SpanBERT: Improving Pre-training by Representing and Predicting Spans

Mandar Joshi † Danqi Chen †§ Yinhan Liu §
Daniel S. Weld †ϵ Luke Zettlemoyer †§ Omer Levy §

† Allen School of Computer Science & Engineering, University of Washington, Seattle, WA
{mandar90,weld,lsz}@cs.washington.edu
‡ Computer Science Department, Princeton University, Princeton, NJ
danqic@cs.princeton.edu
ϵ Allen Institute of Artificial Intelligence, Seattle
{danw}@allenai.org
§ Facebook AI Research, Seattle
{danqi,yinhanliu,lsz,omerlevy}@fb.com

Abstract

We present SpanBERT, a pre-training method that is designed to better represent and predict spans of text. Our approach extends BERT by (1) masking contiguous random spans, rather than random tokens, and (2) training the span boundary representations to predict the entire content of the masked span, without relying on the individual token representations within it. SpanBERT consistently outperforms BERT and our better-tuned baselines, with substantial gains on span selection tasks such as question answering and coreference resolution. In particular, with the same training data and model size as BERTlarge, our single model obtains 94.6% and 88.7% F1 on SQuAD 1.1 and 2.0 respectively. We also achieve a new state of the art on the OntoNotes coreference resolution task (79.6% F1), strong performance on the TACRED relation extraction benchmark, and even gains on GLUE.1

1 Introduction

Pre-training methods like BERT (Devlin et al., 2019) have shown strong performance gains using self-supervised training that masks individual words or subword units. However, many NLP tasks involve reasoning about relationships between two or more spans of text. For example, in extractive question answering (Rajpurkar et al., 2016), determining that the “Denver Broncos” is a type of “NFL team” is critical for answering the question “Which NFL team won Super Bowl 50?” Such spans provide a more challenging target for self supervision tasks, for example, predicting “Denver Broncos” is much harder than predicting only “Denver” when you know the next word is “Broncos”. In this paper, we introduce a span-level pretraining approach that consistently outperforms BERT, with the largest gains on span selection tasks such as question answering and coreference resolution.

We present SpanBERT, a pre-training method that is designed to better represent and predict spans of text. Our method differs from BERT in both the masking scheme and the training objectives. First, we mask random contiguous spans, rather than random individual tokens. Second, we introduce a novel span-boundary objective (SBO) so the model learns to predict the entire masked span from the observed tokens at its boundary. Span-based masking forces the model to predict entire spans solely using the context in which they appear. Furthermore, the SBO encourages the model to store this span-level information at the boundary tokens, which can be easily accessed during the fine-tuning stage. Figure 1 illustrates our approach.

To implement SpanBERT, we build on a well-tuned replica of BERT, which itself substantially outperforms the original BERT. While building on our baseline, we find that pre-training on single segments, instead of two half-length segments with the next sentence prediction (NSP) objective,
\[
\mathcal{L}(\text{football}) = \mathcal{L}_{\text{MLM}}(\text{football}) + \mathcal{L}_{\text{SBO}}(\text{football}) \\
= -\log P(\text{football} \mid x_7) - \log P(\text{football} \mid x_4, x_9, p_3)
\]

Figure 1: An illustration of SpanBERT training. The span an American football game is masked. The SBO uses the output representations of the boundary tokens, \(x_4\) and \(x_9\) (in blue), to predict each token in the masked span.

considerably improves performance on most downstream tasks. Therefore, we add our modifications on top of the tuned single-sequence BERT baseline.

Together, our pre-training process yields models that outperform all BERT baselines on a wide variety of tasks, and reach substantially better performance on span selection tasks in particular. Specifically, our method reaches 94.6\% and 88.7\% F1 on SQuAD 1.1 and 2.0 (Rajpurkar et al., 2016, 2018), respectively—reducing error by as much as 27\% compared with our tuned BERT replica. We also observe similar gains on five additional extractive question answering benchmarks (NewsQA, TriviaQA, SearchQA, HotpotQA, and Natural Questions).\(^2\)

SpanBERT also arrives at a new state of the art on the challenging CoNLL-2012 ("OntoNotes") shared task for document-level coreference resolution, where we reach 79.6\% F1, exceeding the previous top model by 6.6\% absolute. Finally, we demonstrate that SpanBERT also helps on tasks that do not explicitly involve span selection, and show that our approach even improves performance on TACRED (Zhang et al., 2017) and GLUE (Wang et al., 2019).

Whereas others show the benefits of adding more data (Yang et al., 2019) and increasing model size (Lample and Conneau, 2019), this work demonstrates the importance of designing good pre-training tasks and objectives, which can also have a remarkable impact.

## 2 Background: BERT

BERT (Devlin et al., 2019) is a self-supervised approach for pre-training a deep transformer encoder (Vaswani et al., 2017), before fine-tuning it for a particular downstream task. BERT optimizes two training objectives—masked language model (MLM) and next sentence prediction (NSP)—which only require a large collection of unlabeled text.

Notation Given a sequence of word or subword tokens \(X = (x_1, x_2, \ldots, x_n)\), BERT trains an encoder that produces a contextualized vector representation for each token:

\[
\text{enc}(x_1, x_2, \ldots, x_n) = x_1, x_2, \ldots, x_n.
\]

**Masked Language Model** Also known as a cloze test, MLM is the task of predicting missing tokens in a sequence from their placeholders. Specifically, a subset of tokens \(Y \subseteq X\) is sampled and substituted with a different set of tokens. In BERT’s implementation, \(Y\) accounts for 15\% of the tokens in \(X\); of those, 80\% are replaced with [MASK], 10\% are replaced with a random token (according to the unigram distribution), and 10\% are kept unchanged. The task is to predict the original tokens in \(Y\) from the modified input.

BERT selects each token in \(Y\) independently by randomly selecting a subset. In SpanBERT, we define \(Y\) by randomly selecting contiguous spans (Section 3.1).
Next Sentence Prediction The NSP task takes two sequences \((X_A, X_B)\) as input, and predicts whether \(X_B\) is the direct continuation of \(X_A\). This is implemented in BERT by first reading \(X_A\) from the corpus, and then (1) either reading \(X_B\) from the point where \(X_A\) ended, or (2) randomly sampling \(X_B\) from a different point in the corpus. The two sequences are separated by a special [SEP] token. Additionally, a special [CLS] token is added to \(X_A, X_B\) to form the input, where the target of [CLS] is whether \(X_B\) indeed follows \(X_A\) in the corpus.

In summary, BERT optimizes the MLM and the NSP objectives by masking word pieces uniformly at random in data generated by the bi-sequence sampling procedure. In the next section, we will present our modifications to the data pipeline, masking, and pre-training objectives.

3 Model

We present SpanBERT, a self-supervised pretraining method designed to better represent and predict spans of text. Our approach is inspired by BERT (Devlin et al., 2019), but deviates from its bi-text classification framework in three ways. First, we use a different random process to mask spans of tokens, rather than individual ones. We also introduce a novel auxiliary objective—the SBO—which tries to predict the entire masked span using only the representations of the tokens at the span’s boundary. Finally, SpanBERT samples a single contiguous segment of text for each training example (instead of two), and thus does not use BERT’s next sentence prediction objective, which we omit.

3.1 Span Masking

Given a sequence of tokens \(X = (x_1, x_2, \ldots, x_n)\), we select a subset of tokens \(Y \subseteq X\) by iteratively sampling spans of text until the masking budget (e.g., 15% of \(X\)) has been spent. At each iteration, we first sample a span length (number of words) from a geometric distribution \(\ell \sim \text{Geo}(p)\), which is skewed towards shorter spans. We then randomly (uniformly) select the starting point for the span to be masked. We always sample a sequence of complete words (instead of subword tokens) and the starting point must be the beginning of one word. Following preliminary trials,\(^3\) we set \(p = 0.2\), and also clip \(\ell\) at \(\ell_{\text{max}} = 10\). This yields a mean span length of mean \(\ell = 3.8\). Figure 2 shows the distribution of span mask lengths.

As in BERT, we also mask 15% of the tokens in total: replacing 80% of the masked tokens with [MASK], 10% with random tokens, and 10% with the original tokens. However, we perform this replacement at the span level and not for each token individually; that is, all the tokens in a span are replaced with [MASK] or sampled tokens.

3.2 Span Boundary Objective

Span selection models (Lee et al., 2016, 2017; He et al., 2018) typically create a fixed-length representation of a span using its boundary tokens (start and end). To support such models, we would ideally like the representations for the end of the span to summarize as much of the internal span content as possible. We do so by introducing a span boundary objective that involves predicting each token of a masked span using only the representations of the observed tokens at the boundaries (Figure 1).

Formally, we denote the output of the transformer encoder for each token in the sequence by \(x_1, \ldots, x_n\). Given a masked span of tokens \((x_s, \ldots, x_e) \in Y\), where \((s, e)\) indicates its start and end positions, we represent each token \(x_i\) in the span using the output encodings of the external boundary tokens \(x_{i-1}\) and \(x_{i+1}\), as well as the position embedding of the target token \(p_{i-s+1}\):

\[
y_i = f(x_{i-1}, x_{i+1}, p_{i-s+1})
\]
where position embeddings \(\mathbf{p}_x, \mathbf{p}_2, \ldots\) mark relative positions of the masked tokens with respect to the left boundary token \(x_{s-1}\). We implement the representation function \(f(\cdot)\) as a 2-layer feed-forward network with GeLU activations (Hendrycks and Gimpel, 2016) and layer normalization (Ba et al., 2016):

\[
\begin{align*}
\mathbf{h}_0 &= [x_{s-1}; x_{c+1}; \mathbf{p}_{t-s+1}] \\
\mathbf{h}_1 &= \text{LayerNorm}(\text{GeLU}(\mathbf{W}_1 \mathbf{h}_0)) \\
\mathbf{y}_t &= \text{LayerNorm}(\text{GeLU}(\mathbf{W}_2 \mathbf{h}_1))
\end{align*}
\]

We then use the vector representation \(\mathbf{y}_t\) to predict the token \(x_t\) and compute the cross-entropy loss exactly like the MLM objective.

SpanBERT sums the loss from both the span boundary and the regular masked language model objectives for each token \(x_t\) in the masked span \((x_s, \ldots, x_c)\), while reusing the input embedding (Press and Wolf, 2017) for the target tokens in both MLM and SBO:

\[
\mathcal{L}(x_t) = \mathcal{L}_{\text{MLM}}(x_t) + \mathcal{L}_{\text{SBO}}(x_t) = -\log P(x_t \mid x_t) - \log P(x_t \mid y_t)
\]

### 3.3 Single-Sequence Training

As described in Section 2, BERT’s examples contain two sequences of text \((X_A, X_B)\), and an objective that trains the model to predict whether they are connected (NSP). We find that this setting is almost always worse than simply using a single sequence without the NSP objective (see Section 5 for further details). We conjecture that single-sequence training is superior to bi-sequence training with NSP because (a) the model benefits from longer full-length contexts, or (b) conditioning on, often unrelated, context from another document adds noise to the masked language model. Therefore, in our approach, we remove both the NSP objective and the two-segment sampling procedure, and simply sample a single contiguous segment of up to \(n = 512\) tokens, rather than two half-segments that sum up to \(n\) tokens together.

In summary, SpanBERT pre-trains span representations by: (1) masking spans of full words using a geometric distribution based masking scheme (Section 3.1), (2) optimizing an auxiliary span-boundary objective (Section 3.2) in addition to MLM using a single-sequence data pipeline (Section 3.3). A procedural description can be found in Appendix A.

### 4 Experimental Setup

#### 4.1 Tasks

We evaluate on a comprehensive suite of tasks, including seven question answering tasks, coreference resolution, nine tasks in the GLUE benchmark (Wang et al., 2019), and relation extraction. We expect that the span selection tasks, question answering and coreference resolution, will particularly benefit from our span-based pre-training.

**Extractive Question Answering** Given a short passage of text and a question as input, the task of extractive question answering is to select a contiguous span of text in the passage as the answer.

We first evaluate on SQuAD 1.1 and 2.0 (Rajpurkar et al., 2016, 2018), which have served as major question answering benchmarks, particularly for pre-trained models (Peters et al., 2018; Devlin et al., 2019; Yang et al., 2019). We also evaluate on five more datasets from the MRQA shared task (Fisch et al., 2019): NewsQA (Trischler et al., 2017), SearchQA (Dunn et al., 2017), TriviaQA (Joshi et al., 2018), HotpotQA (Yang et al., 2018), and Natural Questions (Kwiatkowski et al., 2019). Because the MRQA shared task does not have a public test set, we split the development set in half to make new development and test sets. The datasets vary in both domain and collection methodology, making this collection a good test bed for evaluating whether our pre-trained models can generalize well across different data distributions.

Following BERT (Devlin et al., 2019), we use the same QA model architecture for all the datasets. We first convert the passage \(P = (p_1, p_2, \ldots, p_l)\) and question \(Q = (q_1, q_2, \ldots, q_r)\) into a single sequence \(X = [CLS]p_1p_2\ldots p_l[SEP]q_1q_2\ldots q_r[SEP]\), pass it to the pre-trained transformer encoder, and train two linear classifiers independently on top of it for predicting the answer span boundary (start and end). For the unanswerable questions in SQuAD 2.0, we simply set the

---

4https://github.com/mrqa/MRQA-Shared-Task-2019. MRQA changed the original datasets to unify them into the same format, e.g., all the contexts are truncated to a maximum of 800 tokens and only answerable questions are kept.
answer span to be the special token [CLS] for both training and testing.

**Coreference Resolution**

Coreference resolution is the task of clustering mentions in text which refer to the same real-world entities. We evaluate on the CoNLL-2012 shared task (Pradhan et al., 2012) for document-level coreference resolution. We use the independent version of the Joshi et al. (2019b) implementation of the higher-order coreference model (Lee et al., 2018). The document is divided into non-overlapping segments of a pre-defined length. Each segment is encoded independently by the pre-trained transformer encoder, which replaces the original LSTM-based encoder, with the original NER tags such as ‘‘[CLS]’’ [SUBJ-PER] [OBJ-LOC].

**Relation Extraction**

TACRED (Zhang et al., 2017) is a challenging relation extraction dataset. Given one sentence and two spans within it—subject and object—the task is to predict the relation between the spans from 42 pre-defined relation types, including no_relation. We follow the entity masking schema from Zhang et al. (2017) and replace the subject and object entities by their NER tags such as ‘‘[CLS]’’ [SUBJ-PER]

GLUE

The General Language Understanding Evaluation (GLUE) benchmark (Wang et al., 2019) consists of 9 sentence-level classification tasks:

- **Two sentence-level classification** tasks including CoLA (Warstadt et al., 2018) for evaluating linguistic acceptability and SST-2 (Socher et al., 2013) for sentiment classification.
- **Three sentence-pair similarity** tasks including MRPC (Dolan and Brockett, 2005), a binary paraphrasing task sentence pairs from news sources, STS-B (Cer et al., 2017), a graded similarity task for news headlines, and QQP, a binary paraphrasing task between Quora question pairs.
- **Four natural language inference** tasks including MNLI (Williams et al., 2018), QNLI (Rajpurkar et al., 2016), RTE (Dagan et al., 2005; Bar-Haim et al., 2006; Giampiccolo et al., 2007), and WNLI (Levesque et al., 2011).

Unlike question answering, coreference resolution, and relation extraction, these sentence-level tasks do not require explicit modeling of span-level semantics. However, they might still benefit from implicit span-based reasoning (e.g., the Prime Minister is the head of the government). Following previous work (Devlin et al., 2019; Radford et al., 2018), we exclude WNLI from the results to enable a fair comparison. Although recent work Liu et al. (2019a) has applied several task-specific strategies to increase performance on the individual GLUE tasks, we follow BERT’s single-task setting and only add a linear classifier on top of the [CLS] token to predict the relation type.

4.2 Implementation

We reimplemented BERT’s model and pretraining method in fairseq (Ott et al., 2019).
We used the model configuration of BERT\textsubscript{large} as in Devlin et al. (2019) and also pre-trained all our models on the same corpus: BooksCorpus and English Wikipedia using \textit{cased} Wordpiece tokens.

Compared with the original BERT implementation, the main differences in our implementation include: (a) We use different masks at each epoch while BERT samples 10 different masks for each sequence during data processing. (b) We remove all the short-sequence strategies used before (they sampled shorter sequences with a small probability 0.1; they also first pre-trained with smaller sequence length of 128 for 90\% of the steps). Instead, we always take sequences of up to 512 tokens until it reaches a document boundary. We refer readers to Liu et al. (2019b) for further discussion on these modifications and their effects.

As in BERT, the learning rate is warmed up over the first 10,000 steps to a peak value of 1e-4, and then linearly decayed. We retain $\beta$ hyper-parameters ($\beta_1 = 0.9$, $\beta_2 = 0.999$) and a decoupled weight decay (Loshchilov and Hutter, 2019) of 0.1. We also keep a dropout of 0.1 on all layers and attention weights, and a GeLU activation function (Hendrycks and Gimpel, 2016). We deviate from the optimization by running for 2.4M steps and using an epsilon of 1e-8 for AdamW (Kingma and Ba, 2015), which converges to a better set of model parameters. Our implementation uses a batch size of 256 sequences with a maximum of 512 tokens.\footnote{On the average, this is approximately 390 sequences, because some documents have fewer than 512 tokens.} For the SBO, we use 200 dimension position embeddings $p_1$, $p_2$, \ldots to mark positions relative to the left boundary token. The pre-training was done on 32 Volta V100 GPUs and took 15 days to complete. Fine-tuning is implemented based on HuggingFace’s codebase (Wolf et al., 2019) and more details are given in Appendix B.

\subsection*{4.3 Baselines}

We compare SpanBERT to three baselines:

\textbf{Google BERT} The pre-trained models released by Devlin et al. (2019).\footnote{https://github.com/google-research/bert.}

\textbf{Our BERT} Our reimplementation of BERT with improved data preprocessing and optimization (Section 4.2).

\begin{table}[h]
\centering
\begin{tabular}{llllll}
\hline
 & \textbf{SQuAD 1.1} & & \textbf{SQuAD 2.0} & & \\
 & \textbf{EM} & \textbf{F1} & & \textbf{EM} & \textbf{F1} \\
\hline
Human Perf. & 82.3 & 91.2 & & 86.8 & 89.4 \\
Google BERT & 84.3 & 91.3 & & 80.0 & 83.3 \\
Our BERT & 86.5 & 92.6 & & 82.8 & 85.9 \\
Our BERT-1seq & 87.5 & 93.3 & & 83.8 & 86.6 \\
SpanBERT & \textbf{88.8} & \textbf{94.6} & & \textbf{85.7} & \textbf{88.7} \\
\hline
\end{tabular}
\caption{Test results on SQuAD 1.1 and SQuAD 2.0.}
\end{table}

\textbf{Our BERT-1seq} Our reimplementation of BERT trained on single full-length sequences without NSP (Section 3.3).

\section*{5 Results}

We compare SpanBERT to the baselines per task, and draw conclusions based on the overall trends.

\subsection*{5.1 Per-Task Results}

\textbf{Extractive Question Answering} Table 1 shows the performance on both SQuAD 1.1 and 2.0. SpanBERT exceeds our BERT baseline by 2.0\% and 2.8\% F1, respectively (3.3\% and 5.4\% over Google BERT). In SQuAD 1.1, this result accounts for over 27\% error reduction, reaching 3.4\% F1 above human performance.

Table 2 demonstrates that this trend goes beyond SQuAD, and is consistent in every MRQA dataset. On average, we see a 2.9\% F1 improvement from our reimplementation of BERT. Although some gains are coming from single-sequence training (+1.1\%), most of the improvement stems from span masking and the span boundary objective (+1.8\%), with particularly large gains on TriviaQA (+3.2\%) and HotpotQA (+2.7\%).

\textbf{Coreference Resolution} Table 3 shows the performance on the OntoNotes coreference resolution benchmark. Our BERT reimplementation improves the Google BERT model by 1.2\% on the average F1 metric and single-sequence training brings another 0.5\% gain. Finally, SpanBERT improves considerably on top of that, achieving a new state of the art of 79.6\% F1 (previous best result is 73.0\%).

\textbf{Relation Extraction} Table 4 shows the performance on TACRED. SpanBERT exceeds our reimplementation of BERT by 3.3\% F1 and achieves close to the current state of the art (Soares et al., 2019)—Our model performs better than
<table>
<thead>
<tr>
<th>NewsQA</th>
<th>TriviaQA</th>
<th>SearchQA</th>
<th>HotpotQA</th>
<th>Natural Questions</th>
<th>Avg.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Google BERT</td>
<td>68.8</td>
<td>77.5</td>
<td>81.7</td>
<td>78.3</td>
<td>79.9</td>
</tr>
<tr>
<td>Our BERT</td>
<td>71.0</td>
<td>79.0</td>
<td>81.8</td>
<td>80.5</td>
<td>80.5</td>
</tr>
<tr>
<td>Our BERT-1seq</td>
<td>71.9</td>
<td>80.4</td>
<td>84.0</td>
<td>80.3</td>
<td>81.8</td>
</tr>
<tr>
<td>SpanBERT</td>
<td><strong>73.6</strong></td>
<td><strong>83.6</strong></td>
<td><strong>84.8</strong></td>
<td><strong>83.0</strong></td>
<td><strong>82.5</strong></td>
</tr>
</tbody>
</table>

Table 2: Performance (F1) on the five MRQA extractive question answering tasks.

<table>
<thead>
<tr>
<th>MUC</th>
<th>B³</th>
<th>CEAFφ₄</th>
</tr>
</thead>
<tbody>
<tr>
<td>P</td>
<td>R</td>
<td>F1</td>
</tr>
<tr>
<td>-----</td>
<td>----</td>
<td>-------</td>
</tr>
<tr>
<td>Prev. SotA: (Lee et al., 2018)</td>
<td>81.4</td>
<td>79.5</td>
</tr>
<tr>
<td>Google BERT</td>
<td>84.9</td>
<td>82.5</td>
</tr>
<tr>
<td>Our BERT</td>
<td>85.1</td>
<td>83.5</td>
</tr>
<tr>
<td>Our BERT-1seq</td>
<td>85.5</td>
<td>84.1</td>
</tr>
<tr>
<td>SpanBERT</td>
<td><strong>85.8</strong></td>
<td><strong>84.8</strong></td>
</tr>
</tbody>
</table>

Table 3: Performance on the OntoNotes coreference resolution benchmark. The main evaluation is the average F1 of three metrics: MUC, B³, and CEAFφ₄ on the test set.

<table>
<thead>
<tr>
<th>p</th>
<th>R</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>BERT_EM(Soares et al., 2019)</td>
<td>—</td>
<td>70.1</td>
</tr>
<tr>
<td>BERT_EM+MTB*</td>
<td>—</td>
<td><strong>71.5</strong></td>
</tr>
<tr>
<td>Google BERT</td>
<td>69.1</td>
<td>63.9</td>
</tr>
<tr>
<td>Our BERT</td>
<td>67.8</td>
<td>67.2</td>
</tr>
<tr>
<td>Our BERT-1seq</td>
<td><strong>72.4</strong></td>
<td>67.9</td>
</tr>
<tr>
<td>SpanBERT</td>
<td>70.8</td>
<td><strong>70.9</strong></td>
</tr>
</tbody>
</table>

Table 4: Test performance on the TACRED relation extraction benchmark. BERT_Elarge and BERT_EM+MTB from Soares et al. (2019) are the current state-of-the-art. *: BERT_EM+MTB incorporated an intermediate ‘matching the blanks’ pre-training on the entity-linked text based on English Wikipedia, which is not a direct comparison to ours trained only from raw text.

5.2 Overall Trends

We compared our approach to three BERT baselines on 17 benchmarks, and found that SpanBERT outperforms BERT on almost every task. In 14 tasks, SpanBERT performed better than all baselines. In two tasks (MRPC and QQP), it performed on-par in terms of accuracy with single-sequence trained BERT, but still outperformed the other baselines. In one task (SST-2), Google’s BERT baseline performed better than SpanBERT by 0.4% accuracy.

When considering the magnitude of the gains, it appears that SpanBERT is especially better at extractive question answering. In SQuAD 1.1, for example, we observe a solid gain of 2.0% F1 even though the baseline is already well above human performance. On MRQA, SpanBERT improves between 2.0% (Natural Questions) and 4.6% (TriviaQA) F1 on top of our BERT baseline.

Finally, we observe that single-sequence training works considerably better than bi-sequence training without the NSP objective substantially improves CoLA, and yields smaller (but considerable) improvements on MRPC and MNLI. The main gains from SpanBERT are in the SQuAD-based QNLI dataset (+1.3%) and in RTE (+6.9%), the latter accounting for most of the rise in SpanBERT’s GLUE average.

5.2 Overall Trends

We compared our approach to three BERT baselines on 17 benchmarks, and found that SpanBERT outperforms BERT on almost every task. In 14 tasks, SpanBERT performed better than all baselines. In two tasks (MRPC and QQP), it performed on-par in terms of accuracy with single-sequence trained BERT, but still outperformed the other baselines. In one task (SST-2), Google’s BERT baseline performed better than SpanBERT by 0.4% accuracy.

When considering the magnitude of the gains, it appears that SpanBERT is especially better at extractive question answering. In SQuAD 1.1, for example, we observe a solid gain of 2.0% F1 even though the baseline is already well above human performance. On MRQA, SpanBERT improves between 2.0% (Natural Questions) and 4.6% (TriviaQA) F1 on top of our BERT baseline.

Finally, we observe that single-sequence training works considerably better than bi-sequence training without the NSP objective substantially improves CoLA, and yields smaller (but considerable) improvements on MRPC and MNLI. The main gains from SpanBERT are in the SQuAD-based QNLI dataset (+1.3%) and in RTE (+6.9%), the latter accounting for most of the rise in SpanBERT’s GLUE average.

5.2 Overall Trends

We compared our approach to three BERT baselines on 17 benchmarks, and found that SpanBERT outperforms BERT on almost every task. In 14 tasks, SpanBERT performed better than all baselines. In two tasks (MRPC and QQP), it performed on-par in terms of accuracy with single-sequence trained BERT, but still outperformed the other baselines. In one task (SST-2), Google’s BERT baseline performed better than SpanBERT by 0.4% accuracy.

When considering the magnitude of the gains, it appears that SpanBERT is especially better at extractive question answering. In SQuAD 1.1, for example, we observe a solid gain of 2.0% F1 even though the baseline is already well above human performance. On MRQA, SpanBERT improves between 2.0% (Natural Questions) and 4.6% (TriviaQA) F1 on top of our BERT baseline.

Finally, we observe that single-sequence training works considerably better than bi-sequence training without the NSP objective substantially improves CoLA, and yields smaller (but considerable) improvements on MRPC and MNLI. The main gains from SpanBERT are in the SQuAD-based QNLI dataset (+1.3%) and in RTE (+6.9%), the latter accounting for most of the rise in SpanBERT’s GLUE average.

5.2 Overall Trends

We compared our approach to three BERT baselines on 17 benchmarks, and found that SpanBERT outperforms BERT on almost every task. In 14 tasks, SpanBERT performed better than all baselines. In two tasks (MRPC and QQP), it performed on-par in terms of accuracy with single-sequence trained BERT, but still outperformed the other baselines. In one task (SST-2), Google’s BERT baseline performed better than SpanBERT by 0.4% accuracy.

When considering the magnitude of the gains, it appears that SpanBERT is especially better at extractive question answering. In SQuAD 1.1, for example, we observe a solid gain of 2.0% F1 even though the baseline is already well above human performance. On MRQA, SpanBERT improves between 2.0% (Natural Questions) and 4.6% (TriviaQA) F1 on top of our BERT baseline.

Finally, we observe that single-sequence training works considerably better than bi-sequence training without the NSP objective substantially improves CoLA, and yields smaller (but considerable) improvements on MRPC and MNLI. The main gains from SpanBERT are in the SQuAD-based QNLI dataset (+1.3%) and in RTE (+6.9%), the latter accounting for most of the rise in SpanBERT’s GLUE average.

5.2 Overall Trends

We compared our approach to three BERT baselines on 17 benchmarks, and found that SpanBERT outperforms BERT on almost every task. In 14 tasks, SpanBERT performed better than all baselines. In two tasks (MRPC and QQP), it performed on-par in terms of accuracy with single-sequence trained BERT, but still outperformed the other baselines. In one task (SST-2), Google’s BERT baseline performed better than SpanBERT by 0.4% accuracy.

When considering the magnitude of the gains, it appears that SpanBERT is especially better at extractive question answering. In SQuAD 1.1, for example, we observe a solid gain of 2.0% F1 even though the baseline is already well above human performance. On MRQA, SpanBERT improves between 2.0% (Natural Questions) and 4.6% (TriviaQA) F1 on top of our BERT baseline.

Finally, we observe that single-sequence training works considerably better than bi-sequence training without the NSP objective substantially improves CoLA, and yields smaller (but considerable) improvements on MRPC and MNLI. The main gains from SpanBERT are in the SQuAD-based QNLI dataset (+1.3%) and in RTE (+6.9%), the latter accounting for most of the rise in SpanBERT’s GLUE average.
training with NSP with BERT’s choice of sequence lengths for a wide variety of tasks. This is surprising because BERT’s ablations showed gains from the NSP objective (Devlin et al., 2019). However, the ablation studies still involved bi-sequence data processing (i.e., the pre-training stage only controlled for the NSP objective while still sampling two half-length sequences). We hypothesize that bi-sequence training, as it is implemented in BERT (see Section 2), impedes the model from learning longer-range features, and consequently hurts performance on many downstream tasks.

6 Ablation Studies

We compare our random span masking scheme with linguistically-informed masking schemes, and find that masking random spans is a competitive and often better approach. We then study the impact of the SBO, and contrast it with BERT’s NSP objective.10

6.1 Masking Schemes

Previous work (Sun et al., 2019) has shown improvements in downstream task performance by masking linguistically informed spans during pre-training for Chinese data. We compare our random span masking scheme with masking of linguistically informed spans. Specifically, we train the following five baseline models differing only in the way tokens are masked.

Subword Tokens We sample random Wordpiece tokens, as in the original BERT.

Whole Words We sample random words, and then mask all of the subword tokens in those words. The total number of masked subtokens is around 15%.

Named Entities At 50% of the time, we sample from named entities in the text, and sample random whole words for the other 50%. The total number of masked subtokens is 15%. Specifically, we run spaCy’s named entity recognizer (Honnibal and Montani, 2017)11 on the corpus and select all the non-numerical named entity mentions as candidates.

Noun Phrases Similar to Named Entities, we sample from noun phrases at 50% of the time. The noun phrases are extracted by running spaCy’s constituency parser.

Geometric Spans We sample random spans from a geometric distribution, as in our SpanBERT (see Section 3.1).

Table 6 shows how different pre-training masking schemes affect performance on the development set of a selection of tasks. All the models are evaluated on the development sets and are based on the default BERT setup of bi-sequence training with NSP; the results are not directly comparable to the main evaluation. With the exception of coreference resolution, masking random spans is preferable to other strategies. Although linguistic masking schemes (named entities and noun phrases) are often competitive with random spans, their performance is not consistent; for instance, masking noun phrases achieves parity with random spans on NewsQA, but underperforms on TriviaQA (−1.1% F1).

On coreference resolution, we see that masking random subword tokens is preferable to any form of span masking. Nevertheless, we shall see in the following experiment that combining random span masking with the span boundary objective can improve upon this result considerably.

6.2 Auxiliary Objectives

In Section 5, we saw that bi-sequence training with the NSP objective can hurt performance on

---

10To save time and resources, we use the checkpoints at 1.2M steps for all the ablation experiments.

11https://spacy.io/.
### Table 6: The effect of replacing BERT’s original masking scheme (Subword Tokens) with different masking schemes. Results are F1 scores for QA tasks and accuracy for MNLI and QNLI on the development sets. All the models are based on bi-sequence training with NSP.

| Subword Tokens | 83.8 | 72.0 | 76.3 | 77.7 | 86.7 | 92.5 | 83.2 |
| Whole Words    | 84.3 | 72.8 | 77.1 | 76.6 | 86.3 | 92.8 | 82.9 |
| Named Entities | 84.8 | 72.7 | 78.7 | 75.6 | 86.0 | 93.1 | 83.2 |
| Noun Phrases   | 85.0 | 73.0 | 77.7 | 76.7 | 86.5 | 93.2 | 83.5 |
| Geometric Spans| **85.4** | **73.0** | **78.8** | 76.4 | **87.0** | **93.3** | 83.4 |

### Table 7: The effects of different auxiliary objectives, given MLM over random spans as the primary objective.

| Span Masking (2seq) + NSP | 85.4 | 73.0 | 78.8 | 76.4 | 87.0 | 93.3 | 83.4 |
| Span Masking (1seq)       | 86.7 | 73.4 | 80.0 | 76.3 | 87.3 | 93.8 | 83.8 |
| Span Masking (1seq) + SBO | **86.8** | **74.1** | **80.3** | **79.0** | **87.6** | **93.9** | **84.0** |

### 7 Related Work

Pre-trained contextualized word representations that can be trained from unlabeled text (Dai and Le, 2015; Melamud et al., 2016; Peters et al., 2018) have had immense impact on NLP lately, particularly as methods for initializing a large model before fine-tuning it for a specific task (Howard and Ruder, 2018; Radford et al., 2018; Devlin et al., 2019). Beyond differences in model hyperparameters and corpora, these methods mainly differ in their pre-training tasks and loss functions, with a considerable amount of contemporary literature proposing augmentations of BERT’s MLM objective.

While previous and concurrent work has looked at masking (Sun et al., 2019) or dropping (Song et al., 2019; Chan et al., 2019) multiple words from the input—particularly as pretraining for language generation tasks—SpanBERT pretrains span representations (Lee et al., 2016), which are widely used for question answering, coreference resolution, and a variety of other tasks. ERNIE (Sun et al., 2019) shows improvements on Chinese NLP tasks using phrase and named entity masking. MASS (Song et al., 2019) focuses on language generation tasks, and adopts the encoder-decoder framework to reconstruct a sentence fragment given the remaining part of the sentence. We attempt to more explicitly model spans using the SBO objective, and show that (geometrically distributed) random span masking works as well, and sometimes better than, masking linguistically-coherent spans. We evaluate on English benchmarks for question answering, relation extraction, and coreference resolution in addition to GLUE.

A different ERNIE (Zhang et al., 2019) focuses on integrating structured knowledge bases with contextualized representations with an eye on knowledge-driven tasks like entity typing and relation classification. UNILM (Dong et al., 2019) uses multiple language modeling objectives—unidirectional (both left-to-right and right-to-left), bidirectional, and sequence-to-sequence prediction—to aid generation tasks like summarization and question generation. XLM (Lample and Conneau, 2019) explores cross-lingual pre-training for multilingual tasks such as translation and cross-lingual classification. Kermit (Chan et al., 2019), an insertion based approach, fills in missing tokens...
(instead of predicting masked ones) during pre-training; they show improvements on machine translation and zero-shot question answering.

Concurrent with our work, RoBERTa (Liu et al., 2019b) presents a replication study of BERT pre-training that measures the impact of many key hyperparameters and training data size. Also concurrent, XLNet (Yang et al., 2019) combines an autoregressive loss and the Transformer-XL (Dai et al., 2019) architecture with a more than an eight-fold increase in data to achieve current state-of-the-art results on multiple benchmarks. XLNet also masks spans (of 1–5 tokens) during pre-training, but predicts them autoregressively. Our model focuses on incorporating span-based pre-training, and as a side effect, we present a stronger BERT baseline while controlling for the corpus, architecture, and the number of parameters.

Related to our SBO objective, pair2vec (Joshi et al., 2019a) encodes word-pair relations using a negative sampling-based multivariate objective during pre-training. Later, the word-pair representations are injected into the attention-layer of downstream tasks, and thus encode limited downstream context. Unlike pair2vec, our SBO objective yields “pair” (start and end tokens of spans) representations which more fully encode the context during both pre-training and finetuning, and are thus more appropriately viewed as span representations. Stern et al. (2018) focus on improving language generation speed using a block-wise parallel decoding scheme; they make predictions for multiple time steps in parallel and then back off to the longest prefix validated by a scoring model. Also related are sentence representation methods (Kiros et al., 2015; Logeswaran and Lee, 2018), which focus on predicting surrounding contexts from sentence embeddings.

8 Conclusion

We presented a new method for span-based pre-training which extends BERT by (1) masking contiguous random spans, rather than random tokens, and (2) training the span boundary representations to predict the entire content of the masked span, without relying on the individual token representations within it. Together, our pre-training process yields models that outperform all BERT baselines on a variety of tasks, and reach substantially better performance on span selection tasks in particular.

Appendices

A Pre-training Procedure

We describe our pre-training procedure as follows:

1. Divide the corpus into single contiguous blocks of up to 512 tokens.

2. At each step of pre-training:
   
   (a) Sample a batch of blocks uniformly at random.
   
   (b) Mask 15% of word pieces in each block in the batch using the span masking scheme (Section 3.1).
   
   (c) For each masked token $x_i$, optimize $L(x_i) = L_{MLM}(x_i) + L_{SBO}(x_i)$ (Section 3.2).

B Fine-tuning Hyperparameters

We apply the following fine-tuning hyperparameters to all methods, including the baselines.

Extractive Question Answering  For all the question answering tasks, we use $\text{max}_\text{seq} \text{length} = 512$ and a sliding window of size 128 if the lengths are longer than 512. We choose learning rates from $\{5e^{-6}, 1e^{-5}, 2e^{-5}, 3e^{-5}, 5e^{-5}\}$ and batch sizes from $\{16, 32\}$ and fine-tune four epochs for all the datasets.

Coreference Resolution  We divide the documents into multiple chunks of lengths up to $\text{max}_\text{seq} \text{length}$ and encode each chunk independently. We choose $\text{max}_\text{seq} \text{length}$ from $\{128, 256, 384, 512\}$, BERT learning rates from $\{1e^{-5}, 2e^{-5}\}$, task-specific learning rates from $\{1e^{-4}, 2e^{-4}, 3e^{-4}\}$, and fine-tune 20 epochs for all the datasets. We use batch size $= 1$ (one document) for all the experiments.

TACRED/GLUE  We use $\text{max}_\text{seq} \text{length} = 128$ and choose learning rates from $\{5e^{-6}, 1e^{-5}, 2e^{-5}, 3e^{-5}, 5e^{-5}\}$ and batch sizes from $\{16, 32\}$ and fine-tuning 10 epochs for all the datasets. The only exception is CoLA, where we used four epochs (following Devlin et al., 2019), because 10 epochs lead to severe overfitting.
Acknowledgments

We would like to thank Pranav Rajpurkar and Robin Jia for patiently helping us evaluate SpanBERT on SQuAD. We thank the anonymous reviewers, the action editor, and our colleagues at Facebook AI Research and the University of Washington for their insightful feedback that helped improve the paper.

References


Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee,


Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019. XLNet: Generalized autoregressive

