Attending to Long-Distance Document Context for Sequence Labeling

Matthew Jörke*
Computer Science Department
Stanford University
joerke@stanford.edu

Jon Gillick
School of Information
UC Berkeley
jongillick@berkeley.edu

Matthew Sims
School of Information
UC Berkeley
mbsims@berkeley.edu

David Bamman
School of Information
UC Berkeley
dbamman@berkeley.edu

Abstract

We present in this work a method for incorporating global context in long documents when making local decisions in sequence labeling problems like NER. Inspired by work in featurized log-linear models (Chieu and Ng, 2002; Sutton and McCallum, 2004), our model learns to attend to multiple mentions of the same word type in generating a representation for each token in context, extending that work to learning representations that can be incorporated into modern neural models. Attending to broader context at test time provides complementary information to pretraining (Gururangan et al., 2020), yields strong gains over equivalently parameterized models lacking such context, and performs best at recognizing entities with high TF-IDF scores (i.e., those that are important within a document).

1 Introduction

Many of the main datasets used in NLP are comprised of relatively short documents: English OntoNotes (Weischedel et al., 2012), for example, contains an average of 223 tokens per document, the WSJ portion of the Penn Treebank (Marcus et al., 1993) averages 501 tokens, the IMDb dataset (Maas et al., 2011) averages 272 tokens, and SQuAD 2.0 (Rajpurkar et al., 2018) contains an average of 134 tokens per passage. This focus has, in turn, led to the development of models specifically optimized for the characteristics of short documents, including a pervasive focus on the *sentence* as the atomic unit of analysis for such tasks as NER and parsing, and influencing the maximum context length of contextual language models like BERT (Devlin et al., 2019) to be limited to 512 tokens.

At the same time, however, longer documents are increasingly the objects of empirical study in areas as diverse as computational social science and the digital humanities—including novels (Piper, 2018; Underwood, 2019), scientific articles (Jurgens et al., 2018) and political manifestos (Menini et al., 2017; Denny and Spirling, 2018). These long documents present not only challenges for NLP (such as any task, like coreference resolution, whose computational complexity is superlinear in the size of the document) but opportunities as well, since the longer document context presents greater opportunity for learning better representations.

Recent work in NLP has begun exploring this link between longer documents and representation learning. First, while contextualized models (e.g. Peters et al., 2018; Devlin et al., 2019) generally consider the context of a few sentences, several recent advancements have enabled significantly longer input sequences (e.g. Dai et al., 2019; Beltagy et al., 2020; Kitaev et al., 2020; Rae et al., 2019); most, however, are either incapable of processing book-level documents or prohibitively resource-intensive for standard use.

Second, domain- and task-adaptive pretraining has proven especially effective for adapting the weights of general-purpose language models to the distribution of a particular domain or task (Gururangan et al., 2020; Han and Eisenstein, 2019; Beltagy et al., 2019; Lee et al., 2020). While longer documents are able provide more context for these models to adapt to, pretraining operates at the broad level of a domain, and is unable to exploit new con-

*Work completed while at UC Berkeley.

Findings of the Association for Computational Linguistics: EMNLP 2020, pages 3692–3704

text at *evaluation* time in unseen test documents. To highlight the value of considering document context at test time, consider the following sentence from E.M. Forster's *A Room with a View* (1908):

> "Mr. Beebe!" said the maid, and the new rector of Summer Street was shown in; he had at once started on friendly relations, owing to Lucy's praise of him in her letters from Florence.

From the context of this sentence alone, it is unclear if *Florence* refers to "a city in Italy" or "a person named Florence"; this local contextual ambiguity might lead an NER system to classify *Florence* as either a PERSON or LOCATION.

However, examining the broader document context clarifies this entity type: other mentions of *Florence* within the text more clearly indicate that it refers to the city:

- "I saw him in *Florence*," said Lucy...
- As her time at *Florence* drew to its close...
- ...two carriages stopped, half into *Florence*...

We might hypothesize, in fact, that a model that can attend to multiple mentions of a term like *Florence* in a document will perform better at recognizing important entities—those that are frequently mentioned within it and that may be infrequently seen outside of it. This fundamental idea—that multiple mentions of a term can provide shared information to help disambiguate each one—originates in featurized log-linear models that incorporate global information in making local predictions (Chieu and Ng, 2002; Sutton and McCallum, 2004; Liu et al., 2010); we extend that work here to the context of learning representations that can be incorporated into state-of-the-art neural models, explicitly learning to attend over relevant context sequences that are available only at test time, providing a complementary source of information to domain- and task-adaptive pretraining.

This work makes the following contributions:

1. We present Doc-ARC (Document-Attentive Representation of Context), an attention-based method for incorporating document context in sequence labeling tasks, and demonstrate improvements over equivalently parameterized models without document attention.
2. We evaluate Doc-ARC on three datasets containing long documents from different domains (literature, biomedical texts, and news), and present a new dataset of the full text of biomedical articles paired with labeled annotations of their abstracts in the GENIA/JNLPBA dataset (Collier and Kim, 2004).
3. We demonstrate that Doc-ARC outperforms alternative methods at recognizing important document entities (defined as those with a high TF-IDF score), identifying tangible scenarios where it would be advantageous to use.

2 Doc-ARC

The core idea behind Doc-ARC is to leverage nearby representations of the same word when generating a representation for a given token. Rather than representing *Florence* above through a contextual representation scoped only over one sentence, we represent it through a weighted combination of that token itself and other instances of *Florence* in the document. By attending over multiple instances of the same word, we are able to preserve the importance of the specific local context of a token, while also reasoning about its broader use in the rest of the document. While this model has application to a wide range of NLP tasks, we focus on the sequence labeling problem of NER.

2.1 Model Overview

Figure 1 illustrates this model for a sample text from the JNLPBA corpus. Consider a sequence $\mathbf{x} = \{x_1, \ldots, x_n\}$ with corresponding labels $\mathbf{y} = \{y_1, \ldots, y_n\}$, drawn from a document $\mathcal{D}$. Other sequences in $\mathcal{D}$ may or may not have labels and the labeled set may or may not be contiguous.

Let $e(\mathbf{x})$ be an encoding of $\mathbf{x}$ under some language model (e.g. BERT). When predicting a label, we consider both $e(\mathbf{x})$, the original encoding of the target sequence, and $c(\mathbf{x})$, an attention-weighted sum over the encodings of each $x_i \in \mathbf{x}$ as they appear in the context of $\mathcal{D}$.

Formally, let us define $\mathcal{V}(x_i)$ to be the word type (drawn from vocabulary $\mathcal{V}$) for token x_i.[1] We define $\mathcal{S}_K(x_i) = \{(\mathbf{s}_k, i_k)\}_{k=1}^{K}$ to be the K closest sequences to $\mathbf{x}$ in $\mathcal{D}$ which also contain a token of type $\mathcal{V}(x_i)$,[2] where $\mathbf{s}_k$ is the k-th closest context sequence to $\mathbf{x}$ and i_k denotes the index of $\mathcal{V}(x_i)$

[1] Here, we refer to a word *token* as an occurrence of a given word type. This is not to be confused with WordPiece tokens; we do not attend over subword representations. Throughout this work, we average over subword representations after running a sequence through BERT to convert from WordPiece representations to word-level representations.

[2] The "closest" sequences are those with the minimum absolute difference in sentence index to the target sentence.

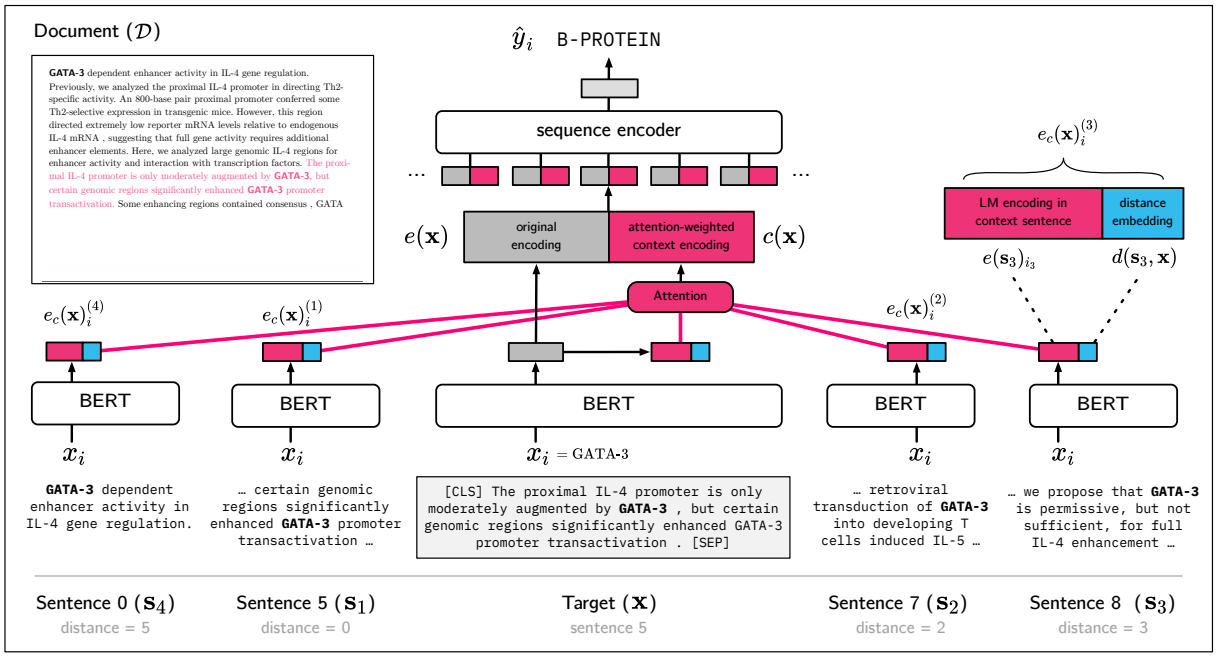

Figure 1: Overview of Doc-ARC with an example from the JNLPBA corpus, a dataset for named-entity recognition in biomedical research papers. The model attends over the representation of x_i = GATA-3 in context sentences $\mathbf{s}_k$ to product the context encoding $c(\mathbf{x})$. The BERT base model can be left trainable (dynamic Doc-ARC) for small encoders or frozen (static Doc-ARC) for large encoders.

in $\mathbf{s}_k$. For each $x_i \in \mathbf{x}$ and each $k \leq K$, our model fetches $e_c(x_i)^{(k)}$, an encoding of $\mathcal{V}(x_i)$ as it appears in the context of $\mathbf{s}_k$,

$$e_c(x_i)^{(k)} = [e(\mathbf{s}_k)_{i_k}; d(\mathbf{s}_k, \mathbf{x})], \quad (\mathbf{s}_k, i_k) \in \mathcal{S}_K(x_i) \tag{1}$$

with $d(\mathbf{s}_k, \mathbf{x})$ denoting a bucketed embedding of the distance between $\mathbf{s}_k$ and $\mathbf{x}$. We adapt our distance buckets from Lee et al. (2017).

Finally, we compute $c(x_i)$ by attending over each of the $e_c(x_i)^{(k)}$.

$$c(x_i) = \sum_{k=1}^{K} \alpha_k \cdot e_c(x_i)^{(k)} \tag{2}$$

$$\alpha_k \propto \exp\left(\mathbf{w}_{\text{attn}}{}^{\top} e_c(x_i)^{(k)}\right) \tag{3}$$

If a given word type has $K' < K$ occurences in $\mathcal{D}$, we only attend over these K' relevant instances. We allow sequences to attend over the target occurrence itself; that is, $(\mathbf{x}, i) \in \mathcal{S}_K(x_i)$.

Our model generates a prediction by passing this composite representation through a sequence encoder f_s (such as a bidirectional LSTM, GRU, or Transformer layer), and generating a distribution over labels through a softmax function:

$$\mathbf{z} = f_s([e(\mathbf{x}); c(\mathbf{x})]) \tag{4}$$

$$p(\mathbf{y} \mid \mathbf{x}, \mathcal{D}) = \text{softmax}(\mathbf{z})$$

2.2 Static and Dynamic Doc-ARC

When processing a single target sequence of length N words, our model must process $O(NK)$ context sequences. If the context representation $e_c(\mathbf{x})$ is allowed to be trainable, $O(NK)$ model activation copies are stored for each target sentence, which becomes prohibitively expensive for large encoders.

Though optimizations can be made using GPU/TPU parallelism (e.g. Raffel et al., 2019) and/or memory-efficient encoders (e.g. Kitaev et al., 2020; Lan et al., 2019), our work adopts a different focus. Instead, we consider two simple cases which encapsulate the trade-offs inherent to this method, regardless of encoder architecture:

Static. Our static variant of Doc-ARC assumes that $e(\cdot)$ is fixed throughout training. This variant is applicable when the encoder is a memory-intensive language model such as BERT. To offset the effects of freezing BERT, we pass the context representations through a trainable 1-layer context encoder f_c, which we found crucial to good performance in our experiments.

$$e_c(\mathbf{x})^{(k)} = f_c\left(e_c(x_1)^{(k)}, \ldots, e_c(x_n)^{(k)}\right) \tag{5}$$

To compute $c(\mathbf{x})$, we first gather all of the unique sequences that $\mathbf{x}$ will attend over, compute the representations of the attended sequences with a

Dataset	Documents			Sentences		Tokens	
	TRAIN	DEV	TEST	LABELED	UNLABELED	LABELED	UNLABELED
LitBank	80	10	10	8,562	617,490	210,532	13,116,998
JNLPBA	714	168	168	10,116	562,994	273,315	9,803,762
OntoNotes_{1000}	434	70	34	63,765		1,125,758	

Table 1: Dataset statistics. JNLPBA consists of many small documents (research papers), while LitBank consists of considerably fewer, long documents (novels). Both LitBank and JNLPBA have approximately the same ratio of labeled to unlabeled data (1-2%), providing complementary settings for evaluating Doc-ARC. OntoNotes_{1000} has the shortest documents on average, but each document is fully labeled.

frozen base model, and cache these representations in CPU memory.

Dynamic. Our dynamic variant assumes that $e(\cdot)$ is trainable, which necessitates a memory-efficient encoder (see §4.2). Here, each of the $O(NK)$ context sequences are processed by the encoder in a single batch, including duplicate sentences. Activations for all the sequences are held in GPU memory. We process single target sequence batches with gradient accumulation to achieve larger effective batch sizes. We do not include the context encoder f_c.

3 Datasets

We evaluate our model on three named entity recognition (NER) datasets: LitBank (Bamman et al., 2019), JNLPBA (Collier and Kim, 2004) and OntoNotes (Weischedel et al., 2012). Table 1 lists descriptive statistics for each dataset.

LitBank. The LitBank dataset (Bamman et al., 2019) is comprised of relatively long documents drawn from 100 English novels, with each document containing annotations for roughly 2,000 words. This dataset contains annotations for nested entities using six of the ACE 2005 (Walker et al., 2006) categories (PER, LOC, FAC, GPE, ORG, VEH). We convert that hierarchy into a flat structure suitable for NER by preserving only the outermost layer for any nested structure (using the same process used by JNLPBA for GENIA, described below); all annotations nested within another are removed. We use the same training, development and test splits reported in Bamman et al. (2019).

While the labeled documents in LitBank are already quite long, they represent less than 2% of the novels they are drawn from—the average full text novel in this collection is approximately 133,000 words. We draw on this broader context by treating the remainder of the novel as unlabeled document context that we can exploit.

JNLPBA. To test our performance in the biomedical domain, we use data from the JNLPBA 2004 shared task on entity recognition (Collier and Kim, 2004); this data consists of flat annotations of MEDLINE abstracts extracted from the nested entity annotations in the GENIA corpus (Kim et al., 2003), with five labels (PROTEIN, CELL LINE, CELL TYPE, DNA and RNA).

While the median document length in JNLPBA is only 245 words, these abstracts have a potentially much larger unlabeled context: the full text of the article themselves. One contribution we make in this work is constructing a new dataset by pairing the abstracts in GENIA with their full scientific articles. We do so by converting the MEDLINE identifiers encoded in the JNLPBA dataset to PubMed identifiers using mappings from the National Library of Medicine,[3] querying PubMed to retrieve the article metadata,[4] manually downloading the full-text article pdf, and OCR'ing each pdf using Abbyy FineReader. We are able to pair a total of 882 abstracts in the JNLPBA training set with their full-text articles (44.1%) and 168 abstracts in the test set (41.6%). To enable hyperparameter tuning, we divide the training set into 714 documents for training and 168 documents for development, holding out the 168 original test documents for evaluation. The average length of the unlabeled document context in this dataset is 9,337 words.

OntoNotes_{1000}. The OntoNotes 5.0 dataset (Weischedel et al., 2012) provides named entity annotations for a subset of documents, with 18 entity classes, including PERSON, LOCATION, MONEY and WORK OF ART. While the median length of documents in this collection is quite short at 277 words, we simulate a scenario of longer document context by only focusing on documents in

[3] https://ii.nlm.nih.gov/MUID_to_PMID.shtml

[4] https://pubmed.ncbi.nlm.nih.gov/

	LitBank		JNLPBA		OntoNotes_{1000}	
Base Model	**Doc-ARC**	**BERT + LSTM**	**Doc-ARC**	**BERT + LSTM**	**Doc-ARC**	**BERT + LSTM**
$\text{BERT}_{\text{BASE}}$	**75.75** (0.45)	74.22 (0.49)	71.17 (0.49)	69.28 (0.39)	**84.25** (0.41)	82.20 (0.47)
$\text{BERT}_{\text{TAPT}}$	74.28 (0.80)	72.08 (0.84)	**71.43** (0.93)	69.77 (1.22)	83.75 (0.51)	82.35 (0.56)

Table 2: Static Doc-ARC results. We report mean (SD) test F_1 scores across 5 runs. Our baseline comparison (BERT+LSTM) has a comparable number of trainable parameters, but lacks attention over context occurrences. Each Doc-ARC model was hyperparameter tuned over K, listed in the Appendix A.2.

OntoNotes that are over 1,000 words in length.

We use the same training, development, and test splits of this data used in Pradhan et al. (2013), using the BIO labels in the OntoNotes-5.0-NER-BIO repository.[5] Subsetting the data to only those documents within these partitions with over 1,000 words yields a total of 434 training documents, 70 development documents, and 34 test documents.

Preprocessing. For Doc-ARC (both static and dynamic), all labeled sequences are kept at their original length; none were longer than BERT's maximum input length (512). All unlabeled (context) sequences longer than 256 tokens are partitioned into chunks of length ≤ 256 tokens, since this limits the complexity of computing $c(\mathbf{x})$ (see §2.2). For baselines, unlabeled sequences are disregarded.

4 Experiments

We evaluate our static and dynamic Doc-ARC models on LitBank, JNLPBA, and OntoNotes_{1000}. To enable a fair comparison of the specific contribution of document-level attention, each Doc-ARC model is compared to a baseline which lacks contextual inputs and has a comparable number of trainable parameters.

4.1 Static Doc-ARC

We compute $e(\mathbf{x})$ from a frozen $\text{BERT}_{\text{BASE}}$ model, using the last four layers of BERT as a token's representation. To offset the effects of freezing BERT's weights, we let f_s and f_c be trainable bi-LSTMs. We perform hyperparameter tuning on the development set over K for each model.

Task-adaptive pretraining (TAPT). The availability of unlabeled data drawn from the same documents as a labeled dataset is exactly the scenario that task-adaptive pretraining (Gururangan et al., 2020) has demonstrated sizeable effects for. To investigate this in the context of this NER task, we pretrain $\text{BERT}_{\text{BASE}}$ on the training documents' full text (both labeled and unlabeled) for 100 epochs, yielding a $\text{BERT}_{\text{TAPT}}$ model for each dataset.

Baselines. We compare each static Doc-ARC model to a baseline with a comparable number of trainable and non-trainable parameters (frozen BERT representations input into two stacked bi-LSTMs), but lacking attention over neighboring sequences; using the notation from §2, the only input to the baseline model is $e(\mathbf{x})$, and not $c(\mathbf{x})$. We train this baseline model on the labeled set only.

Results. Table 2 lists results for Doc-ARC on all three datasets with the encoder fixed to both $\text{BERT}_{\text{BASE}}$ and $\text{BERT}_{\text{TAPT}}$. We find that Doc-ARC performs above the baselines for all trials, a difference that can reasonably be attributed to Doc-ARC's document-level contextual attention mechanism.

We find that task-adaptive pretraining is least beneficial for LitBank and OntoNotes_{1000} (perhaps due to the similarity in domain to BERT's training data of BookCorpus and Wikipedia), and most helpful for JNLPBA, which has a linguistic domain most distinct from BERT's training data.

4.2 Dynamic Doc-ARC

We compute $e(\mathbf{x})$ from the last layer of a $\text{Transformer}_{\text{TINY}}$ model (Turc et al., 2019), a compact, two-layer Transformer distilled from $\text{BERT}_{\text{BASE}}$, which we will refer to as $\text{BERT}_{\text{TINY}}$. We do not process the context representations through f_c, but maintain that f_s is a trainable bi-LSTM (see Eq. 4). For all datasets, we attend over the $K = 10$ closest sequences, which was the largest configuration that could be trained on a single GPU for all three datatsets.

Baselines. We compare each dynamic Doc-ARC to trainable $\text{BERT}_{\text{TINY}}$, as well as $\text{BERT}_{\text{TINY}}$ with one bi-LSTM attached. Analogous to the static case, the dynamic baseline has a comparable number of parameters to dynamic Doc-ARC, but lacks attention over neighboring context sequences.

[5] https://github.com/yuchenlin/OntoNotes-5.0-NER-BIO

Dataset	Doc-ARC	$\text{BERT}_{\text{TINY}}$	+ LSTM
LitBank	**64.47** (1.27)	56.17 (0.83)	60.97 (0.40)
JNLPBA	**65.26** (0.79)	56.96 (0.75)	62.08 (0.66)
OntoNotes_{1000}	**72.55** (0.76)	69.19 (0.67)	71.32 (0.42)

Table 3: Dynamic Doc-ARC results, all evaluated at $K = 10$. The BERT+LSTM baseline has a comparable number of trainable parameters, but lacks attention over context occurrences. We report mean (SD) test F_1 scores across 5 runs.

Results. We find that dynamic Doc-ARC significantly outperforms the baselines. Relative to $\text{BERT}_{\text{TINY}}$+LSTM baselines, we find that dynamic Doc-ARC gains are greater than their static counterparts for LitBank and JNLPBA. Though the dynamic models cannot match the performance of their static analogues, it is worth noting that the static variants have roughly twice as many trainable parameters. Moreover, $\text{BERT}_{\text{BASE}}$ has roughly 25 times as many parameters as $\text{BERT}_{\text{TINY}}$.

4.3 Task Fine-Tuning

To contextualize the performance of our dynamic models, we can consider results for a fully task fine-tuned $\text{BERT}_{\text{BASE}}$ and $\text{BERT}_{\text{TAPT}}$ model; as Table 4 illustrates, when given the ability to fine-tune all of its parameters to the task, performance is significantly higher than the small dynamic models, and comparable to the larger (but static) Doc-ARC models.

While a direct comparison is ill-suited given the disparity in trainable parameters in a task-tuned $\text{BERT}_{\text{BASE}}$ (11 times the number of trainable parameters as a static Doc-ARC and 25 times the number of trainable parameters as a dynamic Doc-ARC), it illustrates one direction of future work: incorporating a task-tuned contextual language model into Doc-ARC.[6] However, even with a static model with an order of magnitude fewer parameters, we find that Doc-ARC can outperform even a trainable BERT baseline for certain classes of important entities, as illustrated in the following section.

[6]Though training a dynamic Doc-ARC model with a large BERT encoder is computationally infeasible, it can be approximated via a two-step training procedure: (1) task fine-tune a BERT model to the labeled training set and (2) train a static Doc-ARC model with the encoder $e(\mathbf{x})$ initialized to these task-tuned BERT weights. In our experiments, this approach yielded only marginal improvements over standard fine-tuning scores (Table 4).

Dataset	$\text{BERT}_{\text{BASE}}$	$\text{BERT}_{\text{TAPT}}$
LitBank	**76.90** (0.61)	76.28 (0.36)
JNLPBA	70.05 (0.81)	**70.62** (0.79)
OntoNotes_{1000}	84.44 (0.18)	**85.22** (0.29)

Table 4: Fully-trainable BERT finetuning results. We report mean (SD) test F_1 scores across 5 runs.

5 Analysis

Doc-ARC was designed to (1) improve the performance of NER systems for rare, but important entities by (2) leveraging rich contextual information in long documents. In this section, we characterize the extent to which these goals were met using both quantitative and qualitative analysis.

5.1 Characterizing Important Entities

We hypothesize that Doc-ARC is most beneficial for rare entities that occur primarily within the context of a single document (such as the names of major characters in a novel). Such entities have a unique relevance only within the context of their document and are often the entities of highest importance for downstream analyses. However, these entities are particularly difficult for NER systems to classify correctly due to their rarity, unusual surface forms, and/or ambiguous meaning across documents. Given that these entities occur multiple times throughout a document and in diverse contexts, Doc-ARC should have the capacity to leverage this additional context for greater accuracy among important entities.

One means to identify important terms in a document is TF-IDF: words with high TF-IDF scores must appear frequently throughout a given document or appear *characteristically* within that document by appearing infrequently in other documents; terms with the highest scores satisfy both criteria. As Figure 2 illustrates, TF-IDF scores have a strong relationship with the presence of entity labels; words with high TF-IDF scores are more likely to be named entities across all three datasets.

Table 5 lists the three entities with the highest TF-IDF scores for each of the datasets, which appear exclusively as named entities and capture important characters (LitBank), proteins (JNLPBA), and political entities (OntoNotes).

Given that TF-IDF is a reasonable indicator for important entities, we analyze Doc-ARC's performance for high TF-IDF words in comparison to alternative models. First, we compute TF-IDF scores

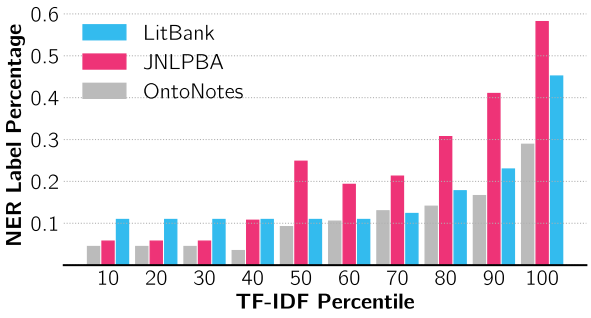

Figure 2: Among words in the labeled test set, we compute the proportion of words that appear with NER labels for each TF-IDF quantile. Across all datasets, words with a higher TF-IDF score are more likely appear as named entities.

Dataset	Top Words	Entity Type(s)
LitBank	Lucilla	Person
	Cresswell	Facility
	Marjoribanks	Person
JNLPBA	Akt-1	Protein
	Plasmin	Protein
	Siah-1	Protein
OntoNotes_{1000}	Linpien	GPE/NORP/LOC
	Dongguan	GPE/ORG
	Koreans	NORP

Table 5: Top three entities with the highest TF-IDF scores across all test sets, with entity type(s).

for all words across all documents for each dataset, using the logarithm of the term-frequency to control for variation in document length. We then restrict our vocabulary to words in the labeled test set that appear with a named entity label at least once, thereby excluding spurious high TF-IDF words (e.g. document-characteristic adjectives and adverbs). We split this vocabulary of high TF-IDF entities at the 90$^{\text{th}}$, 95$^{\text{th}}$, and 99$^{\text{th}}$ percentile and compute word-level F_1 scores within each percentile.[7]

Results. In Figure 3, we compare word-level F1 scores between our best static Doc-ARC models with a fixed $\text{BERT}_{\text{BASE}}$ input (Table 2) and a task-finetuned $\text{BERT}_{\text{BASE}}$ model (Table 4). We plot the difference in word-level F_1 scores across the entire test set and the top 10%, top 5%, and top 1% of TF-IDF entities.

While the static Doc-ARC underperforms a finetuned $\text{BERT}_{\text{BASE}}$ across all words (mirroring the results from Table 4), we find that the static Doc-ARC outperforms finetuned BERT for high TF-IDF

[7]Note that the word-level F_1 scores reported in this section differ from the entity span-level F_1 scores reported in §4, since span-level F_1 measures do not allow for word-level analysis.

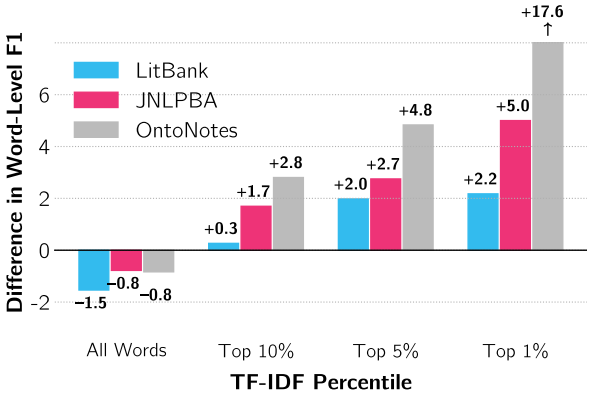

Figure 3: Difference in word level F_1 scores between static Doc-ARC and task-finetuned $\text{BERT}_{\text{BASE}}$, compared across all words and the top TF-IDF entities.

entities. Moreover, these performance gains increase with the TF-IDF threshold, indicating that Doc-ARC's performance is more sensitive to high-importance entities than a standard finetuned BERT model. These results are particularly pronounced for OntoNotes_{1000}, where Doc-ARC outperforms a finetuned BERT model by over 17 points in the top 1% of TF-IDF entities.

5.2 Characterizing Context Attention

We now turn to analyzing our model's use of attention over context occurrences. We parameterize this analysis via the attention width (K) and the attention weight (α_k).

Attention Width. The attention width (K) determines the number of context occurrences a target word can attend over. In order to better understand the impact of the attention width on our model's performance, we plot mean dev F_1 scores across three runs for several values of K in Figure 5. We find that the optimal value of K is dataset-specific and that performance does not monotonically increase with K, indicating that too much context can be detrimental. The maximum dev F_1 scores were used to determine the final hyperparameters in Table 2 (further hyperparameter details can be found the in Appendix A.2).

Attention Weight. In Figure 4, we plot the distribution attention weight as a function of the distance to the target word. Unsurprisingly, the model assigns the highest weight to the target sentence itself (distance $= 0$), including the target occurrence itself or multiple mentions of the target word within the target sentence. Though the attention weight distributions for distances greater than zero tend to

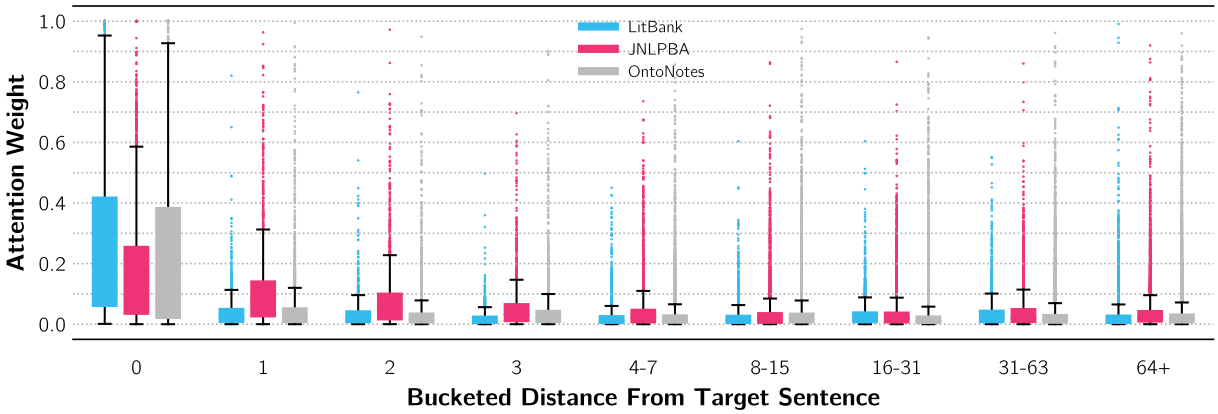

Figure 4: For each distance bucket (x-axis), we plot the distribution of attention weights assigned to context sentence in each bucket. A context sentence's distance to the target sentence is measured via absolute difference in sentence index. A distance of zero corresponds to mentions of the target word within the target sentence.

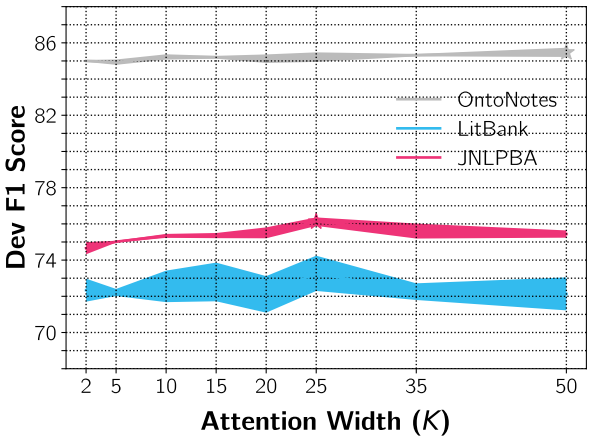

Figure 5: Mean dev F_1 with standard deviations (shaded) across three runs, for various values of K. Each model was trained with $\text{BERT}_{\text{BASE}}$.

have small medians, they have very long tails; for certain rare context sequences, Doc-ARC assigns a weight higher than the target token itself.

6 Related Work

Our work draws from several strands of related research. First, our motivation for this work is rooted in early research exploring the global scope of information across an entire document in making token-level decisions. Chieu and Ng (2002) presents one of the earliest examples of this for NER, employing features scoped over both the local token context and the broader type context in a log-linear classifier. Our use of attention in building a representation of a token that is informed by other instances of the same type is likewise influenced by work on Skip-Chain CRFs (Sutton and McCallum, 2004), which explicitly model the label dependencies between words of the same type, including for the task of NER (Liu et al., 2010).

Second, automatically retrieving relevant context has been shown to improve accuracy across a variety of NLP tasks. Searching for the k most similar context sequences to a given target has been explored for language model pretraining (Gururangan et al., 2020), training (Kaiser et al., 2017; Lample et al., 2019), and inference (Khandelwal et al., 2020); incorporating shared span representations linked through coreference has also been shown to help in multi-task learning (Luan et al., 2018). Recently, Guu et al. (2020) introduced a neural knowledge retriever for open-domain question answering, trained to retrieve the k most relevant documents during all of pretraining, finetuning, and inference. Though named-entity masking had previously shown not to improve standard BERT pretraining (Joshi et al., 2020), Guu et al. (2020) find that it significantly improves retrieval-augmented pretraining. Most prior work has computed similarity in embedding space, using either model internal representations (Khandelwal et al., 2020; Guu et al., 2020) or lightweight sentence encoders (Gururangan et al., 2019). Instead, we adopt word-type identity match as a simpler, yet effective heuristic.

Finally, self-supervised pretraining within relevant domain and/or task data has been widely shown to be beneficial for downstream task accuracy (Gururangan et al., 2020; Han and Eisenstein, 2019; Beltagy et al., 2019; Lee et al., 2020), with applications generally focused on transfer representation learning. Gururangan et al. (2020) additionally investigate human curated task-adaptive pretraining—comparable to our long-document settings—in which labeled annotations are drawn from a larger pool of unlabeled texts.

7 Conclusion

We present in this work a new method for reasoning over the context of long documents by attending over representations of identical word types when generating a representation for a token in sequence labeling tasks like NER. We show that when comparing equivalently parameterized models, incorporating attention over the entire document context leads to performance gains over models that lack that contextual mechanisms; further, the gains are asymmetric, with a substantial increase in accuracy for important entities within a document (defined as those with high TF-IDF scores). In the context of long documents, our approach presents a novel alternative to established methods such as long sequence modeling and task-adaptive pretraining.

Our work's main contribution is a computationally tractable method for attention in long documents, employing exact word match as a complexity-reducing heuristic. Though our attention mechanism is ostensibly simple, Doc-ARC 's strong performance in comparison to non-contextual baselines demonstrates both the value of the exact-match heuristic and the general utility of our framework.

This work leaves open several natural directions for future research, including incorporating document attention within a fully trainable task-tuned BERT model, and broadening the focus of attention beyond identical word types to words that bear other forms of similarity (such as similarity in subword morphology and meaning). Code and data to support this work can be found at https://github.com/mjoerke/Doc-ARC.

Acknowledgments

Many thanks to the anonymous reviewers for their feedback. The research reported in this article was supported by funding from the National Science Foundation (IIS-1813470) and the National Endowment for the Humanities (HAA-256044-17), and by resources provided by NVIDIA and Berkeley Research Computing.

References

David Bamman, Sejal Popat, and Sheng Shen. 2019. An annotated dataset of literary entities. 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 2138–2144.

Iz Beltagy, Kyle Lo, and Arman Cohan. 2019. Scibert: A pretrained language model for scientific text. 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 3606–3611.

Iz Beltagy, Matthew E. Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *arXiv:2004.05150*.

Hai Leong Chieu and Hwee Tou Ng. 2002. Named entity recognition: A maximum entropy approach using global information. In *COLING 2002: The 19th International Conference on Computational Linguistics*.

Nigel Collier and Jin-Dong Kim. 2004. Introduction to the bio-entity recognition task at JNLPBA. In *Proceedings of the International Joint Workshop on Natural Language Processing in Biomedicine and its Applications (NLPBA/BioNLP)*, pages 73–78, Geneva, Switzerland. COLING.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime G Carbonell, Quoc Le, and Ruslan Salakhutdinov. 2019. Transformer-xl: Attentive language models beyond a fixed-length context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2978–2988.

Matthew J. Denny and Arthur Spirling. 2018. Text preprocessing for unsupervised learning: Why it matters, when it misleads, and what to do about it. *Political Analysis*, 26(2):168–189.

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.

Jesse Dodge, Suchin Gururangan, Dallas Card, Roy Schwartz, and Noah A Smith. 2019. Show your work: Improved reporting of experimental results. *arXiv preprint arXiv:1909.03004*.

E.M. Forster. 1908. *A Room with a View*. Edward Arnold.

Suchin Gururangan, Tam Dang, Dallas Card, and Noah A Smith. 2019. Variational pretraining for semi-supervised text classification. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5880–5894.

Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. 2020. Don't stop pretraining: Adapt language models to domains and tasks.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. 2020. Realm: Retrieval-augmented language model pre-training. *arXiv preprint arXiv:2002.08909*.

Xiaochuang Han and Jacob Eisenstein. 2019. Unsupervised domain adaptation of contextualized embeddings for sequence labeling. 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 4229–4239.

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.

David Jurgens, Srijan Kumar, Raine Hoover, Dan McFarland, and Dan Jurafsky. 2018. Measuring the evolution of a scientific field through citation frames. *Transactions of the Association for Computational Linguistics*, 6:391–406.

Lukasz Kaiser, Ofir Nachum, Aurko Roy, and Samy Bengio. 2017. Learning to remember rare events. *ArXiv*, abs/1703.03129.

Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. 2020. Generalization through Memorization: Nearest Neighbor Language Models. In *International Conference on Learning Representations (ICLR)*.

J-D Kim, Tomoko Ohta, Yuka Tateisi, and Jun'ichi Tsujii. 2003. Genia corpus—a semantically annotated corpus for bio-textmining. *Bioinformatics*, 19(suppl_1):i180–i182.

Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. 2020. Reformer: The efficient transformer. In *International Conference on Learning Representations*.

Guillaume Lample, Alexandre Sablayrolles, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2019. Large memory layers with product keys. In *Advances in Neural Information Processing Systems*, pages 8546–8557.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. Albert: A lite bert for self-supervised learning of language representations.

Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. 2020. Biobert: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 36(4):1234–1240.

Kenton Lee, Luheng He, Mike Lewis, and Luke Zettlemoyer. 2017. End-to-end neural coreference resolution. *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*.

Jingchen Liu, Minlie Huang, and Xiaoyan Zhu. 2010. Recognizing biomedical named entities using skip-chain conditional random fields. In *Proceedings of the 2010 Workshop on Biomedical Natural Language Processing*, pages 10–18, Uppsala, Sweden. Association for Computational Linguistics.

Yi Luan, Luheng He, Mari Ostendorf, and Hannaneh Hajishirzi. 2018. Multi-task identification of entities, relations, and coreference for scientific knowledge graph construction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3219–3232, Brussels, Belgium. Association for Computational Linguistics.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 142–150, Stroudsburg, PA, USA. Association for Computational Linguistics.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Stefano Menini, Federico Nanni, Simone Paolo Ponzetto, and Sara Tonelli. 2017. Topic-based agreement and disagreement in US electoral manifestos. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2938–2944, Copenhagen, Denmark. Association for Computational Linguistics.

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.

Andrew Piper. 2018. *Enumerations*. University of Chicago Press.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Hwee Tou Ng, Anders Björkelund, Olga Uryupina, Yuchen Zhang, and Zhi Zhong. 2013. Towards robust linguistic analysis using OntoNotes. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 143–152, Sofia, Bulgaria. Association for Computational Linguistics.

Jack W Rae, Anna Potapenko, Siddhant M Jayakumar, and Timothy P Lillicrap. 2019. Compressive transformers for long-range sequence modelling. *arXiv preprint arXiv:1911.05507*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer.

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.

Charles Sutton and Andrew McCallum. 2004. Collective segmentation and labeling of distant entities in information extraction. In *ICML Workshop on Statistical Relational Learning and Its Connections to Other Fields*.

Iulia Turc, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Well-read students learn better: On the importance of pre-training compact models.

Ted Underwood. 2019. *Distant Horizons: Digital Evidence and Literary Change*. University of Chicago Press.

Christopher Walker, Stephanie Strassel, Julie Medero, and Kazuaki Maeda. 2006. ACE 2005 multilingual training corpus. LDC.

Ralph Weischedel, Sameer Pradhan, Lance Ramshaw, Jeff Kaufman, Michelle Franchini, Mohammed El-Bachouti, Nianwen Xue, Martha Palmer, Jena D. Hwang, Claire Bonial, Jinho Choi, Aous Mansouri, Maha Foster, Abdel aati Hawwary, Mitchell Marcus, Ann Taylor, Craig Greenberg, Eduard Hovy, Robert Belvin, and Ann Houston. 2012. Ontonotes release 5.0.

A Appendix

Following Dodge et al. (2019), we report our computing infrastructure (A.1), hyperparameter details (A.2), running times (A.3), and development set results (A.4) to foster reproducibile results. Our code and datasets are available at https://github.com/mjoerke/Doc-ARC.

A.1 Computing Infrastructure

Each Doc-ARC model and baseline comparison was trained on a single NVIDIA Tesla© K80 GPU with 12GB GPU memory. Task-adaptive pretraining was performed on a Google Cloud© v2-8 TPU.

A.2 Hyperparameters

Parameter	Value(s)
Epochs	30
Patience	3
Batch Size	16
Learning Rate	0.001
K (Attention Width)	$[2, 5, 10, 15, 25, 35, 50]$
H (LSTM Hidden)	256
Trainable Parameters	9.5M
Total Parameters	118M

Table 6: Static Doc-ARC hyperparameters

Static Doc-ARC. We list hyperparameters details for static Doc-ARC results (Table 2) in Table 6. We perform hyperparameter tuning over K only, choosing the optimal K via mean dev F_1 across 3 trials. The final values of K for both $\text{BERT}_{\text{BASE}}$ and $\text{BERT}_{\text{TAPT}}$ are listed in Table 7. For static Doc-ARC with $\text{BERT}_{\text{TAPT}}$, we performed tuning over $K \leq 25$ on LitBank and JNLPBA due to time constraints. Each BERT+LSTM baseline was trained with identical hyperparameters (except for K, which does not apply).

Dataset	Base Model	K
LitBank	$\text{BERT}_{\text{BASE}}$	25
	$\text{BERT}_{\text{TAPT}}$	25
JNLPBA	$\text{BERT}_{\text{BASE}}$	25
	$\text{BERT}_{\text{TAPT}}$	15
OntoNotes_{1000}	$\text{BERT}_{\text{BASE}}$	50
	$\text{BERT}_{\text{TAPT}}$	50

Table 7: Optimal K for each static Doc-ARC model

Dynamic Doc-ARC. We list hyperparameters details for dynamic Doc-ARC results (Table 3) in Table 8. Hyperparameter tuning over K was limited to $K \leq 10$ since this was the largest configuration that could be trained on a single GPU. We perform hyperparameter tuning over K only, choosing the optimal K via mean dev F_1 across 3 trials; all models had the best results for $K = 10$. Each BERT+LSTM baseline was trained with identical hyperparameters (except for K, which does not apply).

Parameter	Value(s)
Epochs	30
Patience	5
Batch Size	4
Learning Rate	0.0001
K (Attention Width)	$[2, 5, \mathbf{10}]$
H (LSTM Hidden)	128
Total Parameters	4.8M

Table 8: Dynamic Document Attention

Task Adaptive Pretraining. We perform task-adaptive pretraining on full texts within the training set for 100 epochs. Pretraining was performed using Google's BERT pretraining code[8]. Hyperparameters for pretraining are listed in Table 9.

Parameter	Value
Epochs	100
Learning Rate	2e-5
Batch Size	32
Max Sequence Length	128
Whole Word Masking	True
Masking Probability	0.15
Short sequence Probability	0
Next-sequence Prediction	True
Warmup	6%

Table 9: Task-Adaptive Pretraining (TAPT) hyperparameters

Task Finetuning. Finetuning hyperparameters for $\text{BERT}_{\text{BASE}}$ results (Table 4) and $\text{BERT}_{\text{TINY}}$ (Table 3) are listed in Table 10.

Parameter	Value
Epochs	10
Patience	3
Learning Rate	2e-5
Batch Size	16
$\text{BERT}_{\text{BASE}}$ Parameters	108M
$\text{BERT}_{\text{TINY}}$ Parameters	4.4M

Table 10: Task finetuning hyperparameters

A.3 Running Times.

For each reported result, we list average training times in Table 11. Note that task-adaptive pretraining was only run once for each dataset.

A.4 Development Set Results.

We reproduce each of the tables in the main paper with development set results. Table 12 lists dev results for Table 2, Table 13 lists dev results for Table 3, and Table 14 lists dev results for Table 4

[8] https://github.com/google-research/bert/blob/master/run_pretraining.py

Model	Dataset	Training Time (Hours:Min)
Static Doc-ARC	LitBank	26:23
	JNLPBA	24:35
	OntoNotes_{1000}	21:27
Static BERT+LSTM baseline	LitBank	00:21
	JNLPBA	00:25
	OntoNotes_{1000}	02:04
Dynamic Doc-ARC	LitBank	01:51
	JNLPBA	02:14
	OntoNotes_{1000}	07:31
Dynamic BERT+LSTM baseline	LitBank	00:07
	JNLPBA	00:09
	OntoNotes_{1000}	00:42
$\text{BERT}_{\text{BASE}}$ finetuning	LitBank	00:27
	JNLPBA	00:26
	OntoNotes_{1000}	02:30
$\text{BERT}_{\text{TINY}}$ finetuning	LitBank	00:01
	JNLPBA	00:02
	OntoNotes_{1000}	00:08
Task-adaptive pretraining	LitBank	04:48
	JNLPBA	04:27
	OntoNotes_{1000}	00:28

Table 11: Average training times for each model.

	LitBank		JNLPBA		OntoNotes_{1000}	
Base Model	**Doc-ARC**	**BERT + LSTM**	**Doc-ARC**	**BERT + LSTM**	**Doc-ARC**	**BERT + LSTM**
$\text{BERT}_{\text{BASE}}$	73.34 (0.77)	71.98 (0.93)	75.88 (0.37)	73.93 (0.30)	85.45 (0.20)	83.64 (0.10)
$\text{BERT}_{\text{TAPT}}$	71.50 (0.29)	68.83 (0.71)	77.11 (0.44)	74.93 (0.28)	85.00 (0.19)	83.33 (0.32)

Table 12: Static Doc-ARC results on the development set. We report mean (SD) F_1 scores across 5 runs.

Dataset	Doc-ARC	$\text{BERT}_{\text{TINY}}$	+ LSTM
LitBank	56.51 (0.92)	45.80 (1.35)	53.52 (0.75)
JNLPBA	72.04 (0.29)	61.03 (0.88)	68.64 (0.24)
OntoNotes_{1000}	74.11 (0.41)	70.21 (0.30)	72.98 (0.42)

Table 13: Dynamic Doc-ARC results on the development set. We report mean (SD) F_1 scores across 5 runs.

Dataset	$\text{BERT}_{\text{BASE}}$	$\text{BERT}_{\text{TAPT}}$
LitBank	73.41 (0.95)	71.49 (0.40)
JNLPBA	74.91 (0.78)	75.96 (0.41)
OntoNotes_{1000}	85.90 (0.36)	86.04 (0.20)

Table 14: BERT finetuning results on the development set. We report mean (SD) F_1 scores across 5 runs.