ADAPTING PRETRAINED LANGUAGE MODELS FOR LONG DOCUMENT CLASSIFICATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Pretrained language models (LMs) have shown excellent results in achieving human like performance on many language tasks. However, the most powerful LMs have one significant drawback: a fixed-sized input. With this constraint, these LMs are unable to utilize the full input of long documents. In this paper, we introduce a new framework to handle documents of arbitrary lengths. We investigate the addition of a recurrent mechanism to extend the input size and utilizing attention to identify the most discriminating segment of the input. We perform extensive validating experiments on patent and Arxiv datasets, both of which have long text. We demonstrate our method significantly outperforms state-of-the-art results reported in recent literature.

1 INTRODUCTION

Neural Network based Language Models (LMs) have seen a flurry of work over the past few years, with new design and implementation improvements to advance state-of-the-art performance in a variety of natural language tasks (Devlin et al., 2018; Dai et al., 2019; Radford et al., 2019; Yang et al., 2019; Liu et al., 2019). LMs are powerful tools because they process a collection of unlabeled text and are able to learn a rich embedding of natural language without supervision. This representation can be re-purposed on subsequent tasks such as classification and sentiment analysis (Korde & Mahender, 2012). Lately, this technique is essential for reaching state of the art performance, as the LM based systems are able to achieve much better results than merely working with a small, labeled dataset. The most successful LMs are able to do this unsupervised learning by using a powerful mechanism called “The Transformer” (Vaswani et al., 2017). The transformer has been shown to learn strong dependencies between its inputs, and can be stacked as many times as the hardware can handle. This ability allows LMs to take in relative large, but still *fixed*, size input.

For some of the largest LMs, the input size can reach up to four thousand tokens, but that is still a limitation insomuch that one cannot have arbitrarily long documents. On many natural language tasks, this fixed input size is sufficient. Reading comprehension tasks, for example, are often used to analyze the quality of LMs (Wang et al., 2019), contain relatively few words on average (31 for WLNI dataset), and do not have sufficiently long dependencies. Additionally, there are multiple tasks where the data is of significant length and must be truncated by these LMs (Lee & Hsiang, 2019). Long and complex text, such as a novels, often have inter-referential pieces of information that transform the meaning when taken together. Reading the final chapter of a book after all the previous ones, takes on a different meaning than reading the same text by itself.

Recurrent Neural Networks (RNNs) have been used for short text, e.g. sentiment analysis by Socher et al. (2011). However, solving the problem of arbitrarily long input requires more than a cursory glance. A first intuition may be to take the pretrained LM, place a RNN after the embedding, and simply feed multiple sequences in. Unfortunately, this causes two problems. First, RNNs are typically trained by backpropagation through time, making them prone to the problem of vanishing or exploding gradients (Pascanu et al., 2012). And while there are many techniques to solve this issue (Williams & Peng, 1990; Mujika et al., 2018), they do not get around the second problem of the significant memory requirements for gradient computation. Even when intentionally selecting a small LM, the size of the retained gradients grows quickly for multiple time steps. Further complicating the issue, the number of parameters, for transformer based networks, is quadratic in relation to the fixed input size.

In order to solve this problem, we look at the brain as a biological inspiration and how it is able to guide attention and behavioral updates. The brain processes an extraordinary amount of data, but even from moment to moment, much of that information is filtered out. What gets filtered is not arbitrary, but is directly influenced by one's objective. This can be seen when watching a video to count the number of basketball passes and missing a large gorilla in the center frame (Chabris & Simons, 2011). Even outside the moment to moment, changes in an individual's behavior are guided by their value structure, self-selected or otherwise. This value structure imposes a framework on how to determine the significance of events (Peterson, 1999). An event that may have been insignificant in the past can go on to take a new meaning once a new value has been gained or an old one has changed Laudet et al. (2006). This increase in valence may cause behavioral change, a reorientation of goals, and a shift in the interpretation of experiences.

Unfortunately, when it comes to modeling this selective learning in Artificial Intelligence systems, typical gradient based methods fall short. For example, a typical neural network model will update all its parameters based on all of the inputs to minimize an objective function. While this is desired for many applications, full input based learning can cause issues for others (Pascanu et al., 2012). Ke et al. (2018) attempt to solve this issue via selective attentive backtracking, but they focus on remembering long-term dependencies and not on utilizing the full context of the input. In this work, we introduce a method that can utilize the entire input, then perform objective-based filtering during learning in the domain extremely complex data that is language.

The two areas we focus on specifically are scientific papers and patents for classification. Scientific papers are important area of investigation as many get uploaded to the internet every day, and automatic categorization could be a great use. For example, a statistics paper may not be categorized as machine learning by its authors, but could be of interest to the machine learning community. Additionally, Tshitoyan et al. (2019) have shown LMs can be combined with these papers to capture material science concepts and can recommend materials for functional applications several years before their discovery. Patent reading, for the purpose of classification, is a typical activity for lawyers trying to find relevant documents. With the number of patents filed as increased nearly every year since 2003 (WIPO, 2018), using an automated system to perform classification has been a continuously growing area of interest (Trappey et al., 2006).

In this work, we use an attention mechanism to discover the significant portions of text for which to perform backpropagation over a pretrained LM. We find this attention is vital because the most significant portion of an input sequence may occur anywhere throughout a document. While the datasets we study often start with highly discriminative features (titles and abstracts), we conduct experiments to show our attention mechanism can find the important parts of text even when it does not occur in the first segment. For more general texts, the discriminative features may be scattered throughout the text, or have long range dependencies. And while it is the case that using the gradients from only the first input segment performs well, often better than just using a baseline of the original language model with the input truncated to fit the max size, using the attention mechanism consistently improves performance and achieves the best out of our experimented language models. Therefore, our contributions are as follows:

1. We introduce a new framework for performing inference over arbitrary length documents.
2. We perform extensive validating experiments to contextualize our method within related work, showing our method to significantly outperform alternative methods.

2 RELATED WORKS

Language model pre-training has become a popular method for tackling many natural language understanding tasks. Bidirectional Encoder Representations from Transformers (BERT) (Devlin et al., 2018) and Generative Pre-Training (GPT) (Radford et al., 2018) are two well-known language model pre-training methods that we utilize in this work. BERT is trained by selectively conditioning on part of the input sequence, masking the rest, and attempting to predict the masked tokens; and is also trained by doing a next sentence prediction task. The token used for the next sentence prediction task is reused for classification after the LM has been trained. GPT trains by conditioning on a set of input text and tries to predict the next word in the sequence. By training on Wikipedia and books, GPT is able to generate novel sequences of text. It is also able to perform classification by adding a

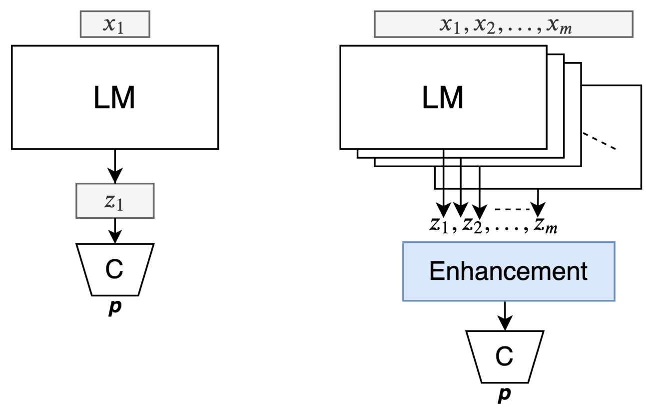


Figure 1: **Left**: The base language model (Base BERT/GPT) for classification. **Right**: The framework of language model enhancements for classification.

classifying token at the end of the text, encoding the token into a latent representation, and using it as input to new linear classifier set specifically for the task. These language model pre-training methods become state-of-art on natural language processing benchmarks such as GLUE (Wang et al., 2019) and SQUAD (Rajpurkar et al., 2016), achieving close to human-level performance.

As both BERT and GPT are based on Transformers, their computational and storage costs scale quadratically with the input sequence length. This limits their application to mostly relatively short pieces of texts. To the best of our knowledge, we are not aware of any works applying these pre-trained language models for long document classification. So in the following we discuss a few other deep learning-based text classification approaches, with special attention to the classification of scientific papers and patents.

Dai & Le (2015) consider pre-training recurrent neural networks with large corpus of texts, and show improved performance on several text classification tasks. Kim (2014) introduced convolutional neural networks for text/sentence classification. Yang et al. (2016) introduced a hierarchical attention mechanism for document classification that attends to interesting sentences and words in a document. However, the length of documents considered in later two works are relatively short, with the corpus consisting of mostly individual sentences or online reviews.

For patent classification, Li et al. (2018) present a deep learning algorithm for patent classification based on CNN and continuous skip-gram embedding called Deep Patent. They were the first to apply deep learning to large scale real-world patent classification. Lee & Hsiang (2019) used the pretrained BERT model to classify patents at the section and subclass level, only taking the title and abstract, or the first claim, as input.

For scientific paper classification, He et al. (2019) introduce a relatively large scientific paper dataset and perform classification by using a multi-network approach. They introduce a reinforcement-based RNN Attention Learning scheme. This is used to select short text sequences to be parsed by a CNN, and the representation learned by the CNN is fed back into the RNN for subsequent text selections and eventual classification.

Very recently, Cohan et al. (2019) construct a joint sentence representation that allows BERT Transformer layers to directly utilize contextual information from all words in all sentences. However, their task is classification at the sentence level for a single input.

3 Methods

Our approach for classifying long documents is to divide the long document into a sequence of segments, each of which is short enough to be processed by a pretrained language model. The information, from the learned language model representation of each segment, is utilized to produce a classifier (see Figure 1(right)). This strategy of information combination is very simple, and our main contribution in this work is to investigate the effectiveness of different combination strategies.

Let $\boldsymbol{x} = (x_1, x_2, \ldots, x_m)$ be a document, where x_i is a fixed-length sequence of tokens (segment), and m the number of segments in the document. Let $y \in \mathbb{Y}$ be the respective labels in a k-class classification problem. We use $\boldsymbol{z}_i = \mathrm{LM}(x_i) \in \mathbb{R}^d$ to denote the d-dimensional latent representa-

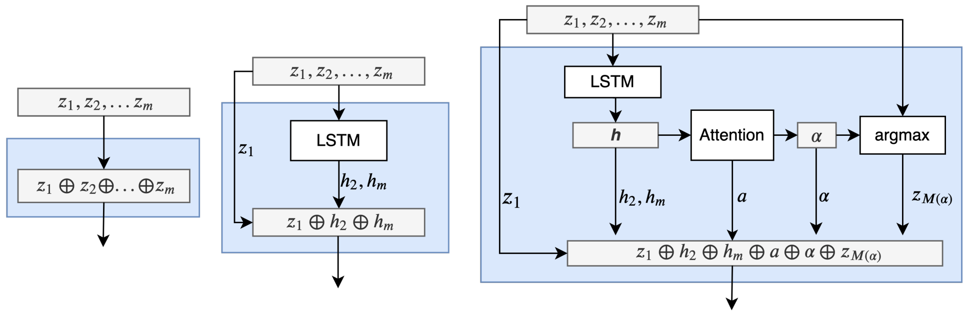


Figure 2: Three enhancements for base LM. **Left**: The concatenation model (Cat-BERT/GPT). **Center**: The RNN based model (RNN-BERT/GPT). **Right**: The full attention model (ATT-BERT/GPT).

tion, of the segment x_i, for classification (e.g. the representation of the CLS token in BERT) from a selected language model LM. Let $C_{\boldsymbol{W},\boldsymbol{b}}(v) = \sigma(\boldsymbol{W}\boldsymbol{v} + \boldsymbol{b})$ be a linear classifier followed by the softmax function, and let $\boldsymbol{p}$ the vector of probabilities of $\boldsymbol{x}$ being assigned to each class.

Base Language Model (Base LM) In the usual application of deep language models such as BERT and GPT for text classification, the input text is truncated at a fixed length (256, 512, etc) due to limits in the size of the model. This corresponds to our basic model:

$$\boldsymbol{p} = C_{\boldsymbol{W},\boldsymbol{b}}(\mathrm{LM}(x_1)), \tag{1}$$

where $\boldsymbol{W} \in \mathbb{R}^{k \times d}$, $\boldsymbol{b} \in \mathbb{R}^k$, and we assume the segment length of x_1 equals to the input size limit of the language model. This model is depicted in Figure 1(left).

Next we describe the three enhancements to the base LM, shown in Figure 2.

Concatenated Language Model (Cat-LM) The first enhancement is a natural extension to improve the basic model by including information from more segments $x_2, \ldots, x_m$. A very simple way to do this is to concatenate the representation $\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_m$ before the classification layer. This leads to the model:

$$\boldsymbol{p} = C_{\boldsymbol{W},\boldsymbol{b}}(\mathrm{LM}(x_1) \oplus \mathrm{LM}(x_2) \cdots \oplus \mathrm{LM}(x_m)), \tag{2}$$

where $\boldsymbol{W} \in \mathbb{R}^{k \times md}$, $\boldsymbol{b} \in \mathbb{R}^k$. This model is difficult to perform backpropagation directly because we cannot hold m copies of the LM parameters in memory at the same time. We solve this problem by stopping the backpropgation paths of some of the segments, namely $\boldsymbol{z}_2, \ldots, \boldsymbol{z}_m$. Section 3.1 discusses this approximation.

RNN-augmented Language Model (RNN-LM) The third model we want to consider is one that summarizes the information from $\boldsymbol{z}_1, \ldots, \boldsymbol{z}_m$ using a bidirectional LSTM. Let $(\boldsymbol{h}_1, \boldsymbol{h}_2, \ldots, \boldsymbol{h}_m) = \mathrm{biLSTM}(\mathrm{z}_1, \ldots, \mathrm{z}_\mathrm{m})$ be the q dimensional hidden state representations from a bidirectional LSTM, where $\boldsymbol{h}_i \in \mathbb{R}^q$. The biLSTM-based model can be written as:

$$\boldsymbol{p} = C_{\boldsymbol{W},\boldsymbol{b}}(\mathrm{LM}(x_1) \oplus \boldsymbol{h}_2 \oplus \boldsymbol{h}_m), \tag{3}$$

where $\boldsymbol{W} \in \mathbb{R}^{k \times (d+2q)}$, $\boldsymbol{b} \in \mathbb{R}^k$. For this model we also stop the gradient computation at $\boldsymbol{z}_2, \ldots, \boldsymbol{z}_m$, and do not backpropagate beyond the LSTM parameters.

Attention-based Language Model (ATT-LM) For our attention based model, we use the same attention mechanism as described in Yang et al. (2016). We define our attention variables as follows:

$$\begin{aligned} \boldsymbol{u}_i &= \tanh(\boldsymbol{W}_s \boldsymbol{h}_i + \boldsymbol{b}_s), i = 1, \ldots, m \\ \alpha_i &= \sigma(\boldsymbol{u}_i^T \boldsymbol{u}_s) \\ \boldsymbol{a} &= \sum_{i=1}^{m} \alpha_i \boldsymbol{h}_i, \end{aligned} \tag{4}$$

where $\boldsymbol{W}_s \in \mathbb{R}^{2q \times q}$, $\boldsymbol{b}_s \in \mathbb{R}^q$, $\boldsymbol{u}_s \in \mathbb{R}^q$ are learned attention parameters for attention over segments. Let $M(\boldsymbol{\alpha}) = \arg\max_{1 \le i \le m} \alpha_i$ be the index of the segment that gives the highest attention weight.

And that gives us our equation for the attention-based model by concatenating a set of relevant features:

$$\boldsymbol{p} = C_{\boldsymbol{W},\boldsymbol{b}}(\mathrm{LM}(x_1) \oplus \boldsymbol{h}_2 \oplus \boldsymbol{h}_m \oplus \boldsymbol{a} \oplus \boldsymbol{\alpha} \oplus \mathrm{LM}(x_{M(\boldsymbol{\alpha})})), \tag{5}$$

where $\boldsymbol{W} \in \mathbb{R}^{k \times (2d+3q+m)}, \boldsymbol{b} \in \mathbb{R}^k$ are the parameters of the linear classifier. For this model, we stop the gradient computations paths for all $\boldsymbol{z}_i$'s apart from $\boldsymbol{z}_1$ and the selected $\boldsymbol{z}_{M(\boldsymbol{\alpha})}$.

For all four methods, we use the negative log likelihood as the loss function:

$$L = -\sum_{i=1}^{n} \log p_{y_i}, \tag{6}$$

where n is the number of training documents.

3.1 Parameter Updates

We run mini-batch stochastic gradient descent for parameter updates. We treat the parameters of the language models and the other parameters (weight matrix $\boldsymbol{W}$ for classification, LSTM parameters, etc) differently. We perform full gradient computation on non-LM parameters, and only approximate gradient computation for the LM's parameters by stopping backpropagation on selected segments (see the model description above and the Appendix Section A).

4 Experiments

Implementation Details. We use PyTorch (Paszke et al., 2017) to conduct all our experiments and the HuggingFace (2019) implementation of both the Base-BERT (110M parameters) pretrained model and the GPT pretrained model of similar size (117M parameters). Due to computational constraints, we do not use BERT-Large or larger GPT models. We use BERT's lower-case tokenizer and the GPT's tokenizer with an added classification token. Both models take a fixed input size of 256 tokens, contain 12 transformer blocks, and have a hidden size of 768 neurons. We use an LSTM (Hochreiter & Schmidhuber, 1997) for our recurrent neural network in the recurrent and attention based methods. We apply dropout (Srivastava et al., 2014) ($p = .1$) before the final linear layer.

Unless otherwise noted, we use a learning rate of $2e^{-5}$, scheduled ADAM optimizer (Kingma & Ba, 2015), train for 3 epochs over each training set, use a training mini-batch size of 32 documents, and set all other hyper parameters to their default values. For ease of processing, we use $m = 8$ segments for the patent datasets as patents have a relatively constrained length and 8 allows for minimal padding to be used. We also use $m = 16$ segments for the Arxiv dataset as one document typically contains 6k words, many of which are removed as non-meaningful or are intentionally truncated as part of the bibliography.

Next, we describe the different datasets we use in our experiments.

Patents. Patents can be broken down into multiple levels of resolution according to the International Patent Classification System (IPC): *Section, Subsection, Class, Subclass*. The most broad category, *Section*, has eight labels (A-H). For example, *Section A* is concerned with *Human Necessities*, while *Section H* is concerned with *Electricity*. We also perform classification experiments on a more detailed level of categorization *Subclass*, with 638 different labels. Patents were gathered from the Google Patents Public Dataset via SQL queries.

We gathered all documents from the United States Patent Office (USPTO) from 2006-2014 for our training set, and use patents from 2015 as our test set. We have 1917334 training and 296724 testing documents, where 15172 and 1835 documents were respectively skipped for missing abstracts.

The text of a patent is composed of different parts: title, abstract, and a list of claims. For our purposes, we consider one patent to be first the title, then the abstract, and followed by each claim in order– claim 1, claim 2, ... until the last claim.

(Inverted) Wireless Patents. We also selected a subset of patent data to perform additional experiments. We selected wireless (H04) due to its large number of training and test examples (the second

Method \ Dataset	Arxiv-4	Arxiv-11	section	subclass	wireless	inverted wireless
Li et al. (2018) Deep Patent	-	-	-	<43	-	-
Lee & Hsiang (2019) PatentBert	-	-	80.98	66.80	-	-
He et al. (2019) Local Word Glimpses	94.18	80.47	-	-	-	-
Base-GPT	96.59	84.62	83.32	67.29	89.82	87.69
Base-BERT	97.06	87.42	83.85	68.31	90.21	87.72
Cat-GPT	96.82	80.03	83.43	66.17	89.34	88.80
Cat-BERT	97.06	87.34	83.99	68.34	90.64	89.39
RNN-GPT	96.98	85.31	83.52	67.72	90.16	89.19
RNN-BERT	97.62	87.72	83.99	68.72	90.51	89.41
ATT-GPT	97.62	85.94	83.66	68.13	90.31	90.08
ATT-BERT	**97.70**	**87.96**	**84.13**	**69.01**	**90.69**	**90.25**

Table 1: Micro F1 results on our datasets.

most of all the *Class* data). Due to its imbalance in subclasses, the *Class* with the most examples (computing) was excluded (dominated by one label with over 75% examples). We use the wireless *Class* to also construct an inverted patent dataset, where a single patent starts with its last claim, up to the first claim in the reverse order, then the abstract, and lastly the title. It is commonly believed that the abstract and the first claim is the most useful in classifying a patent. We create this dataset to present information in reverse order of relevance to test models that bias towards the beginning of documents (e.g., models that truncate beyond a fixed number of tokens). After processing, the wireless *Class* contains 250,982 training and 42,892 testing documents, where 15,172 and 97 documents were skipped, respectively, for missing abstracts.

Arxiv papers. We use the dataset provided by He et al. (2019). It consists of 33388 papers downloaded from the scientific article hosting website Arxiv, from 11 different categories. The least occurring category is “math.AC” with 2885 documents, and the most occuring is ”cs.DS” with 4136 documents. We call this dataset Arxiv-11. They also provide a subset of the data using four categories and 12,195 documents, which we refer to as Arxiv-4. All downloaded pdf documents were converted to txt files, with no document less than 1,000 words. We randomly sample 90% for training and use the remaining 10% for test.

5 RESULTS

Patents and Arxiv Datasets. We report our main results in Table 1, where the numbers are micro-F1 scores. We make three observations. First we compare our methods to the previous work using DeepPatent (Li et al., 2018), PatentBert (Lee & Hsiang, 2019) and Local Word Glimpses (He et al., 2019). Table 1 shows Base LMs perform well, but the usage of attention is superior in all cases. Second, we compare Base LMs against the three enhancements across 4 variants of the patent dataset and 2 variants of the Arxiv dataset. We note that RNN-based and attention-based LMs show consistent improvement over base LMs. Among the three enhancements, attention-based (highlighted in the last line) is strongest of all. It is also interesting to note that BERT-based models consistently outperform GPT-based models, and this is likely due to the use of bidirectional contexts in BERT.

Inverted Patents. As shown in recent work (Lee & Hsiang, 2019; Li et al., 2018), only the abstract or first claim on a patent is needed for high performance. To analyze the effect of attention discovering the location of discriminating content, we invert the structure of the patent – reversing the order of claims with last claim (least important) at the beginning and first claim (most important) at the end, this is followed by the abstract, then lastly the title. Comparing the results from the second last column (wireless) and last column (inverted wireless), we see that Base-LMs which only take into account of first 256 tokens, suffer from a drop of F1 scores of more than 2.0 on inverted wireless.

Method \ Param	3 epochs 2e-5 lr	3 epochs 3e-5 lr	3 epochs 4e-5 lr	3 epochs 5e-5 lr	8 epochs 2e-5 lr	8 epochs 5e-5 lr
Base GPT	96.19	96.59	96.43	96.19	96.27	96.35
Base BERT	96.90	97.06	96.43	96.27	96.58	97.06
Cat-GPT	93.88	94.28	95.63	95.31	96.74	96.82
Cat-BERT	96.82	96.98	96.27	96.66	96.90	97.06
RNN-GPT	96.03	96.82	96.59	96.98	96.90	96.74
RNN-BERT	97.62	96.74	97.06	96.74	96.51	96.90
ATT-GPT	95.79	97.38	97.62	97.62	96.43	97.14
ATT-BERT	97.06	**97.70**	96.27	95.87	96.98	97.14

Table 2: The effect of changing the learning rate, and epoch, hyper-parameters for all models on the Arxiv-4 dataset.

Cat-LMs and RNN-LMs reduce the gap in F1 scores to about 1.0, while the attention-based models ATT-LMs perform the best, with the gap between wireless and inverted wireless less than 0.5.

Exploration of Hyper-parameters. Next, we investigate the effects of different hyper-parameters on the various models using a small set of training data. For this, we use the Arxiv-4 dataset. While the attention based models seem to do well, table 2 demonstrates the unpredictable nature of using different learning rates along with different training epochs. This behavior aligns with the claims of Devlin et al. (2018), who also found fine-tuning was sometimes unstable on small datasets.

Training and Evaluation Time. Lastly, we compare the training time and evaluation speed of our models on the Arxiv-11 dataset. As shown in Table 3, due to the multiple forward passes, all of the enhanced variants of the LMs require nearly 3-4x training time and over 2x to evaluate. Considering there are substantially more operations required to process all the input text, this slow down is better than expected. Furthermore, the difference in training time between the attention-based models and non-attention-based is surprising, given the fact that the LM's parameters must be updated with another input's gradients. This points towards further gradient computations being feasible for architectures, and hardware, that can handle the additional required memory.

5.1 Ablation Experiment

In order to understand the effects of the attention, we introduce an ablated model that allows us to investigate how attention influences and guides back-propagation. This model is similar to ATT-BERT, but we remove $LM(x_1)$, h_2, or h_m from the classifier. Therefore it can be written as:

$$\boldsymbol{p} = C_{W,b}(a \oplus \boldsymbol{\alpha} \oplus \mathrm{LM}(x_{M(\alpha)})), \tag{7}$$

with $W \in \mathbb{R}^{k\times(q+m+d)}$, $b \in \mathbb{R}^k$ being the parameters of the linear classifier. Using this model, we carry out two experiments.

Shuffling Experiment. First, we perform a shuffled input experiment to investigate the effect of using attention to select which segments to retain gradients for. We use all the same setup as the experiments for the Arxiv-11 column in Table 1, except for each training iteration, we randomly permute each segment x_1 between all mini-batch examples. This means methods that use x_1 will no longer be able to rely upon those gradients to give an informative update to the model's parameters.

	Base-BERT	Cat-BERT	RNN-BERT	ATT-BERT
BERT Training time	1.000	2.711	3.079	3.610
GPT Training time	1.000	3.088	3.551	3.558
BERT Evaluation time	1.000	2.712	2.940	3.147
GPT Evaluation time	1.000	2.520	2.689	2.839

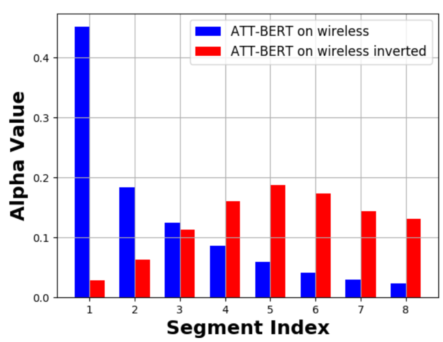
Table 3: An analysis of the different model run times as a factor of the baseline method.

	Base BERT	Cat-BERT	RNN-BERT	ATT-BERT	Ablated ATT-BERT
x_1 gradients	11.92	12.96	11.92	81.05	81.11
No x_1 gradients	11.92	46.72	70.59	81.08	81.11

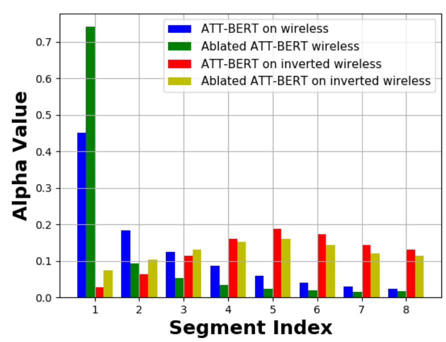
Table 4: BERT model results when shuffling every x_1 in the Arxiv-11 dataset.

Table 4 shows how the non-attention based methods guess the maximum occurring class when gradients are updated using x_1. And it also shows how these methods perform poorly without gradients to update the language model parameters. On the other hand, the attention-based methods are able to perform well with a loss of information, and ATT-BERT is relatively unaffected by the loss of gradients from x_1.

Attention $\boldsymbol{\alpha}$ Comparisons. Second, we measure and compare the α values on the wireless and inverted wireless patent dataset. We average the α value for each of the eight input segments over the entire test set. Figure 3(**a**) comparatively shows the alpha values of our attention based model on the wireless and inverted wireless dataset; this clearly demonstrates the attention mechanism is able to accurately pick up on the important sections of text. Figure 3(**b**) shows the comparison between ATT-BERT and the ablated model on both the wireless and inverted wireless datasets. The α values for both models on the inverted dataset seem relatively similar, with the ablated model placing heavier emphasis on the first few segments. However, the effect of including $x_1 \in \mathcal{X}$ can clearly been seen on the wireless dataset, where the α_1 value is over 0.7. This means enabling backpropagation to occur for input x_1 has a positive effect for ATT-BERT and, moreover, that the title and abstract in a patent are of high discriminative importance.



(a)



(b)

Figure 3: **(a)** An analysis of the effect on attention $\boldsymbol{\alpha}$ values between the original wireless dataset, and the inverted wireless dataset. **(b)** A comparison of $\boldsymbol{\alpha}$ values with the ablated model.

6 CONCLUSIONS

In this work, we achieve state of the art results on multiple long document classification tasks by utilizing pretrained language models with attention based enhancements. With language modeling continuing to see improvements every month, we show how to different models can be integrated into our system. We performed numerous experiments to clearly demonstrate the value added by using attention to learn from important segments. Furthermore, we showed that the additional gradient computation as a result of attention is marginal when compared to the consistent improvement of results; and we analyzed the effects of the attention mechanism through the loss of input information via both shuffling and a carefully constructed dataset augmentation.

REFERENCES

Christopher F Chabris and Daniel Simons. *The invisible gorilla: And other ways our intuitions deceive us*. Harmony, 2011.

Arman Cohan, Iz Beltagy, Daniel King, Bhavana Dalvi, and Daniel S. Weld. Pretrained language models for sequential sentence classification. *CoRR*, abs/1909.04054, 2019. URL `https://arxiv.org/pdf/1909.04054.pdf`.

Andrew M Dai and Quoc V Le. Semi-supervised sequence learning. In *Advances in Neural Information Processing Systems*, pp. 3079–3087, 2015.

Zihang Dai, Zhilin Yang, Yiming Yang, William W Cohen, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov. Transformer-XL: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*, 2019.

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

Jun He, Liqun Wang, Liu Liu, Jiao Feng, and Hao Wu. Long document classification from local word glimpses via recurrent attention learning. *IEEE Access*, 7:40707–40718, 2019. URL `https://doi.org/10.1109/ACCESS.2019.2907992`.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997.

HuggingFace. Pytorch-transformers. `https://github.com/huggingface/pytorch-transformers`, 2019.

Nan Rosemary Ke, Anirudh Goyal GOYAL, Olexa Bilaniuk, Jonathan Binas, Michael C Mozer, Chris Pal, and Yoshua Bengio. Sparse attentive backtracking: Temporal credit assignment through reminding. In *Advances in Neural Information Processing Systems*, pp. 7640–7651, 2018.

Yoon Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR*, 2015.

Vandana Korde and C Namrata Mahender. Text classification and classifiers: A survey. *International Journal of Artificial Intelligence & Applications*, 3(2):85, 2012.

Alexandre B Laudet, Keith Morgen, and William L White. The role of social supports, spirituality, religiousness, life meaning and affiliation with 12-step fellowships in quality of life satisfaction among individuals in recovery from alcohol and drug problems. *Alcoholism treatment quarterly*, 24(1-2):33–73, 2006.

Jieh-Sheng Lee and Jieh Hsiang. PatentBERT: Patent classification with fine-tuning a pre-trained BERT model. *CoRR*, abs/1906.02124, 2019. URL `http://arxiv.org/abs/1906.02124`.

Shaobo Li, Jie Hu, Yuxin Cui, and Jianjun Hu. DeepPatent: patent classification with convolutional neural networks and word embedding. *Scientometrics*, 117(2):721–744, 2018. doi: 10.1007/s11192-018-2905-5.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

Asier Mujika, Florian Meier, and Angelika Steger. Approximating real-time recurrent learning with random kronecker factors. In *Advances in Neural Information Processing Systems*, pp. 6594–6603, 2018.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. Understanding the exploding gradient problem. *CoRR, abs/1211.5063*, 2, 2012.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. In *NIPS-W*, 2017.

J.B. Peterson. *Maps of Meaning: The Architecture of Belief.* Routledge, 1999. ISBN 9780415922227.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. *URL https://s3-us-west-2. amazonaws. com/openai-assets/researchcovers/languageunsupervised/language understanding paper. pdf*, 2018.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019. URL `https://d4mucfpksywv.cloudfront.net/better-language-models/language-models.pdf`.

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

Richard Socher, Jeffrey Pennington, Eric H Huang, Andrew Y Ng, and Christopher D Manning. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pp. 151–161. Association for Computational Linguistics, 2011.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

Amy JC Trappey, Fu-Chiang Hsu, Charles V Trappey, and Chia-I Lin. Development of a patent document classification and search platform using a back-propagation network. *Expert Systems with Applications*, 31(4):755–765, 2006.

Vahe Tshitoyan, John Dagdelen, Leigh Weston, Alexander Dunn, Ziqin Rong, Olga Kononova, Kristin A Persson, Gerbrand Ceder, and Anubhav Jain. Unsupervised word embeddings capture latent knowledge from materials science literature. *Nature*, 571(7763):95, 2019.

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

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *7th International Conference on Learning Representations, ICLR*, 2019.

Ronald J Williams and Jing Peng. An efficient gradient-based algorithm for on-line training of recurrent network trajectories. *Neural computation*, 2(4):490–501, 1990.

WIPO. World intellectual property indicators. *Geneva: World Intellectual Property Organization*, 2018.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. *arXiv preprint arXiv:1906.08237*, 2019.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alexander J. Smola, and Eduard H. Hovy. Hierarchical attention networks for document classification. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2016.

A GRADIENT COMPUTATION FOR LANGUAGE MODEL PARAMETERS

Let us denote our language model representation be $\boldsymbol{z}_i = \mathrm{LM}(x_i) = f_{\boldsymbol{\theta}}(x_i)$, where $\boldsymbol{\theta}$ are the language model parameters. Let $h_{\boldsymbol{\phi}}$ be the function we compute on top of the LM representations, e.g., the classifier, LSTM, etc. The parameter $\boldsymbol{\phi}$ can contain the classification weights W and the LSTM weights. The models considered in this paper can be written as:

$$\boldsymbol{p} = h_{\boldsymbol{\phi}}(f_{\boldsymbol{\theta}}(x_1), \dots, f_{\boldsymbol{\theta}}(x_m))$$

Coupled with the loss function l (log loss) and the target label y, we have

$$l(\boldsymbol{p}, y) = l(h_{\boldsymbol{\phi}}(f_{\boldsymbol{\theta}}(x_1), \dots, f_{\boldsymbol{\theta}}(x_m)), y)$$

Computing the gradient over the LM parameters $\boldsymbol{\theta}$, by chain rule we have

$$\begin{aligned}\frac{\partial}{\partial \boldsymbol{\theta}} l(\boldsymbol{p}, y) = \frac{\partial}{\partial \boldsymbol{z}} l(\boldsymbol{z}, y) \mid_{\boldsymbol{z}=h_{\boldsymbol{\phi}}(\dots)} [&\frac{\partial}{\partial \boldsymbol{u}} h_{\boldsymbol{\phi}}(\boldsymbol{u}, f_{\boldsymbol{\theta}}(x_2), \dots, f_{\boldsymbol{\theta}}(x_m)) \frac{\partial}{\partial \boldsymbol{\theta}} f_{\boldsymbol{\theta}}(x_1) \\ &+ \frac{\partial}{\partial \boldsymbol{u}} h_{\boldsymbol{\phi}}(f_{\boldsymbol{\theta}}(x_1), \boldsymbol{u}, \dots, f_{\boldsymbol{\theta}}(x_m)) \frac{\partial}{\partial \boldsymbol{\theta}} f_{\boldsymbol{\theta}}(x_2) + \dots + \frac{\partial}{\partial \boldsymbol{u}} h_{\boldsymbol{\phi}}(f_{\boldsymbol{\theta}}(x_1), \dots, \boldsymbol{u}) \frac{\partial}{\partial \boldsymbol{\theta}} f_{\boldsymbol{\theta}}(x_m)]\end{aligned} \tag{8}$$

By stopping the gradient computation over $x_2, \dots, x_m$, we are dropping the terms related to $\frac{\partial}{\partial \boldsymbol{\theta}} f_{\boldsymbol{\theta}}(x_i)$, $i \geq 2$, from the above formula. In optimization, we say $\boldsymbol{q}$ is a descent direction if $\langle q, \partial_\theta l \rangle < 0$. The negative gradient $-\partial_{\boldsymbol{\theta}} l$ is clearly a descent direction. The gradient above in Equation (8) is in the form $\partial_{\boldsymbol{\theta}} l = c \sum_{i=1}^m \boldsymbol{g}_i$, where $\boldsymbol{g}_i$ are the gradient term of the ith segment in the equation. By dropping terms in the backpropagation, we are assuming either the contribution from $\boldsymbol{g}_1$ (or $\boldsymbol{g}_{M(\boldsymbol{\alpha})}$ in the attention model of Equation 5) dominates the contributions from other segments, or the gradients $\boldsymbol{g}_i$ from different segments point towards similar directions, so that the truncated gradient is still a descent direction.

In some cases, if the storage of partial gradients $\frac{\partial}{\partial \boldsymbol{u}} h_{\boldsymbol{\phi}}(\boldsymbol{z}_1, \dots, \boldsymbol{u}, \dots, \boldsymbol{z}_m)$ is feasible, it is possible to compute the full gradient in two passes over the mini-batch. For example, in the basic Cat-BERT model for classification, this partial gradient is $\boldsymbol{W}_i$ scaled by the derivative of softmax, where i is the index of the segment. We can compute these $\boldsymbol{W}_i$ with the derivative of the softmax as scaling factors in a first pass, and accumulate gradients over $f_{\boldsymbol{\theta}}(x_i)$ scaled by these factors in a second pass over the mini-batch.