Few-Shot Question Answering by Pretraining Span Selection

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

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

Abstract

In several question answering 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, and observe that standard models perform poorly, highlighting the discrepancy between current pretraining objectives and question answering. We propose a new pretraining scheme tailored for question answering: recurring span selection. 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 on SQuAD with only 128 training examples), while maintaining competitive performance in the high-resource setting.[1]

1 Introduction

The standard approach to question answering is to pretrain a masked language model on raw text, and then fine-tune it with a span selection layer on top (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 answering examples are available. 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 English Wikipedia, and attempt to crowdsource question-answer pairs in other languages or domains of expertise (Tsatsaronis et al., 2015; 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?

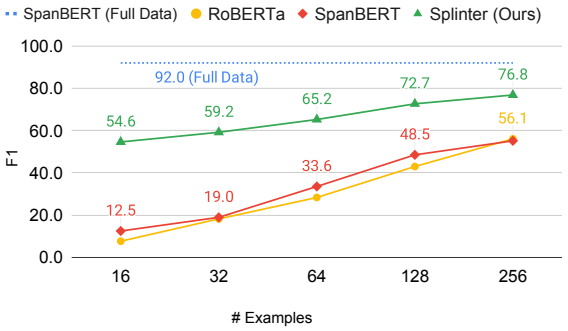

Figure 1: Performance of SpanBERT (red) and RoBERTa (yellow) base-size models on SQuAD, given different amounts of training examples. Our model (Splinter, green) dramatically improves performance. SpanBERT-base trained on the *full* training data of SQuAD (blue, dashed) is shown for reference.

We investigate the task of few-shot question answering by sampling small training sets from existing question answering benchmarks. Despite the use of pretrained models, the standard approach yields poor results when fine-tuning on few examples (Figure 1). For example, RoBERTa-base fine-tuned on 128 question-answer pairs from SQuAD obtains around 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 with the *global* context of the pas-

[*] Equal contribution.

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

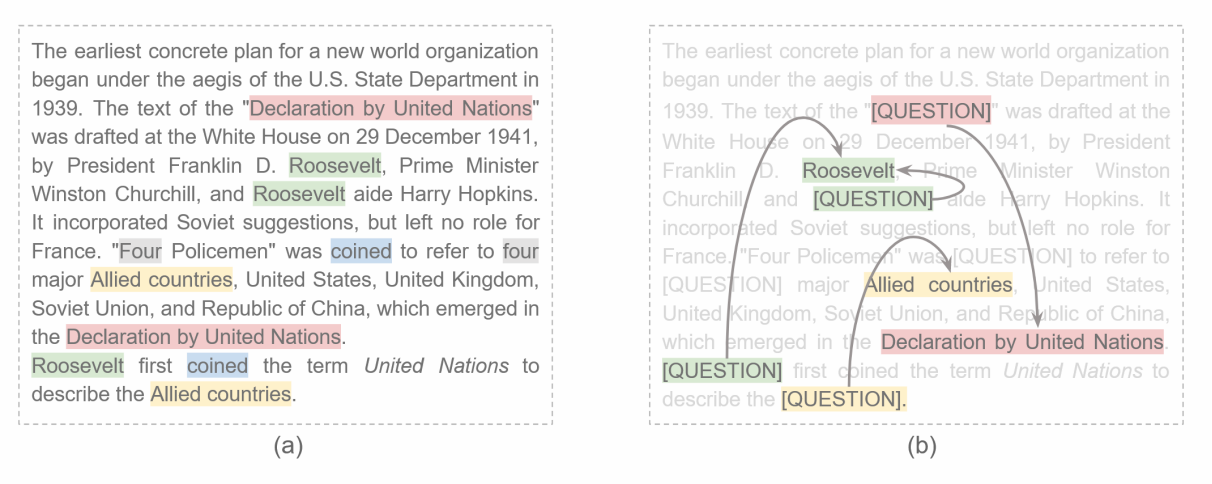

Figure 2: An example paragraph before (a) and after (b) masking recurring spans. Each color represents a different cluster of spans. After masking recurring spans (replacing each with a single `[QUESTION]` token), only one span from each cluster remains unmasked, and is considered the correct answer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

sage. To bridge this gap, we propose (1) a novel self-supervised method for pretraining span selection models, and (2) a question answering layer that aligns a representation of the question with the text.

We introduce *Splinter* (**spl**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 simply 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 de an explicit question representation. The atibility between pretraining and fine-tuning s Splinter an effective few-shot learner.

Splinter exhibits surprisingly high performance given only a few training examples throughout a variety of benchmarks from the MRQA 2019 shared task (Fisch et al., 2019). For example, Splinter-base achieves 72.7 F1 on SQuAD with only 128 examples, outperforming all baselines by a very wide margin. An ablation study shows that the pretraining method and the QASS layer itself (even without pretraining) both contribute to improved performance. Analysis indicates that Splinter's representations change significantly less during fine-tuning compared to the baselines, suggesting that our pretraining is more adequate for question answering. Overall, our results highlight the importance of designing objectives and architectures in the few-shot setting, where an appropriate inductive bias can lead to dramatic performance improvements.

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 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,

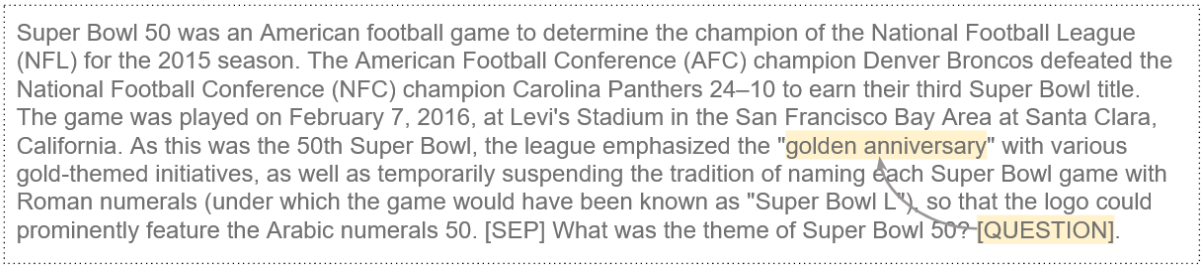

Figure 3: An example of our fine-tuning setup, taken from the development set of SQuAD. The question, followed by the `[QUESTION]` token, is concatenated to the context. The `[QUESTION]` token's representation is then used to select the answer span.

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 position s and end position e 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 start position of the answer span, a probability distribution is induced over the entire sequence by computing the inner product of a learned vector $\mathbf{s}$ with every token representation (the end position is computed similarly using a vector $\mathbf{e}$):

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

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

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 representation $\mathbf{x}_i$ is contextualized with respect to the question. However, the masked language modeling objective does not necessarily encourage this 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 single vector, used to detect the answer in the input text.

3 Splinter

We formulate a new task for pretraining question answering from 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) cloze-style "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. We refer to our pretrained model as *Splinter*.

3.1 Pretraining: Recurring Span Selection

Given an input text T, we find all *recurring spans*: arbitrary 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.[2] The goal of recurring span selection is to predict the correct answer a for a given `[QUESTION]` token $q \in R \setminus \{a\}$, each q thus acting as an independent *cloze-style question*.

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 to-

[2] In practice, only some sets of recurring spans are processed; see *Cluster Selection* below.

ken 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"*.

Taking advantage of recurring *words* in a passage (restricted to nouns or named entities) was proposed in past work as a signal for coreference (Kocijan et al., 2019; Ye et al., 2020). We further discuss this connection in Section 7.

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] This number was chosen to resemble the 15% token-masking ratio of Joshi et al. (2020). Note that in our case, the number of masked tokens is greater than the number of questions.

3.2 Model: Question-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 $\mathbf{x}_q$.

[3] In some cases, the last cluster may have more than one unmasked span.

The rest of our model follows the standard span selection model by computing the start and end position 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:

$$-\log P(s = s_a \mid T, q) - \log P(e = e_a \mid T, q)$$

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 representation captures information about 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.

Split I 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), and 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. Since Split I of the MRQA shared task does not contain test sets, we evaluate using the official development sets as our test sets.

We also select two datasets from Split II of the MRQA shared task 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 in MRQA, containing about 1,500 examples. For each dataset, we sample 400 examples for evaluation (test set), and follow the same protocol we used for large datasets to sample training sets of 16 to 1,024 examples from the remaining data.

To maintain the few-shot setting, every dataset in our benchmark has well-defined training and test sets. To tune hyperparameters, one needs to extract validation data from each training set. For simplicity, we do not perform hyperparameter tuning or model selection (see Section 5), and thus use all of the available few-shot data for training.

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 (Vaswani et al., 2017)), vocabulary (cased wordpieces), and number of parameters (110M) with SpanBERT-base (Joshi et al., 2020). 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 contiguous spans of tokens and optimizing two objectives: (a) masked language modeling, which predicts each masked token from its own vector representation; (b) the span boundary objective, which predicts each masked token from the representations of the unmasked tokens at the start and end of the masked span.

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

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.[4] The learning rate is warmed up for 10k steps to a maximum value of 10^{-4}, after which it decays linearly. 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).[5] Specifically, we train all models using Adam (Kingma and Ba, 2015) with bias-corrected moment estimates for few-shot learning (Zhang et al., 2021). When fine-tuning on 1024 examples or less, we train for either 10 epochs or 200 steps (whichever is larger). For full-size datasets, we train 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 $\mathbf{S}$ and $\mathbf{E}$ matrices in Section 3.2). In our implementation, we chose to discard the pretrained values and fine-tune

[4] 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). See Section 5.1 for additional baselines that control for this difference.

[5] 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).

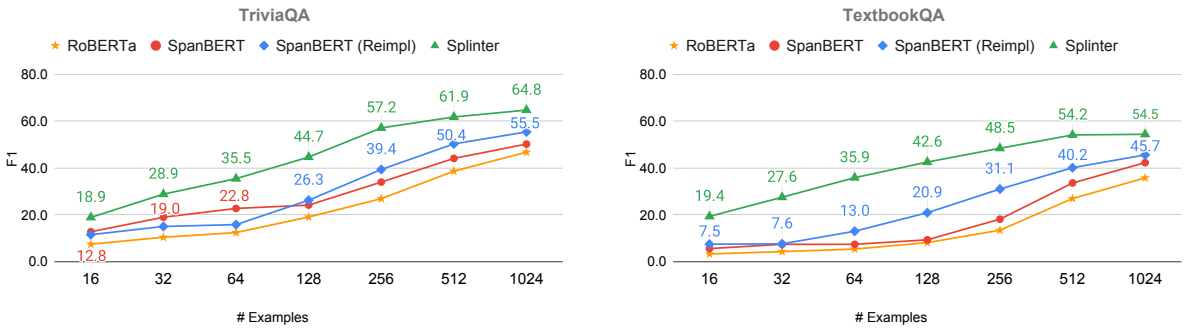

Figure 4: Performance (F1) of Splinter-base (green), compared to all baselines as a function of the number of training examples on two datasets. Each point reflects the average performance across 5 randomly-sampled training sets of the same size.

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 as an 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 question answering models with only hundreds of examples. When trained on large datasets with an order of 100,000 examples, Splinter is competitive with (and often better than) the baselines. Ablation studies demonstrate the contributions of both recurring span selection 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 two datasets, TriviaQA and TextbookQA, as a function of the number of training examples (see Figure 6 in the appendix for the remaining datasets). In addition, Table 1 shows the performance of individual models when given 16, 128, and 1024 training examples across all datasets (see Table 3 in the appendix for additional performance and standard deviation statistics). It is evident that Splinter outperforms all baselines by large margins.

Let us examine the results on SQuAD, for example. Given 16 training examples, Splinter obtains 54.6 F1, significantly higher than the best baseline's 18.2 F1. When the number of training examples is 128, Splinter achieves 72.7 F1, outperforming the baselines by 17 points (our reimplementation of SpanBERT) to 30 (RoBERTa). When considering 1024 examples, there is a 5-point margin between Splinter (82.8 F1) and SpanBERT (77.8 F1). The same trend is seen in the other datasets, whether they are in-domain sampled from larger datasets (e.g. TriviaQA) or not; 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. Even though Splinter was designed for few-shot question answering, it reaches the best result in five out of six datasets. This result suggests that when the target task is extractive question answering, it is better to pretrain with our recurring span selection task than with masked langauge modeling, regardless of the number of annotated training examples.

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 reinitializion.

Independent Contribution of the QASS Layer While the QASS layer is motivated by our pretraining scheme, it can also be used without pretraining. We apply a randomly-initialized QASS layer to our implementation of SpanBERT, and fine-tune it in the few-shot setting. Figure 5 shows the results of this ablation study for two datasets (see Figure 7 in the appendix for more datasets). We observe

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 (Reimpl)	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	43.0	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 (Reimpl)	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	77.8	50.3	57.5	49.3	60.1	67.4	89.3	42.3
SpanBERT (Reimpl)	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 (Reimpl)	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 (MRQA version). All models have the same capacity to BERT-base (110M parameters). NQ stands for Natural Questions.

that replacing the static span selection layer with QASS can significantly improve performance on few-shot question answering. 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 fine-tuning with the QASS layer's pretrained parameters; intuitively, the more similar the pretraining data is to the task, the better the pretrained layer will perform. Figure 5 shows that the advantage of reusing the pretrained QASS is data-dependent, and can result in both performance gains (e.g. extremely low data in SQuAD) and stagnation (e.g. BioASQ with 256 examples or more). Other datasets exhibit similar trends (see appendix). 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.

Model	Representation Similarity
RoBERTa	0.29
SpanBERT	0.23
SpanBERT (Reimpl)	0.19
Splinter	**0.89**

Table 2: Cosine similarity of the representations produced by the transformer encoder before and after fine-tuning on 128 SQuAD examples.

6.4 Analysis

The recurring span selection objective was designed to emulate extractive question answering using unlabeled text. How similar is it to the actual target task? To answer this question, we measure how much each pretrained model's functionality has changed after fine-tuning on 128 examples of SQuAD. For the purpose of this analysis, we measure change in functionality by examining the vector representation of each token as produced by the transformer encoder; specifically, we measure the cosine similarity between the vector produced by

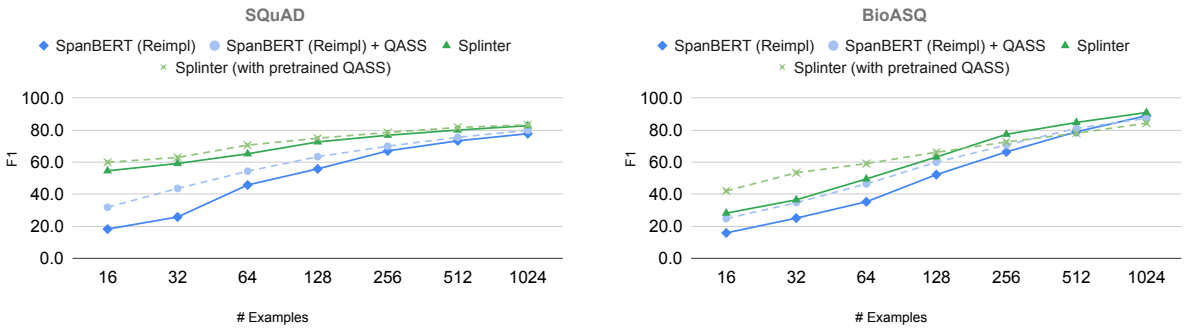

Figure 5: Ablation studies on SQuAD and BioASQ datasets. We examine the role of the QASS layer by fine-tuning it on top of our reimplementation of SpanBERT. In addition, we test whether it is beneficial to keep the pretrained parameters of the QASS layer when fine-tuning Splinter.

the pretrained model and the one produced by the fine-tuned model, given exactly the same input. We average these similarities across every token of 200 examples from SQuAD's test set.

Table 2 shows that Splinter's outputs are very similar before and after fine-tuning (0.89 average cosine similarity), while the other models' representations seem to change drastically. This suggests that fine-tuning with even 128 question-answering examples makes significant modifications to the functionality of pretrained masked language models. Splinter's pretraining, on the other hand, is much more similar to the fine-tuning task, resulting in much more modest changes to the produced vector representations.

7 Related Work

The remarkable results of GPT-3 (Brown et al., 2020) have inspired a renewed interest in few-shot learning. While some work focuses on classification tasks (Schick and Schütze, 2020; Gao et al., 2021), our work investigates few-shot learning in the context of extractive question answering.

One approach to this problem is to create synthetic text-question-answer examples. Both Lewis et al. (2019) and Glass et al. (2020) use the traditional NLP pipeline to select noun phrases and named entities in Wikipedia paragraphs as potential answers, which are then masked from the context to create pseudo-questions. Lewis et al. (2019) use methods from unsupervised machine translation to translate the pseudo-questions into real ones, while Glass et al. (2020) keep the pseudo-questions but use information retrieval to find new text passages that can answer them. Both works assume access to language- and domain-specific NLP tools such as part-of-speech taggers, syntactic parsers, and named-entity recognizers, which might not always be available. Our work deviates from this approach by exploiting the natural phenomenon of *recurring spans* in order to generate multiple question-answer pairs per text passage, without assuming any language- or domain-specific models or resources are available beyond plain text.

Similar ideas to recurring span selection were used for creating synthetic coreference resolution examples (Kocijan et al., 2019; Varkel and Globerson, 2020), which mask single words that occur multiple times in the same context. CorefBERT (Ye et al., 2020) combines this approach with a copy mechanism for predicting the masked word during pretraining, alongside the masked language modeling objective. Unlike our approach, which was designed to align well with span selection, CorefBERT masks only *single-word nouns* (rather than arbitrary spans) and replaces each token in the word with a separate mask token (rather than a single mask for the entire multi-token word). Therefore, it does not emulate extractive question answering. We did not add CorefBERT as a baseline since the performance of both CorefBERT-base and CorefBERT-large was lower than SpanBERT-base's performance on the full-data MRQA benchmark, and pretraining CorefBERT from scratch was beyond our available computational resources.

8 Conclusion

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 surprisingly good 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, the Yandex Initiative for Machine Learning and Intel Corporation. We thank Google's TPU Research Cloud (TRC) for their support in providing TPUs for this research.

References

Giuseppe Attardi. 2015. WikiExtractor.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*.

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.

Tianyu Gao, Adam Fisch, and Danqi Chen. 2021. Making pre-trained language models better few-shot learners. In *Association for Computational Linguistics (ACL)*.

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, Zora Tung, Panupong Pasupat, and Mingwei Chang. 2020. Retrieval augmented language model pre-training. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 3929–3938. PMLR.

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*.

Vid Kocijan, Oana-Maria Camburu, Ana-Maria Cretu, Yordan Yordanov, Phil Blunsom, and Thomas Lukasiewicz. 2019. WikiCREM: A large unsupervised corpus for coreference resolution. 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 4303–4312, Hong Kong, China. Association for Computational Linguistics.

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.

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.

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.

Timo Schick and Hinrich Schütze. 2020. Exploiting cloze questions for few shot text classification and natural language inference.

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.

Yuval Varkel and Amir Globerson. 2020. Pre-training mention representations in coreference models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8534–8540, Online. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30*, pages 5998–6008.

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, 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.

Tianyi Zhang, Felix Wu, Arzoo Katiyar, Kilian Q. Weinberger, and Yoav Artzi. 2021. Revisiting few-sample BERT fine-tuning. In *ICLR*.

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 Additional Results

Few-Shot Results Figure 6 shows the results on the six few-shot question answering datasets not included in Figure 4. In addition, we give the full raw results (including standard deviation) in Table 3.

Ablation Studies Figure 7 shows results of ablation studies on the six question answering datasets not included in Figure 5.

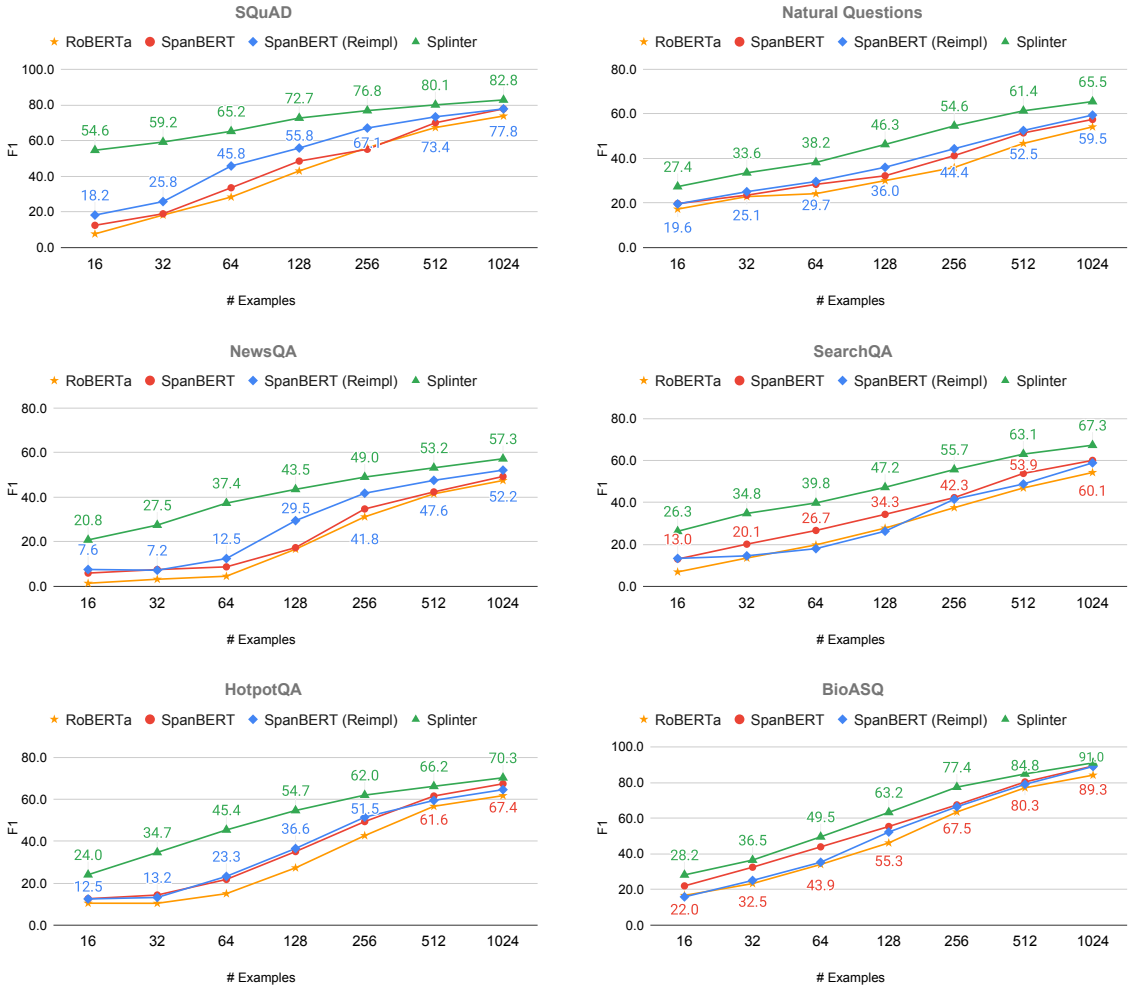

Figure 6: Results complementary to Table 1. Performance (F1) of Splinter-base (green line, triangular points), compared to all baselines as a function of the number of training examples on 4 datasets. Each point reflects the average performance across 5 randomly-sampled training sets of the same size.

Model	SQuAD	TriviaQA	NQ	NewsQA	SearchQA	HotpotQA	BioASQ	TBQA
16 Examples								
RoBERTa	7.7 (4.3)	7.5 (4.4)	17.3 (3.3)	1.4 (0.8)	6.9 (2.7)	10.5 (2.5)	16.7 (7.1)	3.3 (2.1)
SpanBERT	12.5 (5.7)	12.8 (5.4)	19.7 (3.6)	6.0 (1.6)	13.0 (4.2)	12.6 (4.3)	22.0 (4.6)	5.6 (2.5)
(Reimpl)	18.2 (6.7)	11.6 (2.1)	19.6 (3.0)	7.6 (4.1)	13.3 (6.0)	12.5 (5.5)	15.9 (4.4)	7.5 (2.9)
Splinter	**54.6** (6.4)	**18.9** (4.1)	**27.4** (4.6)	**20.8** (2.7)	**26.3** (3.9)	**24.0** (5.0)	**28.2** (4.9)	**19.4** (4.6)
32 Examples								
RoBERTa	18.2 (5.1)	10.5 (1.8)	22.9 (0.7)	3.2 (1.7)	13.5 (1.8)	10.4 (1.9)	23.3 (6.6)	4.3 (0.9)
SpanBERT	19.0 (4.6)	19.0 (4.8)	23.5 (0.9)	7.5 (1.3)	20.1 (3.9)	14.4 (2.9)	32.5 (3.5)	7.4 (1.1)
(Reimpl)	25.8 (7.7)	15.1 (6.4)	25.1 (1.6)	7.2 (4.6)	14.6 (8.5)	13.2 (3.5)	25.1 (3.3)	7.6 (2.3)
Splinter	**59.2** (2.1)	**28.9** (3.1)	**33.6** (2.4)	**27.5** (3.2)	**34.8** (1.8)	**34.7** (3.9)	**36.5** (3.2)	**27.6** (4.3)
64 Examples								
RoBERTa	28.4 (1.7)	12.5 (1.4)	24.2 (1.0)	4.6 (2.8)	19.8 (2.4)	15.0 (3.9)	34.0 (1.8)	5.4 (1.1)
SpanBERT	33.6 (4.3)	22.8 (2.6)	28.4 (1.8)	8.8 (2.4)	26.7 (2.9)	21.8 (1.5)	43.9 (4.5)	7.4 (1.2)
(Reimpl)	45.8 (3.3)	15.9 (6.4)	29.7 (1.5)	12.5 (4.3)	18.0 (4.6)	23.3 (1.1)	35.3 (3.1)	13.0 (6.9)
Splinter	**65.2** (1.4)	**35.5** (3.7)	**38.2** (2.3)	**37.4** (1.2)	**39.8** (3.6)	**45.4** (2.3)	**49.5** (3.6)	**35.9** (3.1)
128 Examples								
RoBERTa	43.0 (7.1)	19.1 (2.9)	30.1 (1.9)	16.7 (3.8)	27.8 (2.5)	27.3 (3.9)	46.1 (1.4)	8.2 (1.1)
SpanBERT	48.5 (7.3)	24.2 (2.1)	32.2 (3.2)	17.4 (3.1)	34.3 (1.1)	35.1 (4.2)	55.3 (3.8)	9.4 (3.0)
(Reimpl)	55.8 (3.7)	26.3 (2.1)	36.0 (1.9)	29.5 (7.3)	26.3 (4.3)	36.6 (3.4)	52.2 (3.2)	20.9 (5.1)
Splinter	**72.7** (1.0)	**44.7** (3.9)	**46.3** (0.8)	**43.5** (1.3)	**47.2** (3.5)	**54.7** (1.4)	**63.2** (4.1)	**42.6** (2.5)
256 Examples								
RoBERTa	56.1 (5.2)	26.9 (3.5)	36.0 (3.2)	31.2 (2.4)	37.5 (1.7)	42.7 (3.1)	63.5 (1.8)	13.5 (1.9)
SpanBERT	55.2 (8.8)	34.0 (5.7)	41.3 (2.2)	34.7 (4.1)	42.3 (4.1)	49.4 (4.0)	67.5 (3.9)	18.2 (4.5)
(Reimpl)	67.1 (2.1)	39.4 (4.0)	44.4 (3.2)	41.8 (1.8)	41.5 (3.2)	51.5 (2.8)	66.4 (2.8)	31.1 (3.4)
Splinter	**76.8** (0.6)	**57.2** (2.2)	**54.6** (1.2)	**49.0** (0.4)	**55.7** (1.9)	**62.0** (1.6)	**77.4** (2.0)	**48.5** (2.2)
512 Examples								
RoBERTa	67.3 (0.7)	38.7 (3.8)	46.7 (2.2)	41.5 (2.2)	46.9 (1.6)	56.7 (1.3)	77.0 (1.9)	27.0 (2.2)
SpanBERT	70.0 (4.3)	44.2 (2.9)	51.5 (1.8)	42.4 (2.6)	53.9 (3.2)	61.6 (1.7)	80.3 (3.0)	33.7 (3.4)
(Reimpl)	73.4 (0.4)	50.4 (2.8)	52.5 (1.9)	47.6 (1.3)	48.8 (4.1)	59.5 (1.5)	79.0 (1.9)	40.2 (0.8)
Splinter	**80.1** (0.4)	**61.9** (1.8)	**61.4** (1.1)	**53.2** (0.9)	**63.1** (1.6)	**66.2** (0.6)	**84.8** (0.9)	**54.2** (1.7)
1024 Examples								
RoBERTa	73.8 (0.8)	46.8 (0.9)	54.2 (1.1)	47.5 (1.1)	54.3 (1.2)	61.8 (1.3)	84.1 (1.1)	35.8 (2.0)
SpanBERT	77.8 (0.9)	50.3 (4.0)	57.5 (0.9)	49.3 (2.0)	60.1 (2.2)	67.4 (1.6)	89.3 (0.6)	42.3 (1.9)
(Reimpl)	77.8 (0.6)	55.5 (1.9)	59.5 (1.7)	52.2 (1.2)	58.9 (1.9)	64.6 (1.2)	89.0 (1.8)	45.7 (1.5)
Splinter	**82.8** (0.8)	**64.8** (0.9)	**65.5** (0.5)	**57.3** (0.8)	**67.3** (1.3)	**70.3** (0.8)	**91.0** (1.0)	**54.5** (1.5)

Table 3: Average performance (F1) across all datasets and training set sizes. We add the standard deviation over the five seeds for each setting in parentheses. NQ and TBQA stand for Natural Questions and TextbookQA respectively. (Reimpl) stands for the SpanBERT (Reimpl) baseline (see Section 5.1).

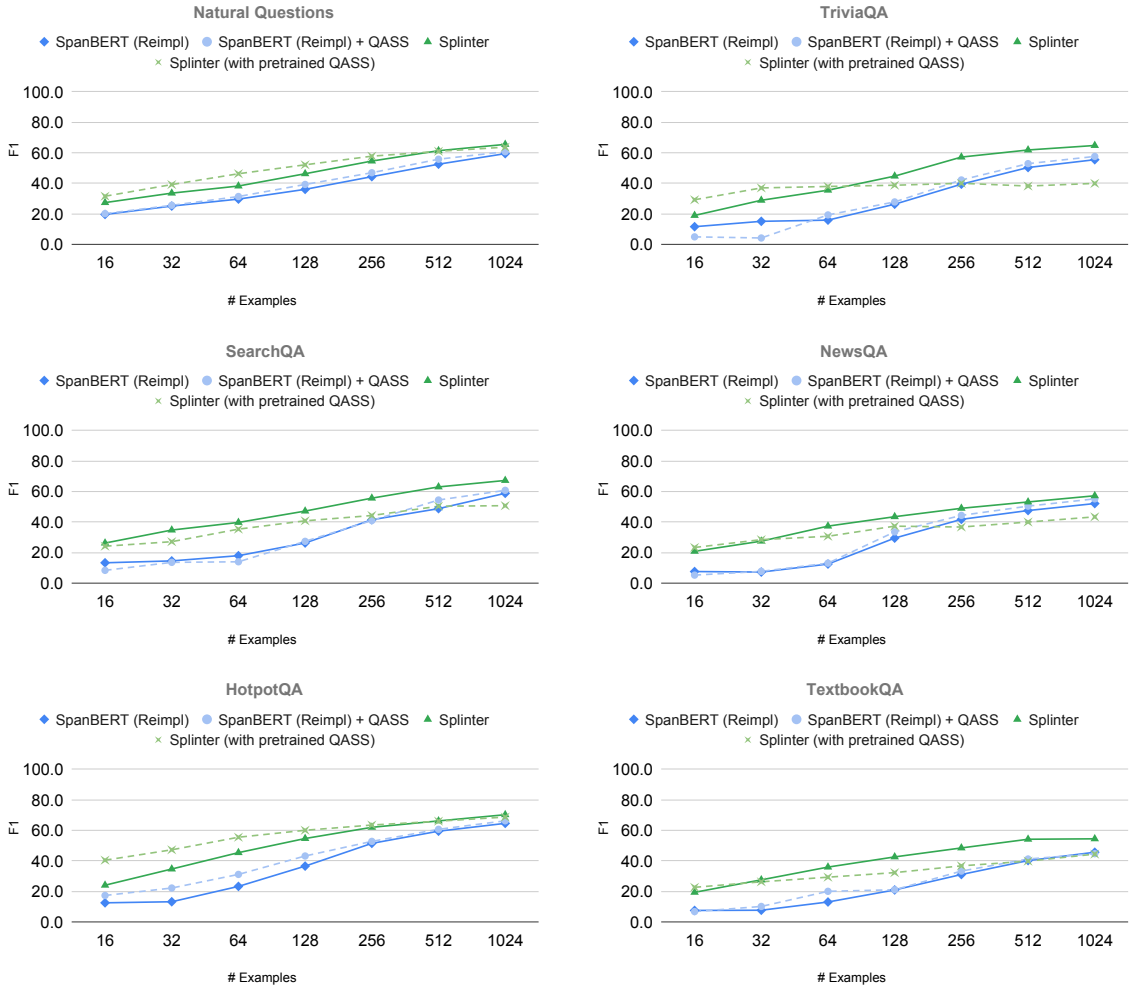

Figure 7: Results complementary to ablation studies (Section 6.3). We examine the role of QASS layer by fine-tuning it on top of our SpanBERT. In addition, we test whether it is beneficial to keep the parameters of QASS from pretraining (Splinter with Head).