Few-Shot Question Answering by Pretraining Span Selection

Ori Ram[*,1] **Yuval Kirstain**[*,1] **Jonathan Berant**[1,2] **Amir Globerson**[1] **Omer Levy**[1]

[1] Tel Aviv University
[2] Allen Institute for AI
{ori.ram,yuval.kirstain,joberant,gamir,levyomer}@cs.tau.ac.il

Abstract

In a number of question answering (QA) benchmarks, pretrained models have reached human parity through fine-tuning on an order of 100,000 annotated questions and answers. We explore the more realistic few-shot setting, where only a few hundred training examples are available. We show that standard span selection models perform poorly, highlighting the fact that current pretraining objective are far removed from question answering. To address this, we propose a new pretraining scheme that is more suitable for extractive question answering. Given a passage with multiple sets of recurring spans, we mask in each set all recurring spans but one, and ask the model to select the correct span in the passage for each masked span. Masked spans are replaced with a special token, viewed as a question representation, that is later used during fine-tuning to select the answer span. The resulting model obtains surprisingly good results on multiple benchmarks, e.g., 72.7 F1 with only 128 examples on SQuAD, while maintaining competitive (and sometimes better) performance in the high-resource setting. Our findings indicate that careful design of pretraining schemes and model architecture can have a dramatic effect on performance in the few-shot settings.

1 Introduction

The standard approach to question answering is to pretrain a masked language model on massive amounts of unlabeled text, and then fine-tune it with a thin span selection layer on top to select the answer (Devlin et al., 2019; Joshi et al., 2020; Liu et al., 2019). While this approach is effective, and sometimes exceeds human performance, its success is based on the assumption that large quantities of annotated question-answer examples are available.

[*] Equal contribution.

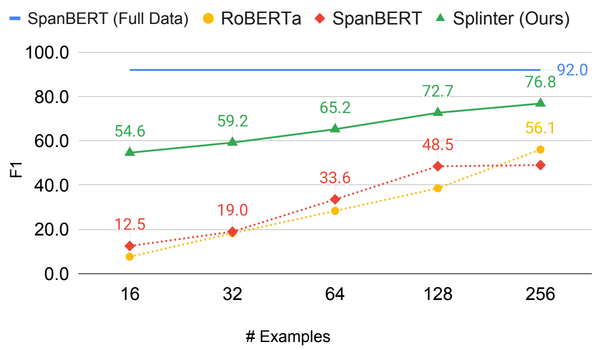

Figure 1: Performance of SpanBERT (red dotted line, diamond points) and RoBERTa (yellow dotted line, circular points) base-size models on SQuAD, given different number of examples (see Section 4 for exact details of the setting). SpanBERT (Base) performance on SQuAD's full data (blue line) is given for reference. Our model (Splinter, green line, triangular points) reduces this gap significantly.

For instance, both SQuAD (Rajpurkar et al., 2016, 2018) and Natural Questions (Kwiatkowski et al., 2019) contain an order of 100,000 question and answer pairs in their training data. This assumption quickly becomes unrealistic as we venture outside the lab conditions of reading comprehension over English Wikipedia, and attempt to crowdsource question-answer pairs in other languages or domains of expertise (Tsatsaronis et al., 2015; Lai et al., 2017; Kembhavi et al., 2017). How do question answering models fare in the more practical case, where an in-house annotation effort can only produce a couple hundred training examples?

This work investigates the task of few-shot question answering by sampling small training sets from existing question answering benchmarks. We observe that despite the use of pretrained models, the standard approach yields poor results when fine-tuning on few examples (Figure 1). For example, RoBERTa-base (Liu et al., 2019) fine-tuned on 128

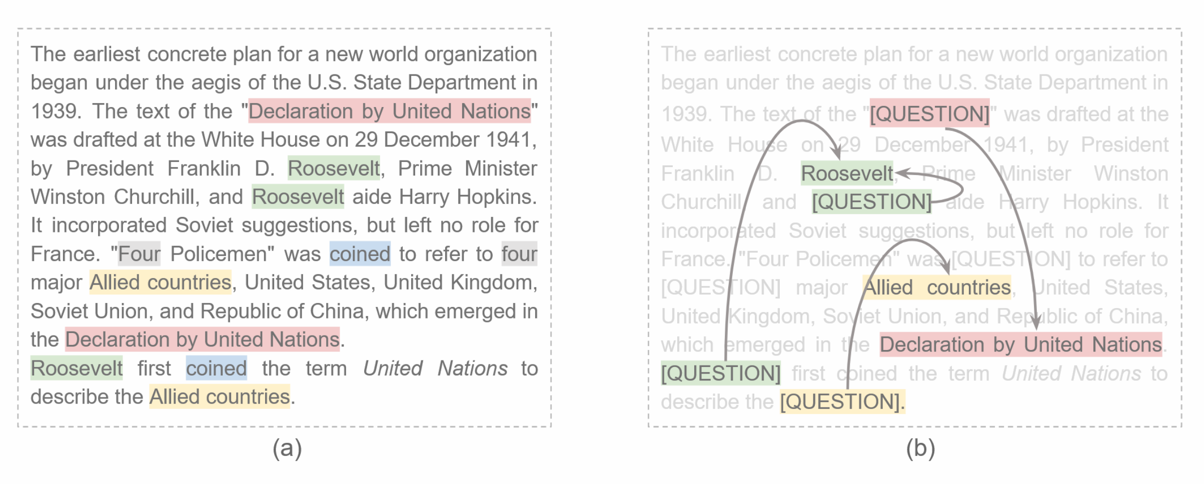

Figure 2: An example paragraph before (a) and after (b) masking recurring spans. Each color represents a different cluster of spans. After masking recurring spans, only one span from each cluster remains unmasked, the answer. The pretraining task is to predict the correct answer for each `[QUESTION]` token.

question-answer pairs from SQuAD obtains just over 40 F1. This is somewhat expected, since the pretraining objective is quite different from the fine-tuning task; while masked language modeling requires mainly *local* context around the masked token, question answering needs to align the question to the *global* context of the passage. To bridge this gap, we propose (1) a novel self-supervised method for explicitly pretraining span selection models, and (2) a question answering layer that explicitly aligns a representation of the question to the passage.

We introduce *Splinter* (**sp**an-**l**evel po**inter**), a pretrained model for few-shot question answering. The challenge in defining such a self-supervised task is how to create question-answer pairs from unlabeled data. Our key observation is that one can leverage *recurring spans*: n-grams, such as named entities, which tend to occur multiple times in a given passage (e.g., *"Roosevelt"*, in Figure 2). We emulate question answering by masking all but one instance of each recurring span with a special `[QUESTION]` token, and asking the model to select the correct span for each such token.

To select an answer span for each `[QUESTION]` token *in parallel*, we introduce a question-aware span selection (QASS) layer, which uses the `[QUESTION]` token's representation to select the answer span. The QASS layer seamlessly integrates with fine-tuning on real question-answer pairs; we append the `[QUESTION]` token to the input question, and use the QASS layer to select the answer span (Figure 3). This is unlike existing models for span selection, which do not include an explicit question representation. The compatibility between pretraining and fine-tuning makes Splinter an effective few-shot learner.

Splinter exhibits surprisingly high performance given only a few training examples throughout a wide variety of question answering benchmarks from the MRQA 2019 shared task (Fisch et al., 2019). For example, experiments on SQuAD show that Splinter-base achieves 72.7 F1 with only 128 examples, outperforming all baselines by very wide margins. An ablation study shows that both the pretraining method and the QASS layer improve performance, with pretraining accounting for most of the gains when the dataset is very small, while the QASS layer gradually becomes more dominant as the number of training examples grows. Overall, our results highlight the importance of designing objectives and architectures in the few-shot setup, where an appropriate inductive bias can lead to dramatic performance improvements.[1]

2 Background

Extractive question answering is a common task in NLP, where the goal is to select a contiguous span a from a given text T that answers a question Q. This format was popularized by SQuAD (Rajpurkar et al., 2016), and has since been adopted by several datasets in various domains (Trischler et al., 2017; Kembhavi et al., 2017) and languages (Lewis et al., 2020; Clark et al., 2020), with some extensions

[1]Our code, models, and datasets are publicly available: `https://github.com/oriram/splinter`

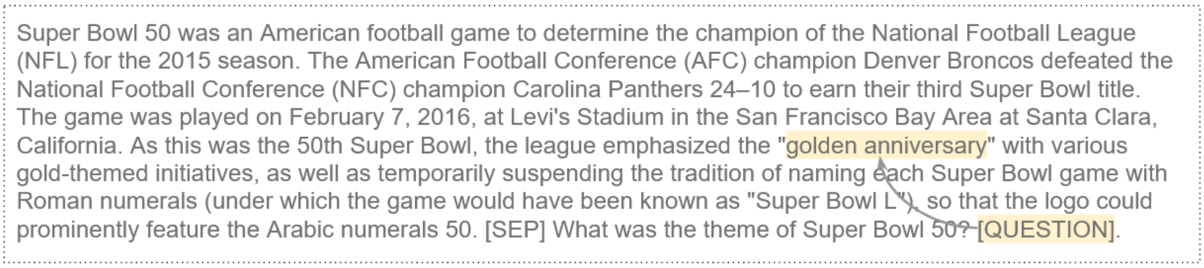

Figure 3: An example of our fine-tuning setup, taken from the development set of SQuAD. The question, followed by the special `[QUESTION]` token, is concatenated to the context. The special token's representation is then used to select the answer span.

allowing for unanswerable questions (Levy et al., 2017; Rajpurkar et al., 2018) or multiple answer spans (Dua et al., 2019; Dasigi et al., 2019). In this work, we follow the assumptions in the recent MRQA 2019 shared task (Fisch et al., 2019) and focus on questions whose answer is a single span.

The standard approach uses a pretrained encoder, such as BERT (Devlin et al., 2019), and adds two parameter vectors $\mathbf{s}, \mathbf{e}$ to the pretrained model in order to detect the start and end of the answer span a, respectively. The input text T and question Q are concatenated and fed into the encoder, producing a contextualized token representation $\mathbf{x}_i$ for each token in the sequence. To predict the answer span's start position, a probability distribution is induced over the entire sequence by computing the inner product of $\mathbf{s}$ with every token representation (the end position's distribution is computed similarly):

$$P(s = i \mid T, Q) = \frac{\exp(\mathbf{x}_i^\top \mathbf{s})}{\sum_j \exp(\mathbf{x}_j^\top \mathbf{s})},$$

$$P(e = i \mid T, Q) = \frac{\exp(\mathbf{x}_i^\top \mathbf{e})}{\sum_j \exp(\mathbf{x}_j^\top \mathbf{e})}.$$

The parameters $\mathbf{s}, \mathbf{e}$ are trained during fine-tuning, using the cross-entropy loss with the start and end positions of the gold answer span.

This approach assumes that each token's representation $\mathbf{x}_i$ is contextualized with respect to the question. However, the masked language modeling objective does not necessarily encourage that form of long-range contextualization in the pretrained model, since many of the masked tokens can be resolved from local cues. Fine-tuning the attention patterns of pretrained masked language models may thus entail an extensive learning effort, difficult to achieve with only a handful of training examples. We overcome this issue by (1) pretraining directly for span selection, and (2) explicitly representing the question with a specialized [QUESTION] token, used to detect the answer in the input text.

3 Splinter

We formulate a new proxy task for pretraining question answering using unlabeled text: *recurring span selection*. We replace spans that appear multiple times in the given text with a special `[QUESTION]` token, except for one occurrence, which acts as the "answer" span for each (masked) "question". The prediction layer is a modification of the standard span selection layer, which replaces the static start and end parameter vectors, $\mathbf{s}$ and $\mathbf{e}$, with dynamically-computed boundary detectors based on the contextualized representation of each `[QUESTION]` token. We reuse this architecture when fine-tuning on question-answer pairs by adding a `[QUESTION]` token at the end of the actual question, thus aligning the pretraining objective with the fine-tuning task.

3.1 Pretraining: Recurring Span Selection

Given an input text T, we find all *recurring spans*: n-grams that appear more than once in the same text. For each set of identical recurring spans R, we select a single occurrence as the *answer* a and replace all other occurrences with a single `[QUESTION]` token. The goal of recurring span selection is to predict the correct answer a for each `[QUESTION]` token $q \in R \setminus \{a\}$.

Figure 2 illustrates this process. In the given passage, the span *"Roosevelt"* appears three times. Two of its instances (the second and third) are replaced with `[QUESTION]`, while one instance (the first) becomes the answer, and remains intact. After masking, the sequence is passed through a transformer encoder, producing contextualized token representations. The model is then tasked with predicting the start and end positions of the answer given each [QUESTION] token representation. In Figure 2b, we observe four instances of this prediction task: two for the *"Roosevelt"* cluster, one for the *"Allied countries"* cluster, and one for *"Declaration by United Nations"*.

Span Filtering To focus pretraining on semantically meaningful spans, we use the following definition for "spans", which filters out recurring spans that are likely to be uninformative: (1) spans must begin and end at word boundaries, (2) we consider only maximal recurring spans, (3) spans containing only stop words are ignored, (4) spans are limited to a maximum of 10 tokens. These simple heuristic filters do not require a model, as opposed to masking schemes in related work (Glass et al., 2020; Ye et al., 2020; Guu et al., 2020), which require part-of-speech taggers, constituency parsers, or named entity recognizers.

Cluster Selection We mask a random subset of recurring span clusters in each text, leaving some recurring spans untouched. Specifically, we replace up to 30 spans with [QUESTION] from each input passage.

3.2 Model: Query-Aware Span Selection

Our approach converts texts into a set of questions that need to be answered simultaneously. The standard approach for extractive question answering (Devlin et al., 2019) is inapplicable, because it uses fixed start and end vectors. Since we have multiple questions, we replace the standard parameter vectors $\mathbf{s}, \mathbf{e}$ with *dynamic* start and end vectors $\mathbf{s}_q, \mathbf{e}_q$, computed from each [QUESTION] token q:

$$\mathbf{s}_q = \mathbf{S}\mathbf{x}_q \qquad \mathbf{e}_q = \mathbf{E}\mathbf{x}_q$$

Here, $\mathbf{S}, \mathbf{E}$ are parameter matrices, which extract ad hoc start and end position detectors $\mathbf{s}_q, \mathbf{e}_q$ from the given [QUESTION] token's representation $\mathbf{x}_q$. The rest of our model follows the standard span selection model by computing the start and end positions' probability distributions. The model can also be viewed as two bilinear functions of the question representation $\mathbf{x}_q$ with each token in the sequence $\mathbf{x}_i$, similar to Dozat and Manning (2017):

$$P(s = i \mid T, q) = \frac{\exp(\mathbf{x}_i^\top \mathbf{S}\mathbf{x}_q)}{\sum_j \exp(\mathbf{x}_j^\top \mathbf{S}\mathbf{x}_q)}$$

$$P(e = i \mid T, q) = \frac{\exp(\mathbf{x}_i^\top \mathbf{E}\mathbf{x}_q)}{\sum_j \exp(\mathbf{x}_j^\top \mathbf{E}\mathbf{x}_q)}$$

Finally, we use the answer's gold start and end points (s_a, e_a) to compute the cross-entropy loss:

$$-\frac{1}{2}\Big(\log P(s = s_a \mid T, q) + \log P(e = e_a \mid T, q)\Big)$$

We refer to this architecture as the question-aware span selection (QASS) layer.

3.3 Fine-Tuning

After pretraining, we assume access to labeled examples, where each training instance is a text T, a question Q, and an answer a that is a span in T. To make this setting similar to pretraining, we simply append a [QUESTION] token to the input sequence, immediately after the question Q (see Figure 3). Selecting the answer span then proceeds exactly as during pretraining. Indeed, the advantage of our approach is that in both pretraining and fine-tuning, the [QUESTION] token's representation captures the meaning of the question that is then used to select the span from context.

4 A Few-Shot QA Benchmark

To evaluate how pretrained models work when only a small amount of labeled data is available for fine-tuning, we simulate various low-data scenarios by sampling subsets of training examples from larger datasets. We use a subset of the MRQA 2019 shared task (Fisch et al., 2019), which contains extractive question answering datasets in a unified format, where the answer is a single span in the given text passage.

Large Datasets Split I of of the MRQA shared task contains 6 large question answering datasets: SQuAD (Rajpurkar et al., 2016), NewsQA (Trischler et al., 2017), TriviaQA (Joshi et al., 2017), SearchQA (Dunn et al., 2017), HotpotQA (Yang et al., 2018), Natural Questions (Kwiatkowski et al., 2019). For each dataset, we sample smaller training datasets from the original training set with sizes changing on a logarithmic scale, from 16 to 1,024 examples. To reduce variance, for each training set size, we sample 5 training sets using different random seeds and report average performance across training sets. We also experiment with fine-tuning the models on the full training sets. All evaluations are conducted on the development set, since the test sets are not publicly available; we thus avoid any hyperparameter tuning or early stopping (see Section 5).

Small Datasets Split II of the MRQA shared task contains datasets that are genuinely small. We select two datasets that were annotated by domain experts: BioASQ (Tsatsaronis et al., 2015) and TextbookQA (Kembhavi et al., 2017). Each of these datasets only has a development set that is publicly available, containing about 1500 examples. For each dataset, we sample 400 examples for evaluation, and follow the same protocol we used for large datasets to sample training sets of 16 to 1,024 examples from the remaining data.

5 Experimental Setup

We describe our experimental setup in detail, including all models and baselines.

5.1 Baselines

Splinter-base shares the same architecture (transformer encoder), vocabulary (cased wordpieces), number of parameters (110M), and data as BERT-base (Devlin et al., 2019). We follow SpanBERT (Joshi et al., 2020) and RoBERTa (Liu et al., 2019), and omit the next sentence prediction objective. In all experiments, we compare Splinter-base to three baselines of the same capacity:

RoBERTa (Liu et al., 2019) A highly-tuned and optimized version of BERT, which is known to perform well on a wide range of natural language understanding tasks.

SpanBERT (Joshi et al., 2020) A BERT-style model that focuses on span representations. SpanBERT is trained by masking spans using a geometric distribution and optimizing two objectives, specifically: (a) masked language modeling, for predicting span tokens from masked tokens, and (b) the span boundary objective, for predicting span tokens from the tokens on the span's boundary.

SpanBERT (Ours) Our reimplementation of SpanBERT, using exactly the same code, data, and hyperparameters as Splinter. This version does not use the span boundary objective, as Joshi et al. (2020) reported no significant improvements from using it in question answering. The baseline aims to control for implementation differences and measure the effect of replacing the masked language modeling objective with recurring span selection.

5.2 Pretraining Implementation

We train Splinter-base using Adam (Kingma and Ba, 2015) for 2.4M training steps with batches of 256 sequences of length 512.[2] The learning rate is warmed up for 10k steps to a maximum value of 10^{-4}, after which linear decay is applied. As in previous work, we use a dropout rate of 0.1 across all layers.

We follow Devlin et al. (2019) and train on English Wikipedia (preprocessed by WikiExtractor as in Attardi (2015)) and the Toronto BookCorpus (Zhu et al., 2015). We base our implementation on the official TensorFlow implementation of BERT, and train on a single eight-core v3 TPU (v3-8) on the Google Cloud Platform.

5.3 Fine-Tuning Implementation

For fine-tuning, we use the hyperparameters from the default configuration of the HuggingFace Transformers package (Wolf et al., 2020).[3] Specifically, we fine-tune all models using Adam (Kingma and Ba, 2015) for either 10 epochs or 200 steps (whichever is larger). The full-size datasets are trained for 2 epochs. We set the batch size to 12 and use a maximal learning rate of $3 \cdot 10^{-5}$, which warms up in the first 10% of the steps, and then decays linearly.

An interesting question is how to fine-tune the QASS layer parameters (i.e., the S and E matrices in Section 3.2). In our implementation, we chose to discard the pretrained values and fine-tune from a random initialization, due to the possible discrepancy between span statistics in pretraining and fine-tuning datasets. However, we report results on fine-tuning without resetting the QASS parameters in the ablation study (Section 6.3).

6 Results

Our experiments show that Splinter dramatically improves performance in the challenging few-shot setting, unlocking the ability to train a reasonable question answering model with only 128 examples. When training in the big data regime, Splinter performs competitively (and sometimes better) with other baselines. Ablation studies demonstrate the

[2] We used this setting to approximate SpanBERT's hyperparameter setting in terms of epochs. That said, SpanBERT-base was trained for a quarter of the steps (600k steps) using four times as many examples per batch (1024 sequences). We train additional baselines that control for this difference (see Section 5.1).

[3] We did rudimentary tuning on the number of steps only, using a held-out portion of the SQuAD training set, since our training sets can be too small for the default values (e.g., running 10 epochs on 16 examples results in 20 update steps).

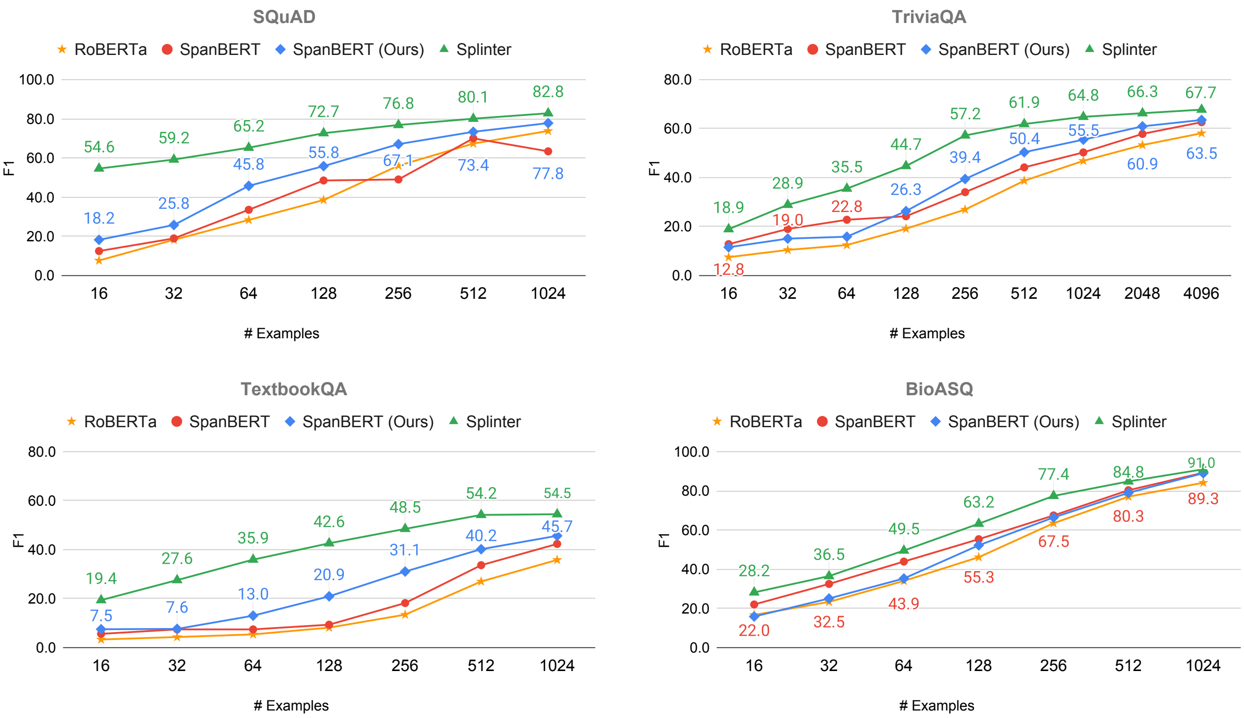

Figure 4: Performance (F1) of Splinter-base (green line, triangular points), compared to all baselines as a function of the number of training examples on 4 datasets. Each point reflects the average performance across 5 randomly-sampled training sets of the same size.

contributions of both the pretraining and the QASS layer.

6.1 Few-Shot Learning

Figure 4 shows the F1 score (Rajpurkar et al., 2016) of Splinter-base, plotted against all baselines for four datasets, as a function of the number of training examples (see Figure 6 in Appendix A for the remaining four datasets). In addition, Table 1 shows the performance of individual models when given 16, 128, and 1024 training examples. It is evident that Splinter outperforms all other baselines by large margins across the board.

Let us examine the results of SQuAD, for example. Given 16 training examples, Splinter obtains 55 F1, significantly above the best baseline's 18 F1 (from our implementation of SpanBERT). When the number of training examples is 128, Splinter achieves 73 F1 while SpanBERT (Ours) has 56 F1. The performance of RoBERTa and SpanBERT in this scenario is 39 F1 and 49 F1, respectively. When considering 1024 examples, there is a 5-point margin between Splinter (83 F1) and SpanBERT (78 F1). The same trend is seen in the other datasets, whether they are sampled from larger datasets (e.g. TriviaQA) or originally small. In TextbookQA, for instance, we observe absolute gaps of 9 to 23 F1 between Splinter and the next-best baseline.

6.2 High-Resource Regime

Table 1 also shows the performance when fine-tuning on the entire training set when an order of 100,000 examples are available (see *Large Datasets* in Section 4). Even though Splinter was designed for few-shot question answering, it performs competitively in the high-resource regime as well, reaching the best result in five out of six datasets.

6.3 Ablation Study

We perform an ablation study to better understand the independent contributions of the pretraining scheme and the QASS layer. We first ablate the effect of pretraining on recurring span selection by applying the QASS layer to pretrained masked language models. We then test whether the QASS layer's pretrained parameters can be reused in Splinter during fine-tuning without reinitializing them in between.

Independent Contribution of the QASS Layer While the query-aware span selection (QASS) layer is motivated by our pretraining method, it can also

Model	SQuAD	TriviaQA	NQ	NewsQA	SearchQA	HotpotQA	BioASQ	TextbookQA
16 Examples								
RoBERTa	7.7	7.5	17.3	1.4	6.9	10.5	16.7	3.3
SpanBERT	12.5	12.8	19.7	6.0	13.0	12.6	22.0	5.6
SpanBERT (Ours)	18.2	11.6	19.6	7.6	13.3	12.5	15.9	7.5
Splinter	**54.6**	**18.9**	**27.4**	**20.8**	**26.3**	**24.0**	**28.2**	**19.4**
128 Examples								
RoBERTa	38.6	19.1	30.1	16.7	27.8	27.3	46.1	8.2
SpanBERT	48.5	24.2	32.2	17.4	34.3	35.1	55.3	9.4
SpanBERT (Ours)	55.8	26.3	36.0	29.5	26.3	36.6	52.2	20.9
Splinter	**72.7**	**44.7**	**46.3**	**43.5**	**47.2**	**54.7**	**63.2**	**42.6**
1024 Examples								
RoBERTa	73.8	46.8	54.2	47.5	54.3	61.8	84.1	35.8
SpanBERT	63.4	50.3	50.2	49.3	60.1	67.4	89.3	42.3
SpanBERT (Ours)	77.8	55.5	59.5	52.2	58.9	64.6	89.0	45.7
Splinter	**82.8**	**64.8**	**65.5**	**57.3**	**67.3**	**70.3**	**91.0**	**54.5**
Full Dataset								
RoBERTa	90.3	74.0	79.6	69.8	81.5	78.7	-	-
SpanBERT	92.0	**77.2**	80.6	**71.3**	80.1	79.6	-	-
SpanBERT (Ours)	92.0	75.8	80.5	71.1	81.4	79.7	-	-
Splinter	**92.2**	76.5	**81.0**	**71.3**	**83.0**	**80.7**	-	-

Table 1: Performance (F1) across all datasets when the number of training examples is 16, 128, and 1024. We also show performance when training on the full-sized large datasets. All models have the same capacity to BERT-base (110M parameters). NQ stands for Natural Questions.

be used independently of Splinter. We apply a randomly-initialized QASS layer to our implementation of SpanBERT, and fine-tune it in the few-shot setting.

Figure 5 shows that replacing the static span selection layer with QASS can significantly improves performance on few-shot question answering, independently. Having said that, most of Splinter's improvements in the extremely low data regime do stem from combining the QASS layer with our pretraining scheme, and this combination still outperforms all other variants as the amount of data grows.

QASS Reinitialization Between pretraining and fine-tuning, we randomly reinitialize the parameters of the QASS layer. We now test the effect of keeping the parameters of the pretrained QASS layer. Intuitively, the more similar the pretraining data is to the downstream task, the better the fully pretrained layer will perform.

Figure 5 shows that the advantage of pretrained QASS is indeed data-dependent, and can result in both performance gains (e.g. extremely low data in SQuAD) and stagnation (e.g. TextbookQA). Roughly, we identify three conditions that determine whether keeping the pretrained head is preferable: (1) when the number of training examples is extremely low, (2) when the target domain is similar to that used at pretraining (e.g., Wikipedia), and (3) when the questions are relatively simple (e.g., SQuAD versus HotpotQA). The latter two conditions pertain to the compatibility between pretraining and fine-tuning tasks; the information learned in the QASS layer is useful as long as the input and output distribution of the task are close to those seen at pretraining time.

7 Related Work

Question answering is an active area of research in NLP. In addition, it has become a standard evaluation suite for testing the language understanding capabilities of language models (Peters et al., 2018; Devlin et al., 2019; Yang et al., 2019; Joshi et al., 2020; Liu et al., 2019), which exhibit state-of-the-art results. The main paradigm suggests fine-tuning on large-scale datasets, e.g. SQuAD (Rajpurkar et al., 2016, 2018). These datasets usually consist of an order of a hundred thousand examples, and thus less focus has been given on the low-resource scenario, where only a handful of training examples is present.

A popular approach for training QA models in such settings is dataset synthesis. This requires both answer generation and question generation

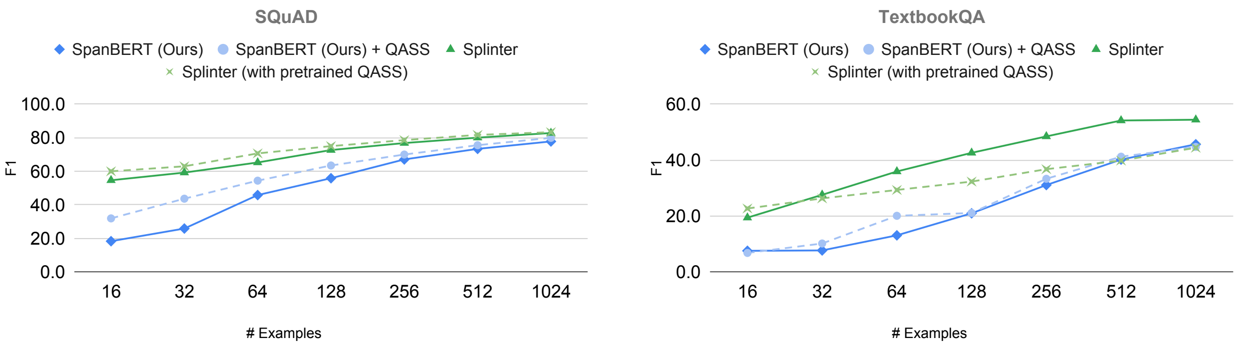

Figure 5: Ablation studies on SQuAD and TextbookQA datasets. We examine the role of QASS layer by fine-tuning it on top of our SpanBERT. In addition, we test whether it is beneficial to keep the parameters of QASS from pretraining (Splinter with Head).

(Lewis et al., 2019; Alberti et al., 2019; Puri et al., 2020). For example, Lewis et al. (2019) generate answer spans given a context, mask the answer, and then "translate" the resulting cloze-style question to a natural question. Glass et al. (2020) also generate answer spans from contexts and mask them, and retrieving passages from the corpus in order to find supporting evidence. To generate meaningful answers, both works assume access to language- and domain-specific NLP tools such as part-of-speech taggers, named-entity recognizers, or syntactic parsers. In addition, each training example corresponds to a single question-answer pair. Our work deviates by exploiting the natural phenomenon of *recurring spans* in order to generate *multiple* question-answer pairs per text passage, without assuming that any language- or domain-dependent models or resources are available.

CorefBERT (Ye et al., 2020) introduced a coreferential reasoning pretraining scheme, where recurring single-token noun phrases are utilized through MLM to train a coreference-aware model. A copy-based objective is incorporated as well. Our pretraining differs from CorefBERT in several ways. First, they assume a POS tagger is at hand. Second, They only consider noun phrases, while Splinter is trained to predict arbitrary spans. Last, CorefBERT trains to "copy" single words, while Splinter points to spans of arbitrary sizes, which better resemble question answering.

8 Conclusion

In this work we explore the few-shot setting of extractive question answering, and demonstrate that existing methods, based on fine-tuning large pretrained language models, fail in this setup. We propose a new pretraining scheme and architecture for span selection that lead to dramatic improvements, reaching decent results even when only an order of a hundred examples are available. Our work shows that choices that are often deemed unimportant when enough data is available, again become crucial in the few-shot setting, opening the door to new methods that take advantage of prior knowledge on the downstream task during model development.

Acknowledgements

This project was funded by the European Research Council (ERC) under the European Unions Horizon 2020 research and innovation programme (grant ERC HOLI 819080), the Blavatnik Fund, the Alon Scholarship, and Intel Corporation. We thank Google's TensorFlow Research Cloud (TFRC) for their support in providing TPUs for this research.

References

Chris Alberti, Daniel Andor, Emily Pitler, Jacob Devlin, and Michael Collins. 2019. Synthetic QA corpora generation with roundtrip consistency. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6168–6173, Florence, Italy. Association for Computational Linguistics.

Giusepppe Attardi. 2015. WikiExtractor.

Jonathan H. Clark, Eunsol Choi, Michael Collins, Dan Garrette, Tom Kwiatkowski, Vitaly Nikolaev, and Jennimaria Palomaki. 2020. TyDi QA: A benchmark for information-seeking question answering in typologically diverse languages. *Transactions of the Association for Computational Linguistics*, 8:454–470.

Pradeep Dasigi, Nelson F. Liu, Ana Marasović, Noah A. Smith, and Matt Gardner. 2019. Quoref:

A reading comprehension dataset with questions requiring coreferential reasoning. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5925–5932, Hong Kong, China. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. 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)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Timothy Dozat and Christopher D. Manning. 2017. Deep biaffine attention for neural dependency parsing. In *ICLR 2017*.

Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. 2019. DROP: A reading comprehension benchmark requiring discrete reasoning over paragraphs. In *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)*, pages 2368–2378, Minneapolis, Minnesota. Association for Computational Linguistics.

Matthew Dunn, Levent Sagun, Mike Higgins, V. Ugur Guney, Volkan Cirik, and Kyunghyun Cho. 2017. SearchQA: A new Q&A dataset augmented with context from a search engine.

Adam Fisch, Alon Talmor, Robin Jia, Minjoon Seo, Eunsol Choi, and Danqi Chen. 2019. MRQA 2019 shared task: Evaluating generalization in reading comprehension. In *Proceedings of the 2nd Workshop on Machine Reading for Question Answering*, pages 1–13, Hong Kong, China. Association for Computational Linguistics.

Michael Glass, Alfio Gliozzo, Rishav Chakravarti, Anthony Ferritto, Lin Pan, G P Shrivatsa Bhargav, Dinesh Garg, and Avi Sil. 2020. Span selection pre-training for question answering. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2773–2782, Online. Association for Computational Linguistics.

Kelvin Guu, Kenton Lee, Z. Tung, Panupong Pasupat, and Ming-Wei Chang. 2020. Realm: Retrieval-augmented language model pre-training. *ArXiv*, abs/2002.08909.

Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S. Weld, Luke Zettlemoyer, and Omer Levy. 2020. SpanBERT: Improving pre-training by representing and predicting spans. *Transactions of the Association for Computational Linguistics*, 8:64–77.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. 2017. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1601–1611, Vancouver, Canada. Association for Computational Linguistics.

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

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR 2015*.

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

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. 2017. RACE: Large-scale ReAding comprehension dataset from examinations. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 785–794, Copenhagen, Denmark. Association for Computational Linguistics.

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

Patrick Lewis, Ludovic Denoyer, and Sebastian Riedel. 2019. Unsupervised question answering by cloze translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4896–4910, Florence, Italy. Association for Computational Linguistics.

Patrick Lewis, Barlas Oguz, Ruty Rinott, Sebastian Riedel, and Holger Schwenk. 2020. MLQA: Evaluating cross-lingual extractive question answering. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7315–7330, Online. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A robustly optimized bert pretraining approach.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Raul Puri, Ryan Spring, Mohammad Shoeybi, Mostofa Patwary, and Bryan Catanzaro. 2020. Training question answering models from synthetic data. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5811–5826, Online. Association for Computational Linguistics.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for SQuAD. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 784–789, Melbourne, Australia. Association for Computational Linguistics.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Adam Trischler, Tong Wang, Xingdi Yuan, Justin Harris, Alessandro Sordoni, Philip Bachman, and Kaheer Suleman. 2017. NewsQA: A machine comprehension dataset. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*, pages 191–200, Vancouver, Canada. Association for Computational Linguistics.

George Tsatsaronis, Georgios Balikas, Prodromos Malakasiotis, Ioannis Partalas, Matthias Zschunke, Michael R. Alvers, Dirk Weissenborn, Anastasia Krithara, Sergios Petridis, Dimitris Polychronopoulos, Yannis Almirantis, John Pavlopoulos, Nicolas Baskiotis, Patrick Gallinari, Thierry Artiéres, Axel-Cyrille Ngonga Ngomo, Norman Heino, Eric Gaussier, Liliana Barrio-Alvers, Michael Schroeder, Ion Androutsopoulos, and Georgios Paliouras. 2015. An overview of the BIOASQ large-scale biomedical semantic indexing and question answering competition. *BMC Bioinformatics*, 16(1):138.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in Neural Information Processing Systems*, volume 32, pages 5753–5763. Curran Associates, Inc.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380, Brussels, Belgium. Association for Computational Linguistics.

Deming Ye, Yankai Lin, Jiaju Du, Zhenghao Liu, Peng Li, Maosong Sun, and Zhiyuan Liu. 2020. Coreferential Reasoning Learning for Language Representation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7170–7186, Online. Association for Computational Linguistics.

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *The IEEE International Conference on Computer Vision (ICCV)*.

A Further Results

Figure 6 shows the results on the four few-shot question answering datasets not included in Figure 4.

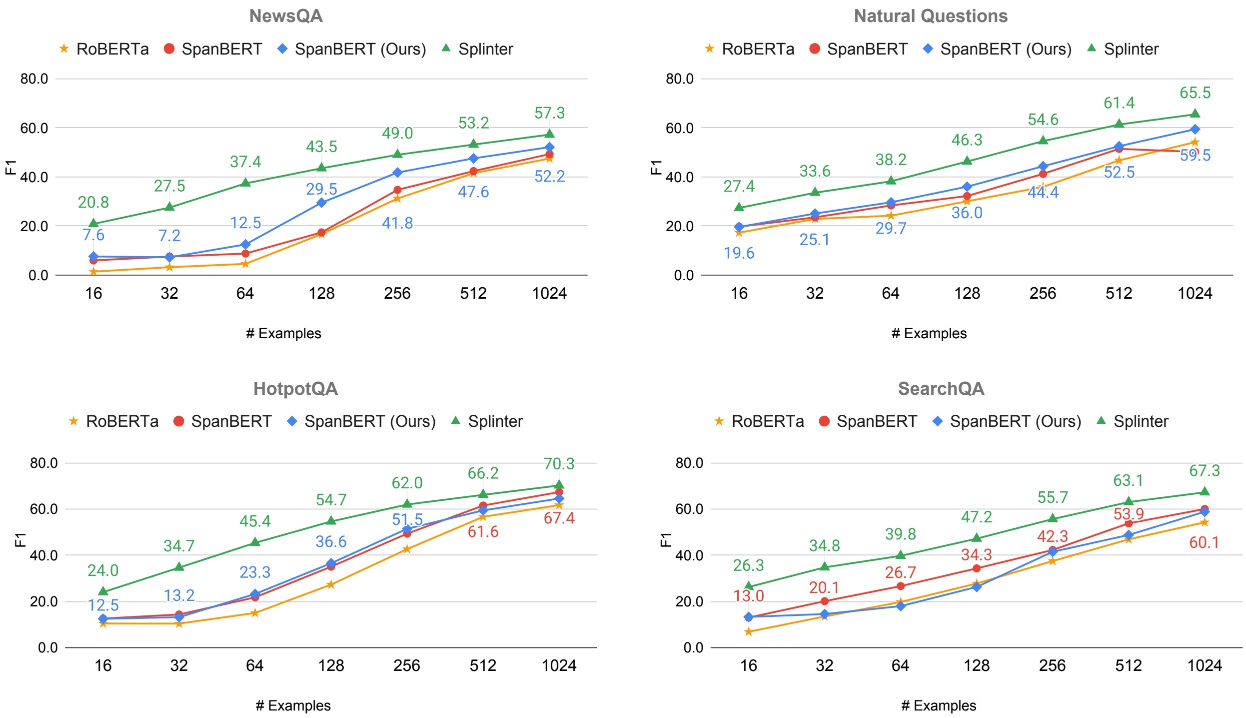

Figure 6: Results complementary to Table 1. Performance (F1) of Splinter-base (green line, triangular points), compared to all baselines as a function of the number of training examples on 4 datasets. Each point reflects the average performance across 5 randomly-sampled training sets of the same size.