DevRev's Expert Answers in a Flash: Improving Domain-Specific QA

Final Evaluation Report

Team 53

Inter IIT Tech Meet 11.0

1 Summary of the Mid Evaluation Report and introduction to the modifications on existing pipeline

In the mid-term report, we developed a highly modular, scalable and efficient pipeline for the task of general-purpose extractive question answering (QA). We employed an embedding-based search method - ANNOY [1] for finding a subset relevant context paragraphs from the given corpus matching with the input question. Then, the input question along with the context paragraph was feeded to a lightweight QA model - ELECTRA-small [2]. The best results obtained on a randomly sampled testing set comprising of 10k questions were - F1 Score = 76.1%, Para Score = 78.0%, QA Score = 78.0% and average inference time per question = 134.0ms.

With the rise in the limit of maximum inference time per question, we have further worked on improving the paragraph retrieval pipeline to perform fine-grained selection of context paragraph from the corpus. Also, we explored the effect of finetuning the trainable modules of our pipeline on the given training dataset. Lastly, we analyse the performance of our pipeline on different themes, and on synthetically generated QA dataset, and investigate how the memory of previously answered questions can be used to further optimize our pipeline.

2 Current Implementation

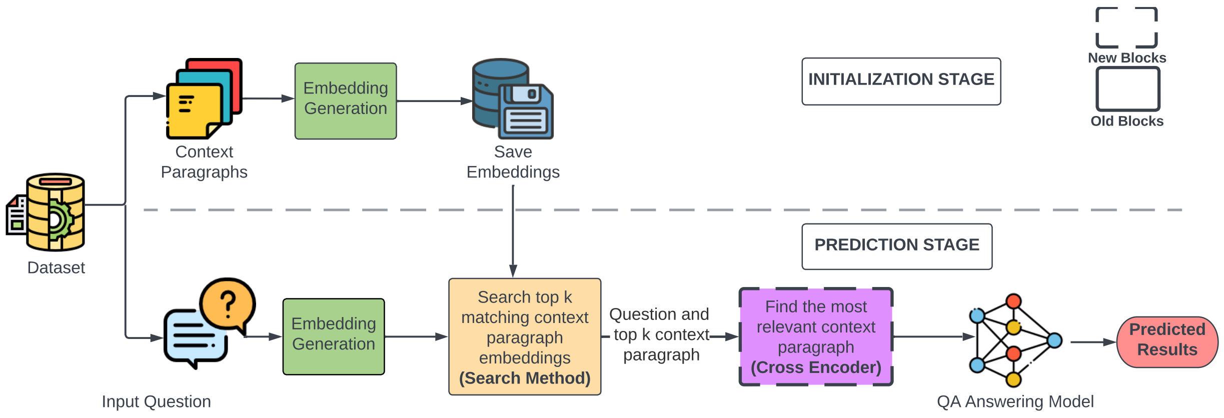

Figure 1: Overall flow diagram of the proposed solution.

The current implementation of our proposed pipeline is provided in Figure 1. We have introduced an additional **Cross-Encoder block** [3] to find the most relevant context paragraph from the subset of paragraphs found by the search method. The use of cross-encoder is necessitated by the fact that the search method only calculates the similarity that exists between the input question and the context paragraphs, whereas the cross-encoder provides fine-grained scoring to the context paragraphs on the basis of their relevance with respect to the input question. Further, we also improve our search method algorithm, and now use DeLADE [4] - a dense hybrid retrieval search method using lexical and semantic matching. DELADE is, in principle, superior to ANNOY, FAISS and other embedding based similarity searches, which is verified by our experiments.

For the purpose of experimentation, we divide the provided training set, comprising of 75k examples, into *train*, *val* and *test* sets by following a 80-10-10 split, ensuring **balanced representation of theme questions** (the relative ratio of questions per theme present in the base dataset was maintained). All evaluation metrics thus quoted are on the 10% *test* split, while the *train* and *val* split are used for finetuning the modules, wherever applicable. The following sections describe the different modules in our pipeline, along with exhaustive experiments to judge their performance. Also, for calculation of QA and Para scores, each theme is accorded the same weighting in the metric formulation.

3 Paragraph Retrieval

The aim of paragraph retrieval stage is to reduce the number of the context paragraphs being passed as input to the QA model. Passing multiple paragraphs to the QA models has several disadvantages. One, the inference time of the model for a particular question increases drastically, and two, the transformer-based models feature a limited input size, which prevents providing multiple context paragraphs as input in a single prediction pass.

Thus to build our paragraph retrieval pipeline, we use DeLADE as the preliminary search algorithm to reduce the search corpus of the context paragraphs for a given theme corresponding to the input question, followed by picking the most relevant context paragraph by ranking the filtered context paragraphs by a Cross-Encoder model. The following sections expound on the different modules used.

3.1 DeLADE

3.1.1 Introduction

DeLADE (Dense Lexical and Expansion) is a similarity searching algorithm that uses dense hybrid representations (DHR) to represent information. DHR is comprised of dense lexical representations (DLR) and dense semantic representations (DSR), which are jointly trained on the text corpus to generate accurate and faster retrieval. The speed benefit over existing embedding search methods (FAISS, ANNOY) is essentially provided by the use of dense representations, which have a lower dimension than the high dimensional embeddings used in existing approaches. Further, to score the paragraphs with respect to a given question during the training process, gated inner product is used. The gating term, which is a multiplicative constant to the inner product between the DHRs, represents the result of lexical matching between the corresponding DLRs.

3.1.2 Implementation Details

For building our solution, we test the variants of DeLADE pretrained on the MS MARCO dataset. Specifically, we use the **DeLADE-CLS-P** version (link) from the HuggingFace repository [5]. The DHRs in this version are trained by combining the lexical representations and [CLS] embeddings from a transformer model (BERT [6]), which leads to generation of robust representations. Two comparison algorithms are offered within the DeLADE ecosystem - Dot and Cosine. These correspond to the dot product and cosine similarity between the DHRs respectively. However, like the other search methods, use of DeLADE necessitates the inclusion of an initialization stage, where the DHRs of all the context paragraphs are stored for a given theme. This ensures that during the prediction stage, only the input question would be converted to its equivalent DHR, which results in overall efficiency of our pipeline during inference. Also, the presence of the initialization stage is supported by the fact that in a real-world setting, the context paragraphs/knowledge bases (KB) would remain static over time. During the periodic addition of new context paragraphs, their DHRs can be generated and stored for future use in the backend.

Table 1: Results on the test set using DeLADE-CLS-P with different comparison methods and ELECTRA-small as the QA model on *test* split.

Model	**Configuration**	**QA score**	**Para score**	**F1 score**	**Mean infer time (ms)**	**Median infer time (ms)**
DeLADE (base)	k=3, Dot	82.8%	84.9%	83.0%	673	874
	k=5, Dot	76.5%	78.1%	79.0%	955	1215
	k=3, Cosine	82.1%	84.3%	82.3%	552	488
	k=5, Cosine	77.9%	79.5%	78.8%	854	859

Table 1 shows the performance of our pipeline with DeLADE as the search method and using ELECTRA-small as the QA model. The higher time limits ($\sim$ 1 sec) allowed us to make DeLADE return more similar context paragraphs, whose number is denoted by 'k', for a given input question. The best metrics are given by our pipeline when DeLADE returns top 3 matching paragraphs using 'Dot' as the comparison method between the question and context paragraph DHRs. When compared with the previous best (as in the mid-evaluation report), using DeLADE leads to a relative increase in both Para and QA scores by around 6% and 9% respectively. The drop in performance of the pipeline with increasing 'k' value can be attributed to the sub-optimal performance of the QA model on being exposed to large context paragraph search space.

3.1.3 Finetuning

To further improve the performance of DeLADE, we tried finetuning it on the *train* split of the provided training dataset. The finetuning settings were kept as default, i.e., AdamW as the optimizer with a learning rate of 7e-6 and batch size of 8. Each epoch comprised of 5000 optimization steps, and the average time per epoch was determined to be around 85 minutes. To better gauge the efficacy of the finetuning process, we formulate a new metric - **Para Accuracy** given *n* sample points, which is mathematically defined as:

$$\text{Para Accuracy} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{1} \text{ if any of returned para match with target para, } \mathbf{0} \text{ otherwise} \times 100\% \tag{1}$$

Table 2: Finetuning results on the *val* set using DeLADE-CLS-P

Optimisation Steps	Para Accuracy	Optimization Steps	Para Accuracy
Base	99.68%	20k	93.03%
5k	95.92%	25k	90.81%
10k	96.03%	30k	90.47%
15k	93.67%	35k	91.34%

The Para Accuracy was determined on the *val* set after each epoch. It can be seen from Table 2 that the the Para Accuracy shows a decreasing trend, implying that the base variant of DeLADE is itself properly optimized, and that finetuning acts as a detriment to its performance. Hence, for our solution, we use the base DeLADE-CLS-P variant without any finetuning, with dot product for similarity measurement.

3.2 Cross Encoder

3.2.1 Introduction

Cross-encoders are transformer-based models which take in two sentences as input, and output a score between 0 and 1 depending on the semantic similarity of the two. The distinctive feature of cross-encoders is the use of siamese and triplet networks [7] to update the model weights during training/finetuning so that meaningful embeddings are generated for scoring, which is done using cosine similarity. The use of cross-encoder ensures that the top 'k' relevant context paragraphs returned by DeLADE can be ranked on the relevance with respect to the input question on a relative scale (in contrast to the absolute scoring done by DeLADE). The context paragraph having the highest relevance score is passed to the QA model, which ensures reduced inference time and improved accuracy of the entire pipeline.

3.2.2 Implementation Details

We use the pretrained cross-encoder models available on HuggingFace repository (link). Targeting version 2 models, we evaluate three variants built on the MiniLM architecture [8], which are pretrained for sentence similarity scoring task on MS MARCO dataset. Each of the variants are specified by a LX nomenclature, which denotes the number of transformer blocks present in the MiniLM model.

Table 3 shows the comparative results obtained by using the cross-encoder model alongwith pretrained DeLADE-CLS-P as the search module using Dot similiarity and ELECTRA-small as the QA model. It can be seen that incorporating the cross-encoder block leads to a drastic increase in the performance of our solution pipeline. On the **ms-marco-MiniLM-L4-v2** checkpoint that performs comparatively better than the other variants, the relative gains obtained over the pipeline omitting the cross-encoder range from 8.5% for the QA score to 9.5% for the Para score, with the average inference time rising only around 50ms. The median inference time per question is found to be 947 ms, which is well under the limit of 1 sec required by the problem statement. Also note the dependence of the subset of relevant context paragraphs (denoted by 'k') being passed to the cross-encoder model. The most optimal results are obtained when the value of 'k' is chosen as 3, which highlights that the cross-encoder performs sub-optimally when it encounters a larger search space.

Table 3: Comparison of different pretrained checkpoints of cross-encoder using DeLADE-CLS-P as the search method and ELECTRA-small as the QA model on *test* split.

Top hits 'k' (from DeLADE)	Cross-encoder checkpoint	QA score	Para score	F1 score	Mean infer time (ms)	Median infer time (ms)
3	ms-marco-MiniLM-L2-v2	88.4%	91.5%	88.8%	612	738
5	ms-marco-MiniLM-L2-v2	88.3%	91.3%	88.7%	686	1006
3	**ms-marco-MiniLM-L4-v2**	**89.8%**	**93.0%**	**90.3%**	**716**	**947**
5	ms-marco-MiniLM-L4-v2	86.1%	89.2%	90.3%	959	1596
3	ms-marco-MiniLM-L6-v2	71.3%	73.9%	90.5%	1312	2019
5	ms-marco-MiniLM-L6-v2	82.7%	85.7%	90.6%	1067	1737

3.2.3 Finetuning

As with the case for DeLADE, we try to improve the performance of the cross-encoder by finetuning it on the *train* split of the provided dataset. AdamW was used as the optimizer with a learning rate of 2e-5, and batch size was kept as 16. The average time per epoch came out to be around 88 minutes. Again, the Para Accuracy (Equation 1) metric was used to gain insights about the improvements (or lack thereof) in the predictive power of the cross-encoder, whose formulation remains the same as in Equation 1.

Table 4: Finetuning results on the *val* set using ms-marco-MiniLM-L4-v2 cross-encoder model

Epoch	Para Accuracy
base	90.3%
4	93.0%
7	92.4%
10	94.4%
14	94.7%
17	**95.1%**
20	92.7%

Table 4 shows the improving trend of the Para Accuracy score till 17 epochs, post which the score starts to decrease. Thus, it can be inferred that the cross-encoder achieves its optimal performance at 17 epochs, post which the effects of over-fitting on the finetuning data leads to a decrease in the Para Accuracy scores over the *val* split. Hence, we conclude to use the finetuned version of cross-encoder (trained till 17 epochs) in our solution pipeline.

4 Question Answering

After retrieving the most relevant paragraph from the given corpus using DeLADE and cross-encoder for a given input question, the next task is to perform extractive question answering on the context paragraph to find the exact answer to the question if the question is answerable. In case the question is not answerable, the QA model returns an appropriate prediction to indicate the same. Kindly note that we still use **ELECTRA-small** model pretrained on SQuAD v2 dataset as the QA model carried over from the mid-evaluation stage, due to obvious benefits of accuracy and efficiency offered by the same over the other tested models.

4.1 Introduction

ELECTRA is a transformer-based model which is pretrained using an efficient replaced token detection method, which allows for training the model in a discriminative setting (i.e., predicting whether the corrupted token was replaced or not). This leads to ELECTRA learning more robust internal text representations than the transformer models built on BERT as the base, while using only a fraction of compute power. The model is also quite scalable, and features reduced inference times per sample in comparison with other transformer models when deployed on same compute environments.

4.2 Implementation Details

For our pipeline, we use the HuggingFace variant of **ELECTRA-small** (link), which is pretrained on uncased English text, and further finetuned for extractive QA on SQuAD v2. ELECTRA-small comprises 12 transformer layers, and has 14M trainable parameters. In contrast, the next better variant of ELECTRA, ELECTRA-base, has 110M trainable parameters while featuring the same number of transformer layers. The rise in the number of trainable parameters greatly increases the inference time of our model (around x7 times), which restricts QA model to ELECTRA-small only. Also, as observed in our mid-evaluation report, ELECTRA in general does not respond well to the model optimizations such as pruning and quantization, with only an insubstantial reduction in inference times. The F1 scores of ELECTRA-small on the *test* set come out to be 93.1%, vindicating its expected near-SOTA performance.

4.3 Finetuning

An attempt to further improve the QA model performance is made by finetuning it on the *train* split of the provided dataset. Finetuning is carried out on default settings, which use AdamW as the optimizer at a learning rate of 2e-5 and a batch size of 32. Each epoch takes approximately 10 minutes for finetuning.

Table 5: Finetuning results on the *val* set for ELECTRA-small

Epoch	F1 sore	Epoch	F1 score
base	**93.0%**	5	91.0%
1	92.0%	6	91.0%
2	92.0%	7	91.0%
3	92.0%	8	91.0%
4	92.0%	9	92.0%

It can be seen from Table 5 that the F1 scores mostly remain constant, with minute decrease in scores. This leads to the conclusion that the available checkpoint of ELECTRA-small is trained to the optimal point, and any further finetuning leads to overfitting on the *train* split (To note is the fact that the training dataset provided to us is also a subset of SQuADv2, and hence the expected behavior). Thus we use ELECTRA-small in our solution without any finetuning.

4.4 Analysis on synthetic data

To further expand the training corpus, we generate synthetic data points across all the 361 themes using the provided training data. The steps to generate the same is as follows:

- **Generating context paragraphs**: We use the method of back-translation to generate synthetic context paragraphs from the context paragraphs present in the provided dataset. Back-translation is the process where the text sequence is translated to a foreign language, and then re-translated to its source language. Such a process is known to retain the semantic meaning of the input sentence, while changing its lexical structure [9]. We use the Marian-MT pretrained models for translation, with the intermediate foreign language being French. Since French has a greater lexical similarity with English (around 27%), the semantic meaning of the paragraphs is maintained.
- **Generating questions**: To generate questions corresponding to the generated context paragraphs from the previous step, we use the pretrained T5 transformer model [10] finetuned on the task of conditional text modelling using *squad_modified_for_t5_qg* dataset. The model checkpoint is used from HuggingFace repository (link). Also, to account for unanswerable questions, we allocate a random context paragraph to a some generated questions, which is different from the one used to generate the same.
- **Generating Answers**: With synthetic questions and context paragraphs generated, the final step is to find accurate extractive answers to the questions. For this purpose, we use the highly accurate pretrained ELECTRA-large model finetuned on the SQuAD v2 dataset (link).

The synthetic dataset comprises of 58,634 questions spread across 361 themes and 13,146 context paragraphs. We perform both evaluation and finetuning of our QA model on the same. As with the provided datatset, we divide the synthetic dataset into *train*, *val* and *test* split following a 80-10-10 split respectively.

Table 6: Finetuning results on the *val* set of the synthetic data for ELECTRA-small

Epoch	F1 score	Epoch	F1 score
base	78.2%	5	88.5%
1	88.2%	**6**	**89.0%**
2	88.5%	7	88.8%
3	88.7%	8	88.6%
4	88.7%	9	88.7%

Table A

QA Model	Finetuned on	Test Dataset	F1 score
Base	None	Test Split of Synthetic Corpus	78.3%
Finetuned	Train Split of Synthetic Corpus	Test Split of Synthetic Corpus	88.6%
Base	None	20% split of 75k data	93.1%
Finetuned	Train Split of Synthetic Corpus	20% split of 75k data	74.8%

Table B

Table 6A shows the results of finetuning the ELECTRA-small model on the *train* split of the synthetic dataset, with F1 scores quoted on the corresponding *val* split. It can be seen that the best results are obtained at the 6th epoch of finetuning, with an F1 score of 89.0%.

Table 6B shows the performance of the base ELECTRA-small on the *test* split of the synthetic dataset, as well as the performance of the finetuned ELECTRA-small on the *test* split of synthetic as well as provided training dataset (denoted by 75k - the number of questions in it). While the finetuned QA model gave higher results than the base model on the *test* split of the synthetic dataset, the results of the finetuned model are worse than the base model on the *test* split of the training dataset. One possible reason for this anomaly could be the the bias of the question-generation and paragraph generation modules, which leads to generation of samples lexically different from those given in the training set, which may lead to different representations in the embedding space.

5 Pipeline analysis

From the exhaustive experiments performed in Section 3 and Section 4, we use **DeLADE-CLS-P** as the search method, **ms-marco-MiniLM-L4-v2** as the cross-encoder model finetuned on the *train* split of the provided dataset till 17th epoch, and base **ELECTRA-small** as the extractive QA model. The results thus obtained on the *test* split are as follows: F1 = **92.8%**, Para Score = **95.1%**, QA Score = **91.5%** and average inference time per question = **767.0ms**.

5.1 Theme-wise analysis

As an investigative study, we find the theme-wise F1 scores on the *test* split using our pipeline. For the questions spread across 361 themes in the *test* split, we find that the questions on around 18 themes return a perfect F1 score of 1, while the F1 scores fall under 79.0% for 16 themes.

Table 7: F1 scores on top 10 themes (Table A) and bottom 10 themes (Table B).

Theme	F1 score	Theme	F1 score
Emotion	100.0%	Pope_John_XXIII	100.0%
Buckingham_Palace	100.0%	Southern_Europe	100.0%
Republic_of_the_Congo	100.0%	Virgil	100.0%
Printed_circuit_board	100.0%	Baptists	100.0%
Labour_Party_(UK)	100.0%	House_music	100.0%

Table A: Top 10

Theme	F1 score	Theme	F1 score
Myanmar	66.0%	Sexual_orientation	72.0%
Hydrogen	68.0%	Florida	73.0%
Botany	68.0%	Bacteria	74.0%
Seven_Years%27_War	69.0%	Bill_%26_Melinda_Gates_Foundation	74.0%
Greeks	71.0%	Elevator	75.0%

Table B: Bottom 10

Table 7 gives the top 10 and bottom 10 themes, arranged first by the average F1 scores for the questions contained within them, and then by the number of questions in each theme. The theme-wise deviations are quite pronounced, with the worst F1 score being reported for the theme - 'Myanmar' - of about 66.0%. To improve upon the scores for the questions on the 16 worst performing themes having F1 scores under 70.0%, we provide an additional finetuning to the QA model on a data corpus comprising questions from those themes itself. This corpus is created from the training dataset provided, as well as from the synthetic corpus generated by us (Section 4.4).

Table 8: F1 scores on top 10 themes (Table A) and bottom 10 themes (Table B) post theme-wise finetuning.

Theme	F1 score	Theme	F1 score
Emotion	96.0%	Pope_John_XXIII	94.0%
Buckingham_Palace	100.0%	Southern_Europe	90.0%
Republic_of_the_Congo	100.0%	Virgil	100.0%
Printed_circuit_board	100.0%	Baptists	97.0%
Labour_Party_(UK)	100.0%	House_music	100.0%

Table A: Top 10

Theme	F1 score	Theme	F1 score
Myanmar	63.0%	Sexual_orientation	72.0%
Hydrogen	68.0%	Florida	78.0%
Botany	71.0%	Bacteria	80.0%
Seven_Years%27_War	71.0%	Bill_%26_Melinda_Gates_Foundation	73.0%
Greeks	69.0%	Elevator	77.0%

Table B: Bottom 10

We try finetuning our QA model on a custom generated corpus comprising of question-context paragraph pairs from the themes which reported F1 scores less than 70.0%, using the provided training dataset. While there is a minor improvement in the F1 scores for the worst performing themes (an average increase of 1.3% over all the 16 themes), the overall decrease in the F1 scores for the top performing themes and some of the worse performing themes, as well as on *test* split of the provided dataset militates the benefits of such a finetuning process. Table 8 shows the F1 scores for the top and bottom 10 themes post the finetuning process for reference.

5.2 Caching of predictions for previously answered questions

We also explore caching of previous context paragraph predictions during the inference process to improve the metrics of our pipeline for questions similar to the ones that have already been answered. To implement such a scheme, we store the DHRs of the questions obtained from DeLADE. For a new question, its DHR is compared with the stored DHRs via cosine similarity. If the similarity score lies above a certain threshold, then the context paragraph predicted for the corresponding DHR embedded question is directly assigned to the input question. This ensures that the similarity matching stage of DeLADE and the cross-encoder module is completely skipped for such questions, and the input question and the determined context paragraph are directly passed to the QA model.

Table 9: Comparative results with different caching thresholds.

Caching Threshold	QA score	Para score	F1 score	Mean infer time (ms)	Median infer time (ms)	% Question Cached
0.75	82.3%	85.4%	91.7%	1100	1414	15.8%
0.80	90.1%	93.5%	92.2%	845	1073	11.5%
0.85	77.5%	80.5%	92.3%	1214	1636	8.1%
w/o Caching	**91.5%**	**95.1%**	**92.8%**	**767**	**1272**	**N/A**

Through experimental analysis, we determine that optimal results are obtained when the threshold is set to 0.80, as can be seen from Table 9. However, it is to be noticed that the caching process only affects a small percentage of the questions in the *test* split of the provided dataset, and the extra temporal overhead of matching the DHRs yields no benefits in inference time. Also the results obtained without caching are much better than those obtained without applying the caching process. Thus it can be concluded that the use of caching scheme yields no obvious benefits in the performance of our pipeline, at least on our *test* split.

Another useful caching approach could be when an input question possesses a high similarity with a previously answered question, in the event of which the QA stage can also be skipped by assigning the answer of the latter to the former. However, we did not implement the same as the percentage of questions impacted by the same in the *test* dataset is quite small.

Nonetheless, we believe that in a real-time environment where QA systems deal with particularly large data, the caching scheme proposed by us can lead to substantive gains over the baseline. This can be attributed to two reasons. One, for large data, sophisticated search algorithms such as those based on hashing can be used to provide constant-time lookup for the DHRs of previously answered questions matching with the DHR of the input question. Currently, we implement this search iteratively, which leads to large temporal overhead. Use of hashing-based search is inefficient for small datasets such as the *test* split. Second, a custom embedding model may be trained to integrate better with our caching method, which is optimized on assigning closely-spaced embeddings that correspond to the same context paragraph, rather than being optimized on the semantic similarity of the questions.

5.3 Final results

After comprehensive analysis built over the different modules of our pipeline, we determine the following configuration which was found to return the most optimal results over the *test* (10%) split of the provided training dataset.

Search method $\longrightarrow$ DeLADE-CLS-P, with Dot similarity metric, with k = 3.
Cross-encoder $\longrightarrow$ ms-marco-MiniLM-L-4-v2 finetuned on the *train* split of the provided dataset, till 17 epochs.
QA Model $\longrightarrow$ ELECTRA-small
Caching Used $\longrightarrow$ No

The scores obtained on the *test* split are as follows: F1 = **92.8%**, Para Score = **95.1%**, QA Score = **91.5%** and average inference time per question = **767.0ms**.

5.4 Results on provided evaluation set (5th February)

The provided evaluation set contained around 1,179 context paragraphs spread across 30 themes. These themes were not contained in the training set provided earlier. Each theme had a median of around 32 paragraphs per theme. The questions were provided for only 14 out of 30 themes, to a total of 944. The median number of questions per theme averaged around 67.

The low number of questions in the provided set motivated us to inflate the same using our synthetic data generation scheme (Section 4.4), so that we can evaluate the performance of our pipeline on some independent test set. The synthetic dataset contained around 463 paragraphs and 1,372 questions across the 14 themes (for which questions were provided in the original evaluation set). We also generate unanswerable questions, which were absent in the evaluation dataset. We finetune the QA model on the complete evaluation set, and compare the F1 results with the untuned base QA model.

Table 10: Evaluation results of the different pipeline configurations on the synthetic dataset

QA Model Finetuned on	**Evaluation Data**	**Batch size**	**F1 score**
NA (Base)	Synthetic data	8	80.7%
Evaluation data			72.4%

It can be seen from Table 10 that the best results are still returned by the base QA model. This corroborates our earlier inference that the QA model is itself trained to the optimal point, and responds poorly to any finetuning. We use the DeLADE and finetuned cross-encoder models as identified on the *test* split of the provided training dataset.

Thus, to generate the submission file on the unseen questions provided in the second stage of the evaluation day, we use the **DeLADE-CLS-P** as the search module, finetuned **ms-marco-MiniLM-L4-v2** as the cross-encoder model, and **ELECTRA-small** as the QA model (as stated in Section 5.3).

Also kindly note that as discussed in Section 4.4, the presence of lexical bias in the synthetic dataset generation process leads to low evaluation F1, QA scores and Para scores. However, as our experiments have substantiated, the pipeline performs quite well on the provided corpus, and that the relative performance of the pipeline on the synthetic dataset usually translates to the provided dataset, and that the absolute performance is a misleading indicator of the performance of our pipeline on the evaluation dataset.

5.5 Concluding Remarks

Our solution pipeline built for the task of QA is highly scalable and efficient, and addresses the constraints provided by the problem statement. The USP of our solution is that it is highly modular, and the various process blocks in our pipeline - search method, cross-encoder and QA, can accommodate different models and architectures as per the end requirements. We provide the link to the different finetuning codes, which can be accessed from (link)

References

[1] Spotify. ANNOY library. `https://github.com/spotify/annoy`.

[2] Kevin Clark, Minh-Thang Luong, Quoc V Le, and Christopher D Manning. Electra: Pre-training text encoders as discriminators rather than generators. arXiv preprint arXiv:2003.10555, 2020.

[3] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. arXiv preprint arXiv:1908.10084, 2019.

[4] Sheng-Chieh Lin and Jimmy Lin. A dense representation framework for lexical and semantic matching. arXiv preprint arXiv:2206.09912, 2022.

[5] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. Huggingface's transformers: State-of-the-art natural language processing. arXiv preprint arXiv:1910.03771, 2019.

[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805, 2018.

[7] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 815–823, 2015.

[8] Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. Advances in Neural Information Processing Systems, 33:5776–5788, 2020.

[9] Anup Kumar Gupta, Vardhan Paliwal, Aryan Rastogi, and Puneet Gupta. Trieste: translation based defense for text classifiers. Journal of Ambient Intelligence and Humanized Computing, pages 1–12, 2022.

[10] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. The Journal of Machine Learning Research, 21(1):5485–5551, 2020.