q, c_i)\) with the max-pooling vector produced by the top \( k \) embeddings of \((t, c_i)\), where the \( c_i \) are
selected by an initial answer reranking model (e.g., TANDA). For example, Table 1 reports a question, $q = \text{What causes heart disease?}$, with some candidate answers, $c_1$, $c_2$, and $c_3$. Selecting the correct answer $c_2$ is difficult, without the information: cardiovascular disease is also called heart disease. This information is provided by $c_1$. Thus, to compute the correctness probability of $c_1$, they exploit the representation of $c_2$, similarly to the way claims are supported in the fact verification.

ASR reduced the error of TANDA by 10% (relative), both on WikiQA and TREC-QA datasets. However, ASR shows two important limitations: first, when attempting the verification step of $t$, the $k$ candidates, used in the max-pooling operation, are ranked only based on the question, i.e., independently of $t$. Second, the support for each $t$ is provided by other answer candidates, which again are retrieved independently of the need of acquiring information for verifying $t$.

In this paper, we provide new answer verification models, which are more efficient and accurate than ASR. We introduce a new architecture, Double Answer Reranking (DAR), which uses two models for reranking target answers and supporting candidates, respectively. To verify $t$, the first, support reranker (SR), sorts $(q, t, c_i)$ triplets to find the best support $s_t$, while the second, answer reranker (AR), orders $(q, t, s_t)$ triplets, thus ranking all target answers $t$.

Additionally, we improve the quality of supports using a second retrieval stage that searches for passage relevant to $(q, t)$. This is important as standard answer candidates provide information only relevant to $q$, thus they not necessarily provide useful context for assessing $t$. As formulating an effective query for retrieving a question/answer pair is a new problem, and can be challenging, we exploit deep passage retrieval (DPR) (Karpukhin et al., 2020). This way, we automatically produce embedding for $(q, t)$ as the target query of a DPR model. As DAR is efficient, it can process many candidates from DPR, making Double Retrieval (DR) effective.

The results derived on three well-known AS2 datasets, WikiQA (Yang et al., 2015), TREC (Wang et al., 2007), and SelQA (Jureczek et al., 2016) show consistent and significant improvement over the state of the art. For example, DAR improves TANDA by 13.6% (relative error reduction), achieving the same accuracy of the computational expensive ASR verification approach (84.36%). Additionally, DAR-DR improves the absolute state of the art, reducing the error by an additional 8%.

We will release the datasets augmented with DPR retrieval (support candidates) for each $(q, a)$ of each of the datasets above.

2 Related work

We focus our research on QA systems based on Information Retrieval. Since early versions, e.g., TREC QA tracks (Voorhees and Tice, 1999), these systems have been based on a search engine, which retrieves documents relevant to the asked questions, followed by an inference step at paragraph level, to extract an answer. Efficient and accurate approaches use passage rerankers to select a piece of text that most likely contains the answer.

The reranking task can be modeled with a binary classifier $f(q, c_i)$ scoring the candidates, $C_k = \{c_1, \ldots, c_k\}$, retrieved for the questions $q$. If we train $f$ on correct/incorrect $(q, c_i)$, its score will also provide the probability $p(q, c_i)$ of a candidate to be correct, with which we can select the best answer, i.e., argmax$_{i=1}^{k} f(c_i)$. Passage reranking was revived by Wang et al. (2007), who defined the task of reranking answer sentences.

In recent work, $p(q, c_i)$ is estimated using neural networks, e.g., encoding question and answer text, separately with a CNN (Severyn and Moschitti, 2015). Also designing attention mechanisms, e.g., Compare-Aggregate (Yoon et al., 2019), inter-weighted alignment networks (Shen et al., 2017). The state of the art is achieved with pre-trained Transformers, e.g., (Garg et al., 2020).

A number of researchers has proposed more than one candidate for the inference stage, e.g., using pairwise model, i.e., binary classifiers of the form $\chi(q, c_i, c_j)$, which determine the partial rank between $c_i$ and $c_j$. For example, (Laskar et al., 2020; Tayyar Madabushi et al., 2018; Rao et al., 2016) use a pairwise loss and encoding. However, these methods have been largely outperformed by the pointwise models based on Transformers.

Bonadiman and Moschitti (2020) designed several joint models that improved early neural models for AS2 but, failed to improve Transformer-based models. Jin et al. (2020) used the relation between candidates in Multi-task learning approach for AS2 but as they did not exploit transformer models, their results are rather lower than the state of the art. Very recently, Zhang et al. (2021) proposed ASR, a model based on a pairwise reranker fed with the embeddings refined by a pairwise approach. This significantly improved the state of the
We use RoBERTa similarly to a multiple-choice where $y$ which models the dependencies between words and
where $t$ (typically represented by CLS), representing
loss:
$$L = -\sum_{i \in \{0,1\}} y_i \times \log(\hat{y}_i)$$ on pairs of text, $\hat{y}_i = p(q, c)$, and $y_0 = 1 - p(q, c)$.

3.2 Pairwise Classifier (PC)
We use RoBERTa similarly to a multiple-choice QA configuration (Zellers et al., 2018). We proceed as in the previous section obtaining the CLS representation for each $(q, c_i)$ pairs. Then, for each $t$, we concatenate the embedding of $(q, t)$ with all the embeddings $(q, c_i)$, where $c_i \neq t$. This way, $(q, t)$ is always in the first position. We train the model again using binary cross-entropy loss. At classification time, we select one $t$ candidate at a time, set it in the first position, followed by all the others, classify all $k$ candidates, and rerank them based on these scores.

3.3 All Candidate Multi-classifier (ACM)
We concatenate the question text with the one of the $k$ answer candidates, i.e., $(q_{SEP}c_1_{SEP}c_2_{SEP}c_k)$, and provide this input to the same Transformer model used for SBC. We use the final hidden vector $E$ corresponding to the first input token $[CLS]$ generated by the Transformer, and a classification layer with weights $W \in \mathbb{R}^{(k) \times |E|}$, and train the model using a standard cross-entropy classification loss: $y \times \log(\text{softmax}(EW^T))$, where $y$ is a one-hot vector representing labels for the $k$ candidates, i.e., $|y| = k$. We use a transformer model fine-tuned with the TANDA-RoBERTa-base or large models, i.e., RoBERTa models fine-tuned on ASNQ (Garg et al., 2020). The scores for the candidate answers are calculated as $p(\{c_1, \ldots, c_k\}) = \text{softmax}(EW^T)$. Then, we rerank $c_i$ according their probability.

3.4 Answer Support Reranker (ASR)
The previous models have been shown to be outperformed by ASR (Zhang et al., 2021).

We described its architecture in Figure 1. It consists of five main components: (i) the primary retrieval, which recuperates documents relevant to a question and produces answer sentence candidates, (ii) an SBC, which provides the embedding of the input $(q, t)$. This is built with the TANDA approach applied to RoBERTa pre-trained transformer (Garg et al., 2020). (iii) The joint representation of $(t, c_i)$, $i = 1, \ldots, k$, where $t$ and $c_i$ are the top-candidates reranked by SBC, and the final $(t, C_k \setminus \{t\})$ representation is obtained with a max-pooling operation over the $k$ pairs, $(t, c_i)$. (iv) The Answer Support Classifier (ASC) classifies each $(t, c_i)$ in four classes: (0) both answer correct, (1) $t$ is correct while $c_i$ is not, (2) vice versa, and (3) both incorrect. This multi-classifier is trained end-to-end with the rest of the network in a multi-task learning fashion, using its specific cross-entropy loss, computed with the labels above. (v) The Final Classification Layer takes in input the concatenation.
tion of the SBC embedding with the max-pooling embedding. Thus, the classifier scores $t$ with respect to $q$, also using the other candidates.

We note that ASC uses pre-trained RoBERTa-base (Liu et al., 2019), to generate $[CLS] \in \mathbb{R}^d$ embedding of $(q, t) = E_t$, $E_j$ is the $[CLS]$ output of another RoBERTa-base Transformer applied to answer pairs, i.e., $(t, c_j)$. Then, $E_j$ is concatenated to the max-pooling tensor from $\hat{E}_1, \ldots, \hat{E}_k$, that is,

$$V = [E_i : \text{Maxpool}([\hat{E}_1, \ldots, \hat{E}_k])],$$ (1)

where $V \in \mathbb{R}^{2d}$ is the final representation of the target answer $t$. Finally, a binary classification layer is built with a feedforward network:

$$p(y_i | q, t, C_k \setminus \{t\}) = \text{softmax}(W V + B),$$

where $W \in \mathbb{R}^{2d \times 2}$ and $B$ are parameters to transform the representation of the target answer $t$ from dimension $2d$ to dimension 2, which represents correct or incorrect labels.

### 4 Double Reranking and Retrieval

ASR is the state-of-the-art model for joint modeling candidates. However, it suffers from three main limitations: (i) it needs to limit $k$ otherwise the complexity may be too high, this means that it cannot consider all supporting candidates, (ii) the top $k$ candidates are the best answer ranked by TANDA, which does not guarantee that these are also the best supports, and (iii) answer candidates may support other target answers but they were not retrieved for this purpose. We address the above drawbacks proposing: (i) double reranking functions, which can efficiently rank support as well as the best target answer, and (ii) a second stage of retrieval that both considers target answer and question to retrieve specific support.

![Figure 1: Answer Support-based Reranker (ASR)](image1)

**Figure 1:** Answer Support-based Reranker (ASR)

![Figure 2: Double Answer Reranker and Retrieval (DAR-DR)](image2)

**Figure 2:** Double Answer Reranker and Retrieval (DAR-DR)

### 4.1 Double Answer Reranking (DAR)

The architecture, shown in Fig. 2, is much simpler than ASR: it just uses one RoBERTa transformer to encode triplets, question, target answer, candidate, i.e., $(q, t, c_j)$, rather than encoding $(q, t)$ and $(t, c_j)$ with two separate transformer models. Then two classification layers operate two different types of ranking of the same triplets: the first, Support Ranker (SR), given $t$, learns to rank the best support, $c_i$ higher. The second, Answer Reranker (AR), given the best support, i.e., $s_t = \text{arg-max}_{c \neq t \in C_k} \text{SR}(q, t, c_i)$, learns to rank the best answer producing, $f = \text{arg-max}_{t \in C_k} \text{AR}(q, t, s_t)$, as the final output.

**Training DAR** Training SR and AR is challenging as, for the former, labels are typically not available in standard dataset. Additionally, defining a support, i.e., a piece of knowledge improving the accuracy of another classifier is not a well-understood problem. Thus, we use feedback from AR directly, i.e., a high relevant support is the one that produces the highest score in AR, if the answer is correct, and the lowest score, otherwise. We train SR and AR, at the same time, in a multi-task learning fashion, also considering that the triplets SR and AR rank are essentially the same: learning their different roles boils down from selecting a subset of triplets for their training, along with the appropriate loss function.

SR has to learn to rank the best supports higher. This can be enforced by requiring that $s_t$ produces the highest score, $\text{AR}(q, t, s_t)$, among $c_i$ scores, $\{\text{AR}(q, t, c_i)\}_i$, if $t$ is correct, and the lowest score, otherwise. We enforce this property with a loss function: given a training example, $(q, C_k)$,
$C_k = \{c_1, \ldots, c_k\}$, where $c_i$ are associated with training label $t_i \in \{+1, -1\}$, we (i) select the best support, according to the current AR model, $s_t = \arg\max_{i \in t} E_t(q, t, c_i)$, and (ii) use the following ranking loss function to train SR:

$$L(q, c_1, \ldots, c_n) = -\log \frac{e^{\text{sim}(q, s_t)}}{\sum_{i=1}^{n} e^{\text{sim}(q, c_i)}}.$$  \hspace{1cm} (2)

This pushes the support that provides the highest confidence score for AR in the top of the rank.

We train AR as a binary classifier with the cross-entropy loss using all triplets, i.e., $(q, t, c_i)^\forall t, c_i \in C_k, t \neq c_i$.

### 4.2 Double Retrieval (DR) with DPR

The right side of Fig. 2 shows two retrieval steps: the first one is the traditional retrieval which, given an initial $q$, recuperates relevant documents, then split them in answer sentence candidates. However, if the objective is to retrieve supporting items to verify a target $t$, the appropriate query should be built with the pair $(q, t)$. For this reason, we propose a secondary retrieval step. We note that (i) DAR approach does not limit the number of initial support to a fixed $k$ as ASR does, either in training or in testing. This makes it suitable to work with more supporting items than those available from the primary search. (ii) Since the semantics of $(q, t)$ is difficult to capture, the usage of neural retrieval fed with the embedding of the pair above is a promising choice.

**Embeddings for support retrieval** We adapted the Dense Passage Retrieval (DPR) by Karpukhin et al. (2020) for our task of support retrieval. We built two encoders $E_Q(\cdot)$ for the pairs $(q, t)$, and $E_P(\cdot)$ for text passages $p$ (typically they are larger than a single sentence). The encoders map the input to a $d$-dimensional real-valued representation, while an indexing process computes representations for all text using $E_P(\cdot)$. The retrieval of relevant content for $(q, t)$ is done in two steps: (i) we compute the $(q, t)$ representation using $E_Q(\cdot)$. (ii) We then retrieve $M$ passages that have vector representations most similar to the pair representation, computed as the dot product of the vectors:

$$\text{sim}(q, p) = E_Q(q, t)^T E_P(p).$$ \hspace{1cm} (3)

The encoder is trained to make the dot-product similarity correspond to the expected ranking. Thus, for training our DPR, we use again the ranking loss in Eq. 2, where the label of $p$ is positive if it contains an answer sentence labelled as positive if the support is part of the paragraph, $s_t \in p$.

### 4.3 Double Ranking and Retrieval

The combination DAR-DR needs to consider the fact that AS2 datasets do not have annotated supports. For standard datasets, we consider candidates as potential supports, where the candidates are also annotated as correctness or incorrect answers. In contrast, when we retrieve new support using the $(q, t)$ query, no label is available. However, our DAR approach does not require support labels, thus we can still train our entire DAR-DR model, by simply considering two sets: standard $C_k$, on which we can train AR, and a set $S$ containing new supports retrieved by DPR. Again, SR can be trained on $C_k \cup S$, as the labels are automatically derived with $s_t = \arg\max_i SR(q, t, c_i)$, where $t \in C_k$ and $c_i \in C_k \cup S \setminus t$.

### 5 Experiments

We compare our models with several baselines we implemented from previous work, and ASR, which is the current state of the art for AS2. For the evaluation, we used three different datasets traditionally used for AS2. Finally, we provide error analysis and model discussion.

#### 5.1 Datasets

**WikiQA** is a QA dataset (Yang et al., 2015) containing a sample of questions and answer-sentence candidates from Bing query logs over Wikipedia. The answers are manually labeled. Some questions have no correct answers (all-), or only correct answers (all+). Table 2 reports the corpus statistics without all— questions, and without both all— and all+ questions (clean). We follow the most used setting: training with the noall— mode and then answer candidate sentences per question in testing with the clean mode.

**TREC-QA** is another popular QA benchmark by Wang et al. (2007). Since the original test set only contain 68 questions and previous method already achieved ceiling performance (Zhang et al., 2021), we combined train., dev. and test sets, removed questions without answers, and randomly re-split into new train., dev. and test sets, which contains 816, 204 and 340 questions, and 32,965, 9,591, and 13,417 question-answer pairs for the train., dev. and test sets, respectively.

**SelQA** is another benchmark for Selection-Based QA (Jurczyk et al., 2016), which composes about 8K factoid questions for the top-10 most prevalent topics among Wikipedia articles. We used the
Table 2: WikiQA dataset statistics

<table>
<thead>
<tr>
<th></th>
<th>Train</th>
<th>Dev</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>#Q</td>
<td>#A</td>
<td>#Q</td>
<td>#A</td>
</tr>
<tr>
<td>no all</td>
<td>8,73</td>
<td>8,672</td>
<td>126</td>
</tr>
<tr>
<td>clean</td>
<td>857</td>
<td>2,351</td>
<td>1,126</td>
</tr>
<tr>
<td></td>
<td>1,126</td>
<td>243</td>
<td>2,341</td>
</tr>
<tr>
<td></td>
<td>237</td>
<td>2,341</td>
<td>121</td>
</tr>
<tr>
<td></td>
<td>121</td>
<td>243</td>
<td>2,351</td>
</tr>
<tr>
<td></td>
<td>1,126</td>
<td>237</td>
<td>2,341</td>
</tr>
</tbody>
</table>

5.2 Training and testing details

**Metrics**  The performance of QA systems is typically measured with Accuracy in providing correct answers, i.e., the percentage of correct responses, which also refers to Precision-at-1 (P@1) in the context of reranking. We also use Mean Average Precision (MAP) and Mean Reciprocal Rank (MRR) evaluated on the test set, using the entire set of candidates for each question (this varies according to the dataset), to have a direct comparison with the state of the art.

**Models**  We use the pre-trained RoBERTa-Base (12 layer) and RoBERTa-Large-MNLI (24 layer) models, which were released as checkpoints for use in downstream tasks.

**Reranker training**  We adopt Adam optimizer (Kingma and Ba, 2014) with a learning rate of 2e-5 for the transfer step on the ASNQ dataset (Garg et al., 2020), and a learning rate of 1e-6 for the adapt step on the target dataset. We apply early stopping on the development set of the target corpus for both fine-tuning steps based on the highest MAP score. We set the max number of epochs equal to 3 and 9 for the adapt and transfer steps, respectively. We set the maximum sequence length for RoBERTa to 128 tokens.

**ASR training**  Again, we use the Adam optimizer with a learning rate of 2e-6 for training the ASR model on the target dataset. We utilize one Tesla V100 GPU with 32GB memory and a train batch size of eight. We use two transformer models for ASR: a RoBERTa Base/Large for PR, and one for the joint model (see Fig. 1). We set the maximum sequence length for RoBERTa to 128 tokens and the number of epochs as 20. We select the best k chosen in (Zhang et al., 2021).

**DAR implementation and training**  For training the DAR model, we also use the Adam optimizer but with a different learning rate, 5e-6. We utilize two Tesla A100 GPUs with 40GB memory and a train batch size of 128. DAR only needs one transformer model: a RoBERTa Base/Large (see Fig. 2). The maximum sequence length and the number of epochs are the same with ASR training, which are 128 and 20 separately.

**DPR implementation and training**  We utilize the same training configuration of the original DPR in Karpukhin et al. (2020). Then, we used it to build a large index having up to 130MM passages extracted from 54MM documents of CommonCrawl2. We selected English Web documents of 5,000 most popular domains, including Wikipedia, from the recent releases of Common Crawl of 2019 and 2020. We then filtered pages that are too short or without proper HTML structures, i.e., having title and content. To retrieve to N candidates, we input our DPR with (q, t) pairs as query to retrieve top 1000 passages.

**DAR-DR implementation and training**  The training configuration is similar to DAR training with the different steps highlighted in Sec. 4.2. For each (q, c_i) of our datasets, we used our DPR for retrieving 1000 supporting paragraphs, which are then split into sentences, s. We rank s according to a $E_Q(q, t) \cdot E_P(s)$, where $E_P(s)$ provides the embedding representation of each s, even though we trained $E_P(\cdot)$ for passages. We select the top 10 sentences as support for all the experiments with DAR-DR. It should be noted that all datasets for retrieval-based QA are based on candidates retrieved with an initial search engine, e.g., Bing, Google, TREC systems. This constitutes the first standard retrieval in our DR approach.

5.3 Comparative/ablated results

We design a set of baselines (see Sec. 3), which also constitute the best ablation systems of our most complex architecture DAR-DR. Indeed, SBC is our reimplementation of TANDA, which corresponds to the basic system (or basic component) of our architecture, it uses only one reranker and no joint inference. PC is the simplest joint model, which still uses only one classifier as SBC but applied to pairs of answers. ASR (ours) is our reimplementation of ASR, which uses an SBC model, a PC model, and an internal SR (called ASC) model as in DAR, used just for classification, no ranking. ASR-Rank extends ASR using the top 3 candidates re-ranked by ASC category 0 score (see Sec. 3.4), instead of

1https://github.com/pytorch/fairseq

2commoncrawl.org
using the standard TANDA rank. We introduced, ASR-Rank to show an approach similar to DAR. **ACM** is a joint model over all \( k \) candidates (theoretically more expressive than just joint models over pairs). **DAR** uses two rerankers as ASR-Rank but only one transformer and our approach to train them. Finally, **DAR-DR** adds to DAR new candidates retrieved by DPR.

**Main results** Table 3 reports P@1, MAP and MRR of models on WikiQA, TREC-QA and SelQA datasets. TANDA and ASR rows report the results obtained by Garg et al. (2020) and Zhang et al. (2021), respectively, which certify the alignment between our previous work setting and implementation. We note that:

(i) P@1, MAP and MRR correlate well, thus, we can focus our analysis on P@1, which typically provides the QA performance. In particular, the AS2 model accuracy numbers are in the lower 80% for all datasets. This means that absolute improvement are not expected to be large, thus we also report the relative error reduction (RER) for P@1, which better shows model differences.

(ii) Our SBC and ASR replicate the performance reported in previous work (WikiQA and TREC-QA), which are the previous state of the art.

(iii) We confirm that ASR, using candidate pairwise information greatly improves on single answer classification models, e.g., we observe a relative error reduction of 13.64% (from 81.89 to 84.36) over TANDA and SBC, which do not use the information from other candidates.

(iv) Our proposed model DAR significantly reduces the error of QA systems with respect to ASR by 4.58% (from 84.36 to 85.19), 2.55% (from 90.88 to 91.18), and 14.47% (from 93.14 to 94.15), on WikiQA, TREC-QA, and SelQA, respectively. It is interesting to note that DAR only uses the half of the parameters of ASR (125M vs. 250M). The combination between the two rerankers for answer and support generates more selective information than max-pooling pairwise embeddings.

(v) To verify that the unique feature of DAR of effectively combining training examples and their losses is a key element, we implemented ASR-Rank, which also selects supporting candidates for ASR, using its internal answer pair classifier, \( \text{ASC}(t, c_i) \). The results derived on WikiQA and TREC-QA show no difference between ASR and ASR-Rank, while the latter underperforms on SelQA. This shows that the improvement produced by DAR is not about selecting the best support in absolute, but it is about selecting the support that can produce the highest confidence in the answer selector (see Sec. 4.1).

(vi) DAR-DR introduces 10 additional supports to DAR processing, retrieved with our modified DPR approach. These new candidates do not have any label indicating if they are good or bad support. They are automatically ranked with the DAR approach. The results show an RAR of 2.27%, 4.93%, and 9.88%, on WikiQA, TREC-QA, and SelQA, respectively. Suggesting that retrieving supporting candidates for \((q, t)\) can be very effective.

(vii) Finally, we perform randomization test (Yeh, 2000) to verify if the models significantly differ in terms of prediction outcome. Specifically, for each model, we compute the best answer for each question and derive binary output based on the ground truth. We then follow the randomization test to measure the statistical significance between two models. We use 100,000 trials for each calculation. The results confirm statistically significant difference between DAR and DAR-DR against all the

<table>
<thead>
<tr>
<th>RoBERTa Base</th>
<th>WikiQA</th>
<th>TREC-QA</th>
<th>SelQA</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>P@1</td>
<td>RER</td>
<td>MAP</td>
</tr>
<tr>
<td></td>
<td>P@1</td>
<td>RER</td>
<td>MAP</td>
</tr>
<tr>
<td></td>
<td>P@1</td>
<td>RER</td>
<td>MAP</td>
</tr>
</tbody>
</table>

Table 3: Performance of different models using RoBERTa base Transformer on WikiQA, TREC-QA and SelQA. RER is the relative error reduction on P@1. † is used to indicate that the difference in P@1 between DAR and DAR-DR, against all the other marked systems, respectively, is statistically significant at 95%.
other models over all datasets, with \( p < 0.05 \), and between DAR and DAR-DR on SelQA.

### Results with large models

We experimented with SBC, ASR, DAR, and DAR-DR models implemented on a larger transformer, i.e., RoBERTa Large, on WikiQA. Table 4 reports the comparative results: SBC and ASR replicate the results by Zhang et al. (2021), i.e., a P@1 of 87.24\% and 89.71\%, respectively; the latter is the state of the art on WikiQA with a P@1 of 89.71\%. Both DAR and DAR-DR improve SBC up to 20\% RAR. However, even DAR-DR is behind ASR, by about 3.21\% of RER. This different outcome with respect previous results on the RoBERTa base can be explained by looking at the column reporting model parameters. As before, ASR uses the double of parameters of DAR, however, in this case the number of parameters is 710M, which is a large number in absolute: although DAR is a better model, it can hardly improve a model with 355M parameters more.

### 5.4 Model discussion and error analysis

Tab. 5 shows a question with the rank provided by SBC. The top-1 answer, \( c_1 \), is incorrect, as it refers to objects of Saturn’s rings, instead of targeting its moons. SBC probably got tricked by the phrase \textit{ranging in size}. ASR also selected \( c_1 \) using the support of the top 3 candidates selected by SBC, i.e., \( c_2, c_3, \) and \( c_4 \). These candidates support \( c_1 \) as they provide more context, e.g., \textit{moon}, which is not in \( c_1 \) but it is required in the question. The main problem of ASR is the fact that correct answers also tend to support imperfect but reasonable answers such as \( c_1 \). In contrast, for each \( t \), DAR learns to select the best support: in the example, it selects the correct answer \( c_2 \) using \( c_4 \) as support. This probably provides phrases such as \textit{seven moons that are large enough} supporting \( c_2 \) phrases such as \textit{have diameters larger than}.

In Tab. 6, we see an example, in which SBC ranks an incorrect answer at the top. It probably prefers \( c_1 \) to the correct answer \( c_2 \) because it matches the main question entity and verb, i.e.,

Table 5: A question with answer candidates, \( c_2 \) and \( c_3 \) are correct.

### 6 Conclusion

In this paper, we propose, DAR, a transformer architecture based on two reranking heads: (i) the answer reranker (AS2 model) and the answer support reranker. We optimize the latter imposing a loss function that penalizes non optimal support for the target answer, thus avoiding the need of defining and manually labeling supporting data. Additionally, we introduce a second retrieval stage based on DPR, where we optimize the score function between answer/question pair and the retrieving passage. The experiments with three well-known datasets for AS2, WikiQA, TREC-QA, and SelQA, show consistent improvement of DAR over the state of the art, and the potential benefit of the secondary retrieval, achieving up to 14.47 of relative error reduction (on SelQA). We will release software, models, and the DPR retrieved data for WikiQA, TREC-QA, and SelQA for fostering research in this field.

