Context Analysis for Pre-trained Masked Language Models

Yi-An Lai **Garima Lalwani** **Yi Zhang**
AWS AI HLT
{yianl,glalwani,yizhngn}@amazon.com

Abstract

Pre-trained language models that learn contextualized word representations from a large unannotated corpus have become a standard component for many state-of-the-art NLP systems. Despite their successful applications in various downstream NLP tasks, the extent of contextual impact on the word representation has not been explored. In this paper, we present a detailed analysis of contextual impact in Transformer- and BiLSTM-based masked language models. We follow two different approaches to evaluate the impact of context: a masking based approach that is architecture agnostic, and a gradient based approach that requires back-propagation through networks. The findings suggest significant differences on the contextual impact between the two model architectures. Through further breakdown of analysis by syntactic categories, we find the contextual impact in Transformer-based MLM aligns well with linguistic intuition. We further explore the Transformer attention pruning based on our findings in contextual analysis.

1 Introduction

Pre-trained masked language models (MLM) such as BERT (Devlin et al., 2019) and ALBERT (Lan et al., 2019) have set state-of-the-art performance on a broad range of NLP tasks. The success is often attributed to their ability to capture complex syntactic and semantic characteristics of word use across diverse linguistic contexts (Peters et al., 2018). Yet, how these pre-trained MLMs make use of the context remains largely unanswered.

Recent studies have started to inspect the linguistic knowledge learned by pre-trained LMs such as word sense (Liu et al., 2019a) , syntactic parse trees (Hewitt and Manning, 2019), and semantic relations (Tenney et al., 2019). Others directly analyze model's intermediate representations and attention weights to understand how they work (Kovaleva et al., 2019; Voita et al., 2019).

While previous works either assume access to model's internal states or take advantage of model's special structures such as self-attention maps, these analysis are difficult to generalize as the architectures evolve. In this paper, our work complements these previous efforts and provides a richer understanding of how pre-trained MLMs leverage context without assumptions on architectures. We aim to answer following questions: (i) How much context is relevant to and used by pre-trained MLMs when composing representations? (ii) How far do MLMs look when leveraging context? That is, what are their effective context window sizes? We further define a target word's essential context as the set of context words whose absence will make the MLM indiscriminate of its prediction. We analyze linguistic characteristics of these essential context words to better understand how MLMs manage context.

We investigate the contextual impacts in MLMs via two approaches. First, we propose the *context perturbation* analysis methodology that gradually masks out context words following a predetermined procedure and measures the change in the target word probability. For example, we iteratively mask words that have the least change to the target word probability until the probability deviates too much from the start. At this point, the remaining words are relevant to and used by the MLM to represent the target word, since further perturbation causes a notable prediction change. Being model agnostic, our approach looks into the contextualization in the MLM task itself, and quantify them only on the output layer. We refrain from inspecting internal representations since new architectures might not have a clear notion of "layer" with inter-leaving jump connections such as those in Guo et al. (2019) and Yao et al. (2020).

Findings of the Association for Computational Linguistics: EMNLP 2020, pages 3789–3804

The second approach is adapted from Falenska and Kuhn (2019) and estimates the impact of an input subword to the target word probability via the norm of the gradients. We study pre-trained MLMs based on two different architectures: Transformer and BiLSTM. The former is essentially BERT and the latter resembles ELMo (Peters et al., 2018). Although the scope in this work is limited to the comparison between two popular architectures, the same novel methodology can be readily applied to multilingual models as well as other Transformer-based models pre-trained with MLM.

From our analysis, when encoding words using sentence-level inputs, we find that BERT is able to leverage 75% of context on average in terms of the sentence length, while BiLSTM has the effective context size of around 30%. The gap is compelling for long-range context more than 20 words away, wherein, BERT still has a 65% chance to leverage the words in comparison to BiLSTM that only has 10% or less to do so. In addition, when restricted to a local context window around the target word, we find that the effective context window size of BERT is around 78% of the sentence length, whereas BiLSTM has a much smaller window size of around 50%. With our extensive study on how different pre-trained MLMs operate when producing contextualized representations and what detailed linguistic behaviors can be observed, we exploited these insights to devise a pilot application. We apply attention pruning that restricts the attention window of BERT based on our findings. Results show that the performance remains the same with its efficiency improved. Our main contributions can be briefly summarized as:

- Standardize the pre-training setup (model size, corpus, objective, etc.) for a fair comparison between different underlying architectures.
- Novel design of a straight-forward and intuitive perturbation-based analysis procedure to quantify impact of context words.
- Gain insights about how different architectures behave differently when encoding contexts, in terms of number of relevant context words, effective context window sizes, and more fine-grained break-down with respect to POS and dependency structures.
- Leverage insights from our analysis to conduct a pilot application of attention pruning on a sequence tagging task.

2 Related Work

Pre-training language models (LM) to learn contextualized word representations from a large amount of unlabeled text has been shown to benefit downstream tasks (Howard and Ruder, 2018; Peters et al., 2018; Radford et al., 2019). Masked language modeling (MLM) introduced in BERT (Devlin et al., 2019) has been widely used as the pre-training task in works including RoBERTa (Liu et al., 2019b), SpanBERT (Joshi et al., 2020), and ALBERT (Lan et al., 2019). Many of them employ the Transformer architecture (Vaswani et al., 2017) that uses multi-head self-attention to capture context.

To assess the linguistic knowledge learned by pre-trained LMs, probing task methodology suggest training supervised models on top of the word representations (Ettinger et al., 2016; Hupkes et al., 2018; Belinkov and Glass, 2019; Hewitt and Liang, 2019). Investigated linguistic aspects span across morphology (Shi et al., 2016; Belinkov et al., 2017; Liu et al., 2019a), syntax (Tenney et al., 2019; Hewitt and Manning, 2019), and semantics (Conneau et al., 2018; Liu et al., 2019a).

Another line of research inspects internal states of pre-trained LMs such as attention weights (Kovaleva et al., 2019; Clark et al., 2019) or intermediate word representations (Coenen et al., 2019; Ethayarajh, 2019) to facilitate our understanding of how pre-trained LMs work. In particular, Voita et al. (2019) studies the evolution of representations from the bottom to top layers and finds that, for MLM, the token identity tends to be recreated at the top layer. A close work to us is Khandelwal et al. (2018), they conduct context analysis on LSTM language models to learn how much context is used and how nearby and long-range context is represented differently.

Our work complements prior efforts by analyzing how models pre-trained by MLM make use of context and provides insights that different architectures can have different patterns to capture context. Distinct from previous works, we leverage no specific model architecture nor intermediate representations while performing the context analysis.

Another related topic is generic model interpretations including LIME (Ribeiro et al., 2016), SHAP (Lundberg and Lee, 2017), and Ancona et al. (2017). Despite the procedural similarity, our work focuses on analyzing how pre-trained MLMs behave when encoding contexts and our methodology is both model-agnostic and training-free.

Model	MNLI-(m/mm)	QQP	QNLI	SST-2	CoLA	STS-B	MRPC	RTE	**Avg**
BERT (Devlin et al., 2019)	84.6/83.4	71.2	90.5	93.5	52.1	85.8	88.9	66.4	79.6
BiLSTM + ELMo	72.9/73.4	65.6	71.7	90.2	35.0	64.0	80.8	50.1	67.1
BERT (ours)	84.6/84.0	71.0	91.5	93.6	55.7	86.2	88.6	67.4	80.3
BiLSTM (ours)	70.9/70.2	63.0	73.7	90.6	30.5	67.6	81.2	54.6	66.9

Table 1: GLUE benchmark test results. BiLSTM+ELMo numbers are cited from (Wang et al., 2018). The comparable performance to previous works validates our pre-training process.

3 Masked Language Modeling

Given a sentence $X = (w_1, w_2, ..., w_L)$ where each word w_i is tokenized into l_i subwords $(s_{i1}, ..., s_{il_i})$, a portion of tokens are randomly masked with the `[MASK]` token. MLMs are trained to recover the original identity of masked tokens by minimizing the negative log likelihood (NLL). In practice, BERT (Devlin et al., 2019) randomly replaces 15% tokens by `[MASK]` for 80% of the cases, keep the original token for 10% of the time, and replace with a random token for the remaining 10% of the cases.

For context analysis, we perform the masking and predictions at the word level. Given a *target word* w_t, all its subwords are masked $X_{\setminus t} = (...s_{(t-1)l_{t-1}}, \texttt{[MASK]}, ..., \texttt{[MASK]}, s_{(t+1)1}...)$. Following Devlin et al. (2019), the conditional probability of w_t can be computed from outputs of MLMs with the independence assumption between subwords:

$$\begin{aligned} P(w_t|X_{\setminus t}) &= P(s_{t1} \dots s_{tl_t}|X_{\setminus t}) \\ &= \prod_{i=1}^{l_t} P(s_{ti}|X_{\setminus t}). \end{aligned} \quad (1)$$

To investigate how MLMs use context, we propose procedures to perturb the input sentence from $X_{\setminus t}$ to $\widetilde{X}_{\setminus t}$ and monitor the change in the target word probability $P(w_t|X_{\setminus t})$.

4 Approach

Our goal is to analyze the behaviors of pre-trained MLMs when leveraging context to recover identity of the masked target word w_t, e.g. to answer questions such as how many context words are considered and how large the context window is. To this end, we apply two analysis approaches. The first one is based on the masking or perturbation of input context which is architecture agnostic. The second gradient-based approach requires back-propagation through networks.

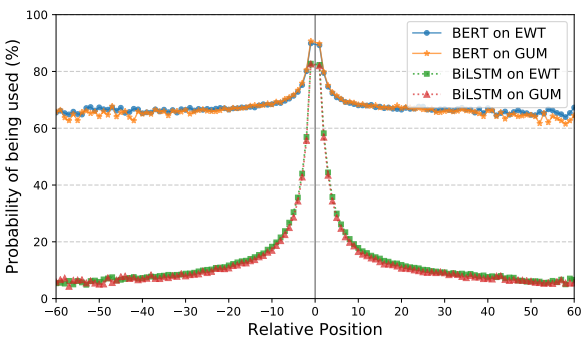
Our first approach performs context perturbation analysis on pre-trained LMs at inference time and measures the change in masked target word probabilities. To answer each question, we start from $X_{\setminus t}$ and design a procedure Ψ that iteratively processes the sentence from last perturbation $\widetilde{X}^{k+1}_{\setminus t} = \Psi(\widetilde{X}^{k}_{\setminus t})$. The patterns of $P(w_t|\widetilde{X}^{k}_{\setminus t})$ offer insights to our question. An example of Ψ is to mask out a context word that causes the least or negligible change in $P(w_t|\widetilde{X}^{k}_{\setminus t})$. It's worth mentioning that as pre-trained LMs are often used off-the-shelf as a general language encoder, we do not further finetune the model on the analysis dataset but directly analyze how they make use of context. In practice, we loop over a sentence word-by-word to set the word as the target first and use rest of words as the context for our masking process. Since we do the context analysis only with model inference, the whole process is fast - around half day on a 4-GPU machine to process 12k sentences.

Our second approach estimates the impact of an input subword s_{ij} to $P(w_t|X_{\setminus t})$ by using derivatives. Specifically, we adapt the IMPACT score proposed in Falenska and Kuhn (2019) to our questions. The score $\text{IMPACT}(s_{ij}, w_t)$ can be computed with the gradients of the negative log likelihood (NLL) with respect to the subword embedding:

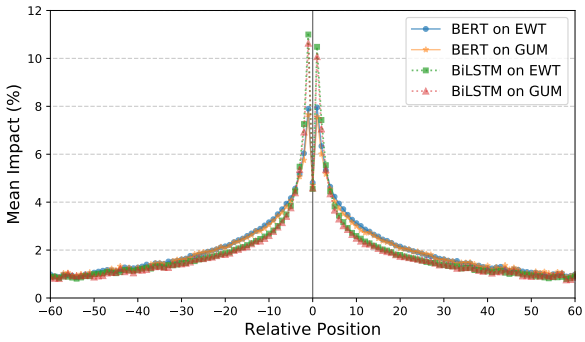
$$\text{IMPACT}(s_{ij}, w_t) = \frac{\| \frac{\partial(\log P(w_t|X_{\setminus t}))}{\partial s_{ij}} \|}{\sum_m^L \sum_n^{l_m} \| \frac{\partial - \log P(w_t|X_{\setminus t})}{\partial s_{mn}} \|}. \quad (2)$$

The l_2-norm of the gradient is used as the impact measure and normalized over all the subwords in a sentence. In practice, we report the impact of a context word w_i by adding up the scores from its subwords $\sum_j^{l_i} \text{IMPACT}(s_{ij}, w_t)$.

We investigate two different encoder architectures of pre-trained MLMs. The first one is BERT that employs 12 Transformer encoder layers, 768 dimension, 3072 feed-forward hidden size, and 110 million parameters. The other uses a standard bidirectional LSTM (Hochreiter and Schmidhuber,



(a) Masking-based context impacts



(b) Gradient-based context impacts

Figure 1: Analysis of how much context is used by MLMs. **(a)** Context words at all relative positions have significantly higher probabilities to be considered by BERT, compared with BiLSTM. **(b)** Gradient-based IMPACT score also shows that BERT considers more distant context than BiLSTM, impact scores are normalized to 100%.

	EWT	GUM
Sentences	9,673	3,197
Words	195,093	67,585
Mean Length	20.17	21.14
Median Length	17	19
Max Length	159	98

Table 2: Statistics of datasets used for analysis

1997) that has 3 layers, 768 embedding dimension, 1200 hidden size, and around 115 million parameters. The BiLSTM model parameters are chosen so that they resemble ELMo while being close to BERT in model size. To have a fair comparison, we pre-train both encoders from scratch on the uncased Wikipedia-book corpus (*wikibook*) with the same pre-training setup as in Devlin et al. (2019). For BiLSTM, we add a linear layer and a LayerNorm (Ba et al., 2016) on top, to project outputs into 768 dimension. We validate our pre-trained models by fine-tuning them on GLUE benchmark (Wang et al., 2018) in single-task manner and report test performance comparable to previous works in Table 1. Our pre-trained BiLSTM-based MLM also gets comparable results to ELMo (Peters et al., 2018).

We perform MLM context analysis on two English datasets from the Universal Dependencies (UD) project, English Web Treebank (EWT) (Silveira et al., 2014) and Georgetown University Multilayer corpus (GUM) (Zeldes, 2017). Datasets from the UD project provide consistent and rich linguistic annotations across diverse genres, enabling us to gain insights towards the contexts in MLMs. We use the training set of each dataset for analysis. EWT consists of $9,673$ sentences from web blogs, emails, reviews, and social media with the median length being 17 and maximum length being 159 words. GUM comprises $3,197$ sentences from Wikipedia, news articles, academic writing, fictions, and how-to guides with the median length being 19 and maximum length being 98 words. The statistics of datasets are summarized in Table 2.

5 How much context is used?

Self-attention is designed to encode information from any position in a sequence, whereas BiLSTMs model context through the combination of long- and short-term memories in both left-to-right and right-to-left directions. For MLMs, the entire sequence is provided to produce contextualized representations, it is unclear how much context in the sequence is used by different MLMs.

In this section, we first propose a perturbation procedure Ψ that iteratively masks out a context word contributing to the least absolute change of the target word probability $P(w_t|\widetilde{X}^k_{\setminus t})$. That is, we incrementally eliminate words that do not penalize MLMs predictions one by one, until further masking cause $P(w_t|\widetilde{X}^k_{\setminus t})$ to deviate too much from the original probability $P(w_t|X_{\setminus t})$. At this point, the remaining unmasked words are considered being used by the MLM since corrupting any of them causes a notable change in target word prediction.

In practice, we identify deviations using the negative log likelihood (NLL) that corresponds to the loss of MLMs. Assuming NLL has a variance of ϵ at the start of masking, we stop the perturbation procedure when the increase on NLL $\log P(w_t|X_{\setminus t}) - \log P(w_t|\widetilde{X}^k_{\setminus t})$ exceeds 2ϵ. We observe that NLLs fluctuate around $[-0.1, 0.1]$ at

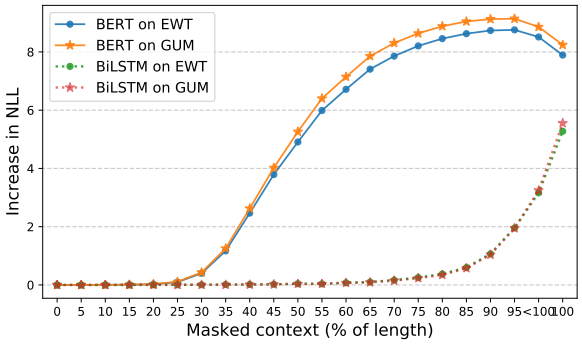
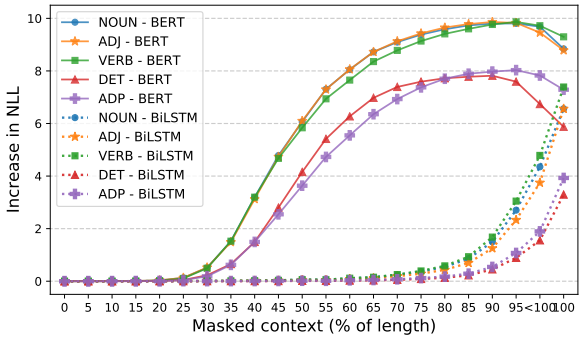
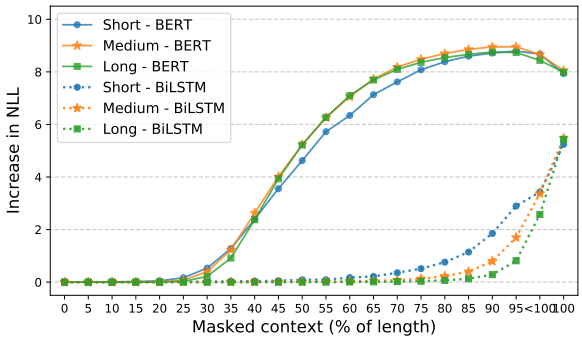


Figure 2: Context usage analysis for MLMs via elimination of irrelevant context. BERT uses about 75% of context while BiLSTM uses around 30%.



(a) Masking-based: Different syntactic categories



(b) Masking-based: Different length buckets

Figure 3: Context usage analysis for MLMs, instances bucketed by syntactic categories of target words or input lengths. **(a)** More context is used to model context words than function words. **(b)** BERT uses fixed amounts of context while BiLSTM's context usage percentage varies by input length.

the start of masking, hence we terminate our procedure when the NLL increase reaches 0.2. We report the *effective context size* in terms of percentage of length to normalize the length impact. The analysis process is repeated using each word in a sentence as the target word for all sentences in the dataset.

For our second approach, we follow equation 2 to calculate the normalized impact of each subword to the target word and aggregate them for each context word to get IMPACT(w_i, w_t). We group the IMPACT scores by relative position of a word w_i to the target word w_t and plot the average. To compare with our first approach, we also use masking-based method to analyze that for a word with a specific relative position, what would be its probability of being used by a MLM.

BERT uses distant context more than BiLSTM. After our masking process, a subset of context words are tagged as "being used" by the pre-trained LM. In Figure 1a, we aggregated results in terms of relative positions (context-word-to-target-word) for all targets and sentences. "Probability of being used %" denotes when a context word appears at a relative position to target, how likely is it to be relevant to the pre-trained LM.

Figure 1a shows that context words at all relative positions have substantially higher probabilities to be considered by BERT than BiLSTM. And BiLSTM focuses sharply on local context words, while BERT leverages words at almost all the positions. A notable observation is that both models consider a lot more often, words within distance around $[-10, 10]$ and BERT has as high as 90% probability to use the words just before and after the target word. Using gradient-based analysis, Figure 1b shows similar results that BERT considers more distant context than BiLSTM and local words have more impact to both models than distant words.

There are notable differences between two analysis approaches. Since the gradient-based IMPACT score is normalized into a distribution across all positions, it does not show the magnitude of the context impact on the two different models. On the other hand, the masking-based analysis shows that BERT uses words at each position more than BiLSTM based on absolute probability values. Another important difference is that the gradient-based approach is a glass-box method and requires back-propagation through networks, assuming the models to be differentiable. On the other hand, the masking-based approach treats the model as a black-box and has no differentiability assumption on models. In the following sections, we will continue analysis with the masking-based approach.

BERT uses 75% of words in a sentence as context while BiLSTM considers 30%. Figure 2 shows the increase in NLL when gradually masking out the least relevant words. BERT's NLL increases considerably when 25% of context are

masked, suggesting that BERT uses around 75% of context. For BiLSTM, its NLL goes up remarkably after 70% of context words are masked, meaning that it considers around 30% of context. Albeit having the same capacity, we observe that BERT uses more than two times of context words into account than BiLSTM. This could explain the superior fine-tuning performance of BERT on tasks demanding more context to solve. We observe that pre-trained MLMs have consistent behaviors across two datasets that have different genres. For the following analysis, we report results combining EWT and GUM datasets.

Content words needs more context than function words. We bucket instances based on the part-of-speech (POS) annotation of the target word. Our analysis covers *content words* including nouns, verbs and adjectives, and *function words* including adpositions and determiners. Figure 3a shows that both models use significantly more context to represent content words than function words, which is aligned with linguistic intuitions (Boyd-Graber and Blei, 2009). The findings also show that MLMs handle content and function words in a similar manner as regular language models do, which are previously analyzed by Wang and Cho (2016); Khandelwal et al. (2018).

BiLSTM context usage percentage varies by input sentence length, whereas for BERT, it doesn't. We categorize sentences with length shorter than 25 as *short*, between 25 and 50 as *medium*, and more than 50 as *long*. Figure 3b shows that BiLSTM uses 35% of context for short sentences, 20% for medium, and only 10% for long sentences. On the other hand, BERT leverages fixed 75% of context words regardless of the sentence length.

6 How far do MLMs look?

In the previous section, we looked at how much context is relevant to the two MLMs via an elimination procedure. From Figure 1a and 1b, we also observe that local context is more impactful than long-range context for MLMs. In this section, we investigate this notion of locality of context even further and try to answer the question of how far away do MLMs actually look at in practice, i.e., what is the *effective context window size (cws)* of each MLM.

For context perturbation analysis, we introduce a locality constraint to the perturbation procedure while masking words. We aim to identify how local versus distant context impacts the target word probability differently. We start with masking all the words around the target, i.e., the model only relies on its priors learned during pre-training (*cws* $\sim 0\%$ [1]). We iteratively increase the *cws* on both sides until all the surrounding context is available (*cws* $\sim 100\%$). Details of the masking procedure can be found in Appendix. We report the increase in NLL compared to when the entire context is available $\log P(w_t|X_{\setminus t}) - \log P(w_t|\widetilde{X}^k_{\setminus t})$, with respect to the increasing *cws*. This process is repeated using each word as the target word, for all the sentences in the dataset. We aggregate and visualize the results similar to section 5 and use the same threshold (0.2) as before to mark the turning point.

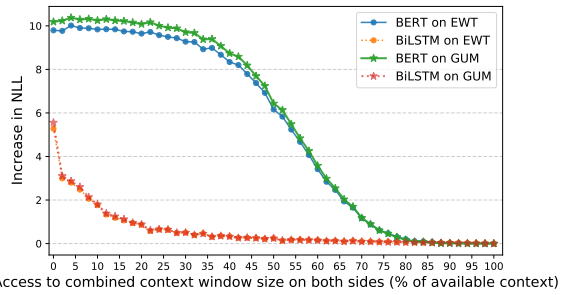
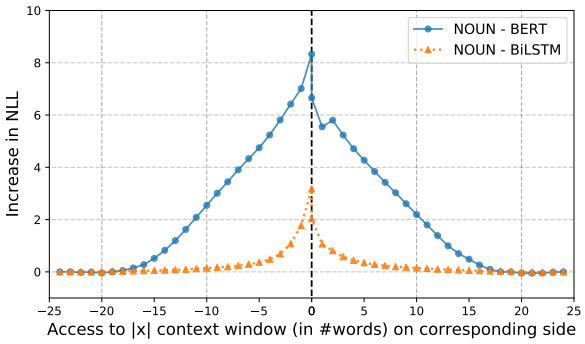


Figure 4: Change in NLL as the context window size around target word (left and right combined) changes

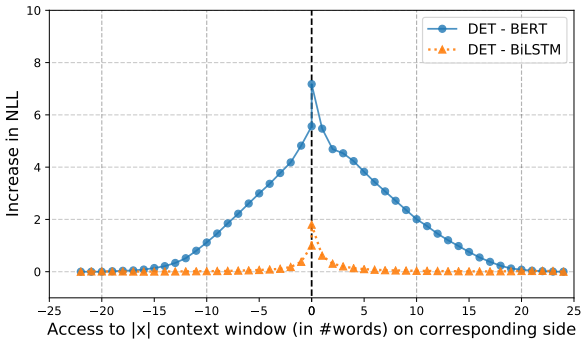
As shown in Figure 4, increasing the *cws* around target word reduces the change of NLL until a point where the gap is closed. The plot clearly highlights the differences in the behavior of two models - for BERT, words within *cws* of 78% impact the model's ability to make target word predictions, whereas, for BiLSTM, only words within *cws* of 50% affect the target word probability. This shows that **BERT**, leveraging entire sequence by self-attention, **looks at a much wider context window size (effective *cws* $\sim 78\%$) in comparison to the recurrent architecture BiLSTM (effective *cws* $\sim 50\%$)**. Besides, BiLSTM shows a clear notion of contextual locality that it tends to consider very local context for target word prediction.

Furthermore, we investigate the symmetricity of *cws* on either side by following the same procedure but now separately on each side of the target word. We iteratively increase *cws* either on left side or right side while keeping the rest of the words unmasked. More details of the analysis procedure can

[1] % here denotes the percent of available context w.r.t. (sentence-length - 1) context words, excluding target word.



(a) Target word belonging to POS - NOUN



(b) Target word belonging to POS - DET

Figure 5: Symmetricity analysis of context window size for two target word syntactic categories from short sentences $l \leq 25$ **(a)** For NOUN as target, BERT looks at words within the window [-16, 16], while BiLSTM has the context window [-7, 7]. **(b)** When target word is DET, BERT looks at words within the window [-14, 18], while BiLSTM has the context window [-1, 3].

be found in the Appendix. The analysis results are further bucketed by the POS categories of target words as well as input sentence lengths, similar to Section 5, to gain more fine-grained insights. In Figure 5, we show the symmetricity analysis of *cws* for short length sentences and target word with POS tags - NOUN and DET. The remaining plots for medium and long length sentences with target word from other POS tags are shown in Appendix due to the lack of space.

From Figure 5, both models show similar behaviors across different POS tags when leveraging symmetric/asymmetric context. The *cws* attended to on either side is rather similar when target words are NOUN, whereas for DET, we observe both models paying more attention to right context words than the left. This observation aligns well with linguistic intuitions for English language. We can also observe the striking difference between two models in effective *cws*, with BERT attending to a much larger *cws* than BiLSTM. The difference in

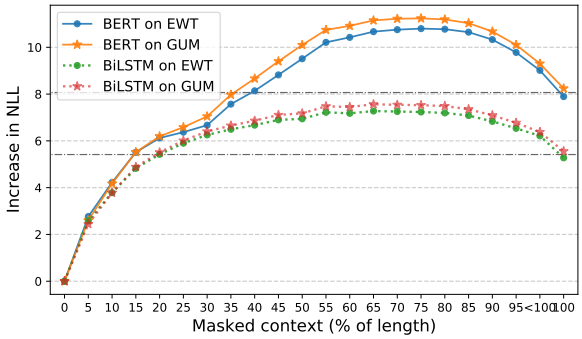


Figure 6: Identifying essential context by masking most important words. 35-40% of context is critical to BERT while BiLSTM sees about 20% as essential.

the left and right *cws* for DET appears to be more pronounced for BiLSTM in comparison to BERT. We hypothesize that this is due to BiLSTM's overall smaller *cws* (left + right) which makes it only attend to the most important words that happen to be mostly in the right context.

7 What kind of context is essential?

There is often a core set of context words that is essential to capture the meaning of target word. For example, *"Many people think cotton is the most comfortable* ____ *to wear in hot weather."* Although most context is helpful to understand the masked word *fabric*, *cotton* and *wear* are essential as it would be almost impossible to make a guess without them.

In this section, we define essential context as words such that when they are absent, MLMs would have no clue about the target word identity, i.e., the target word probability becomes close to masking out the entire sequence $P(w_t|\widetilde{X}_{mask\_all})$. To identify essential context, we design the perturbation Ψ to iteratively mask words bringing largest drop in $P(w_t|\widetilde{X}^k_{\setminus t})$ until we reach a point, where the increase in NLL just exceeds the 100% mask setting $(\log P(w_t|X_{\setminus t}) - \log P(w_t|\widetilde{X}_{mask\_all}))$. The words masked using above procedure are labelled as essential context words. We further analyze linguistic characteristics of the identified essential context words.

BERT sees 35% of context as essential, whereas BiLSTM perceives around 20%. Figure 6 shows that on average, BERT recognizes around 35% of context as essential when making predictions, i.e., when the increase in NLL is on par with masking all context. On the other hand, BiLSTM sees only 20% of context as essential. This implies that BERT would be more robust than the BiLSTM-

Context	Distance	All targets	NOUN	ADJ	VERB	DET	ADP
Full-context	Linear	9.37	9.33	9.23	8.97	9.47	9.47
BERT-essential	Linear	6.25	6.42	5.89	5.87	5.65	6.11
BiLSTM-essential	Linear	5.49	6.43	6.03	6.32	4.20	3.77
Full-context	Tree	3.63	3.37	3.73	2.83	4.13	4.31
BERT-essential	Tree	2.91	2.66	2.88	2.20	3.18	3.46
BiLSTM-essential	Tree	2.74	2.66	2.90	2.28	2.74	2.73

Table 3: Mean distances from essential context words to target words. *Linear* means linear positional distance and *Tree* denotes the dependency tree walk distance. Results are bucketed by part-of-speech tags of target words.

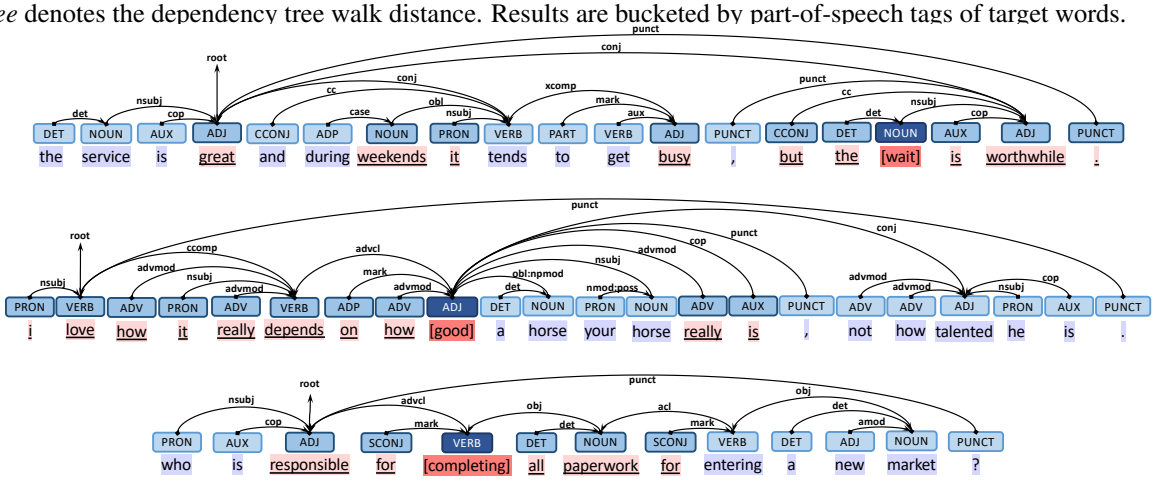


Figure 7: Essential context identified by BERT along with POS tags and dependency trees. Words in brackets are targets. Words underlined are essential.

based encoder in the presence of noisy input, a finding also supported by Yin et al. (2020); Jin et al. (2019), as it will be harder to confuse the model completely given larger size of essential context words set in comparison to BiLSTM.

Essential words are close to target words in both linear position and on dependency tree. Table 3 calculates the mean distances from identified essential words to the target words on combined EWT and GUM datasets. Both the models tend to identify words much closer to the target as essential, whether we consider linear positional distance or node distance in dependency trees. We use annotated dependency relations to extract the traversal paths from each essential word to the target word in dependency tree. We find that the top 10 most frequent dependency paths often correspond with the common syntactic structures in natural language. For example, when target words are NOUN, the top 3 paths are `DET(up:det)`⇒NOUN, `ADP(up:case)`⇒NOUN, `ADJ(up:amod)`⇒NOUN for both models. Further, we also look at the dependency paths of essential words which are unique to each model. The comparison shows that words of common dependency paths are sometimes identified as essential by BERT but not by BiLSTM and vice versa. This suggests that there is room to improve MLMs by making them consistently more aware of input's syntactic structures, possibly by incorporating dependency relations into pre-training. The full lists of top dependency paths are presented in the Appendix.

Figure 7 shows examples of essential words from BERT with POS tags and dependency relations. Words in square brackets are target words and the underlined words are essential words. We observe that words close to the target in the sentence as well as in the dependency tree are often seen as essential. We can also see that BERT often includes the root of the dependency tree as an essential word.

8 Application: Attention Pruning for Transformer

As a pilot application, we leverage insights from analysis in previous sections to perform attention pruning for Transformer. Transformer has achieved impressive results in NLP and has been used for long sequences with more than 10 thousand tokens (Liu et al., 2018). Self-attention for a sequence of

Model	Dev F1	Test F1
BERT - Full	94.9(0.2)	90.8(0.1)
BERT - Dynamic Pruning	94.7(0.2)	90.6(0.2)
BERT - Static Pruning	94.5(0.2)	90.3(0.1)

Table 4: CoNLL-2003 Named Entity Recognition results (5 seeds). The attention pruning based on our findings gives comparable results to the original BERT.

length L is of $\mathcal{O}(L^2)$ complexity in computation and memory. Many works attempt to improve the efficiency of self-attention by restricting the number of tokens that each input query can attend to (Child et al., 2019; Kitaev et al., 2020).

Our analysis in Section 6 shows that BERT has effective *cws* of around 78%. We perform a dynamic attention pruning by making self-attention neglect the furthest 22% of tokens. Due to the $\mathcal{O}(L^2)$ complexity, this could save around 39% of computation in self-attention. We apply this locality constraint to self-attention when fine-tuning BERT on a downstream task. Specifically, we use the CoNLL-2003 Named Entity Recognition (NER) dataset (Sang and Meulder, 2003) with $200k$ words for training. We fine-tune BERT for NER in the same way as in Devlin et al. (2019). We also explore a static attention pruning that restricts the attention span to be within $[-5, +5]$[2]. Results in Table 4 show that BERT with attention pruning has comparable performance to the original BERT, implying successful application of our analysis findings. Note that we use an uncased vocabulary, which could explain the gap compared to Devlin et al. (2019).

9 Conclusion

In our context analysis, we have shown that BERT has an effective context size of around 75% of input length, while BiLSTM has about 30%. The difference in context usage is striking for long-range context beyond 20 words. Our extensive analysis of context window size demonstrate that BERT uses much larger context window size than BiLSTM. Besides, both models often identify words with common syntactic structures as essential context. These findings not only help to better understand contextual impact in masked language models, but also encourage model improvements in efficiency and effectiveness in future works. On top of that, diving deep into the connection between our context analysis and a model's robustness to noisy texts is also an interesting topic to explore.

[2] With average training set sentence length of 14, this span equates to *cws* of 78%.

Acknowledgments

The authors would like to acknowledge the entire AWS Lex Science team for thoughtful discussions, honest feedback, and full support. We are also very grateful to the reviewers for insightful comments and helpful suggestions.

References

Marco Ancona, Enea Ceolini, Cengiz Öztireli, and Markus Gross. 2017. Towards better understanding of gradient-based attribution methods for deep neural networks. *arXiv preprint arXiv:1711.06104*.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.

Yonatan Belinkov and James Glass. 2019. Analysis methods in neural language processing: A survey. *Transactions of the Association for Computational Linguistics*, 7:49–72.

Yonatan Belinkov, Lluís Màrquez i Villodre, Hassan Sajjad, Nadir Durrani, Fahim Dalvi, and James R. Glass. 2017. Evaluating layers of representation in neural machine translation on part-of-speech and semantic tagging tasks. In *IJCNLP*.

Jordan L Boyd-Graber and David M Blei. 2009. Syntactic topic models. In *Advances in neural information processing systems*, pages 185–192.

Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. 2019. Generating long sequences with sparse transformers. *ArXiv*, abs/1904.10509.

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. 2019. What does bert look at? an analysis of bert's attention. *ArXiv*, abs/1906.04341.

Andy Coenen, Emily Reif, Ann Yuan, Been Kim, Adam Pearce, Fernanda B. Viégas, and Martin Wattenberg. 2019. Visualizing and measuring the geometry of bert. In *NeurIPS*.

Alexis Conneau, Germán Kruszewski, Guillaume Lample, Loïc Barrault, and Marco Baroni. 2018. What you can cram into a single vector: Probing sentence embeddings for linguistic properties. In *ACL*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. *ArXiv*, abs/1810.04805.

Kawin Ethayarajh. 2019. How contextual are contextualized word representations? comparing the geometry of bert, elmo, and gpt-2 embeddings. *ArXiv*, abs/1909.00512.

Allyson Ettinger, Ahmed Elgohary, and Philip Resnik. 2016. Probing for semantic evidence of composition by means of simple classification tasks. In *RepEval@ACL*.

Agnieszka Falenska and Jonas Kuhn. 2019. The (non-)utility of structural features in bilstm-based dependency parsers. In *ACL*.

Zhijiang Guo, Yan Zhang, Zhiyang Teng, and Wei Lu. 2019. Densely connected graph convolutional networks for graph-to-sequence learning. *Transactions of the Association for Computational Linguistics*, 7:297–312.

John Hewitt and Percy Liang. 2019. Designing and interpreting probes with control tasks. In *EMNLP/IJCNLP*.

John Hewitt and Christopher D. Manning. 2019. A structural probe for finding syntax in word representations. In *NAACL-HLT*.

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

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *ACL*.

Dieuwke Hupkes, Sara Veldhoen, and Willem Zuidema. 2018. Visualisation and'diagnostic classifiers' reveal how recurrent and recursive neural networks process hierarchical structure. *Journal of Artificial Intelligence Research*, 61:907–926.

Di Jin, Zhijing Jin, Joey Tianyi Zhou, and Peter Szolovits. 2019. Is bert really robust? a strong baseline for natural language attack on text classification and entailment. *arXiv: Computation and Language*.

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.

Urvashi Khandelwal, He He, Peng Qi, and Dan Jurafsky. 2018. Sharp nearby, fuzzy far away: How neural language models use context. In *ACL*.

Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. 2020. Reformer: The efficient transformer. *ArXiv*, abs/2001.04451.

Olga V. Kovaleva, Alexey Romanov, Anna Rogers, and Anna Rumshisky. 2019. Revealing the dark secrets of bert. In *EMNLP/IJCNLP*.

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. *arXiv preprint arXiv:1909.11942*.

Nelson F. Liu, Matt Gardner, Yonatan Belinkov, Matthew E. Peters, and Noah A. Smith. 2019a. Linguistic knowledge and transferability of contextual representations. *ArXiv*, abs/1903.08855.

Peter J. Liu, Mohammad Saleh, Etienne Pot, Ben Goodrich, Ryan Sepassi, Lukasz Kaiser, and Noam Shazeer. 2018. Generating wikipedia by summarizing long sequences. *ArXiv*, abs/1801.10198.

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

Scott M Lundberg and Su-In Lee. 2017. A unified approach to interpreting model predictions. In *Advances in neural information processing systems*, pages 4765–4774.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *ArXiv*, abs/1802.05365.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8):9.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. " why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. *ArXiv*, cs.CL/0306050.

Xing Shi, Inkit Padhi, and Kevin Knight. 2016. Does string-based neural mt learn source syntax? In *EMNLP*.

Natalia Silveira, Timothy Dozat, Marie-Catherine de Marneffe, Samuel Bowman, Miriam Connor, John Bauer, and Christopher D. Manning. 2014. A gold standard dependency corpus for English. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC-2014)*.

Ian Tenney, Patrick Xia, Berlin Chen, Alex Wang, Adam Poliak, R. Thomas McCoy, Najoung Kim, Benjamin Van Durme, Samuel R. Bowman, Dipanjan Das, and Ellie Pavlick. 2019. What do you learn from context? probing for sentence structure in contextualized word representations. *ArXiv*, abs/1905.06316.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all

you need. In *Advances in neural information processing systems*, pages 5998–6008.

Elena Voita, Rico Sennrich, and Ivan Titov. 2019. The bottom-up evolution of representations in the transformer: A study with machine translation and language modeling objectives. In *EMNLP/IJCNLP*.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*.

Tian Wang and Kyunghyun Cho. 2016. Larger-context language modelling with recurrent neural network. In *ACL*.

Shaowei Yao, Tianming Wang, and Xiaojun Wan. 2020. Heterogeneous graph transformer for graph-to-sequence learning. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7145–7154.

Fan Yin, Quanyu Long, Tao Meng, and Kai-Wei Chang. 2020. On the robustness of language encoders against grammatical errors. *arXiv preprint arXiv:2005.05683*.

Amir Zeldes. 2017. The GUM corpus: Creating multilayer resources in the classroom. *Language Resources and Evaluation*, 51(3):581–612.

A Appendix

B Context Window Size Analysis

B.1 Masking Strategies for Context Window Size Analysis

As mentioned in Section 6, for analyzing how far masked LMs look at within the available context, we follow a masking strategy with locality constraints applied. The masking strategy is as follows - we start from no context available, i.e., all the context words masked and iteratively increase the available context window size (*cws*) on both sides simultaneously, till the entire context is available. This procedure is also depicted in Figure 8. For **symmetricity analysis of *cws***, we follow similar process as above but considering each side of the target word separately. Hence, when considering context words to the left, we iteratively increase the *cws* on the left of target word, keeping the rest of the context words on the right unmasked as shown in Figure 9.

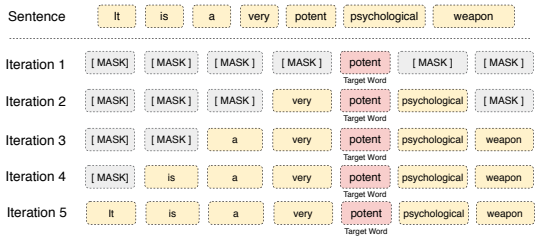


Figure 8: Masking strategy for context window size analysis

B.2 Additional Plots for Symmetricity Analysis of Context Window Size

In Figure 10, we show various plots investigating how context around the target word impact's model performance as we look at left and right context separately. Figures 10a, 10d, 10g, 10j, 10m show left and right *cws* for sentences belonging to **short** length category ($l \leq 25$). The trends show that, where NOUN, ADJ, VERB leverage somewhat symmetric context windows, DET and ADP show asymmetric behavior relying more heavily on right context words for both the models - BERT and BiLSTM. Similar observations can be made for sentences belonging to **medium** length bucket ($l > 25$ *and* $l \leq 50$) with ADP being an exception where BiLSTM shows more symmetric context different than BERT, as shown in Figures 10b, 10e, 10h, 10k, 10n. However, for sentences belonging to

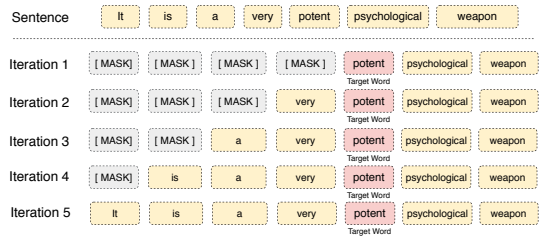


Figure 9: Masking strategy for symmetricity analysis of *cws* on the left

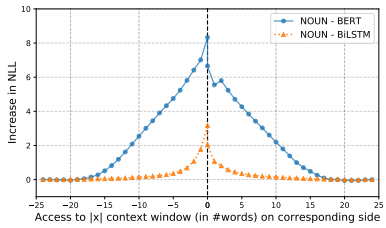
long length bucket ($l > 50$), left and right context window sizes are leveraged quite differently.

We can also see that BiLSTM leverages almost similar number of context words as we moved on to buckets of longer sentence lengths in comparison to BERT which can leverage more context when its available. This is aligned with our observation from Section 5.

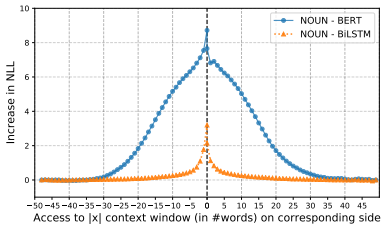
C Dependency Paths from Essential Words to Target Words

Given a target word, BERT or BiLSTM identifies a subset of context words as essential. Based on the dependency relations provided in the datasets, we extract the dependency paths starting from each essential word to the target words, i.e., the path to traverse from an essential word to the given target word in the dependency tree. We summarize the top 10 most frequent dependency paths recognized by BERT or BiLSTM given the target words being a specific part-of-speech category. Table 5, 6, 7, 8, 9 show the results for NOUN, ADJ, VERB, DET, and ADP, respectively. The *up* and *down* denote the direction of traversal, followed by the corresponding relations in the dependency tree. We can see that the top dependency paths for BERT and BiLSTM are largely overlapped with each other. We also observe that these most frequent dependency paths are often aligned with common syntactic patterns. For example, the top 3 paths for NOUN are `DET` $=$`(up:det)`$\Rightarrow$ `NOUN` that could be *"the" cat*, `ADP` $=$`(up:case)`$\Rightarrow$ `NOUN` that could be *"at" home*, and `ADJ` $=$`(up:amod)`$\Rightarrow$ `NOUN` which could be *"white" car*. This implies that both models could be aware of the common syntactic structures in the natural language.

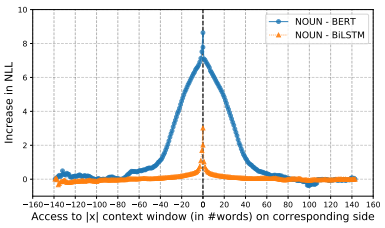
To further compare the behaviors of BERT and BiLSTM when identifying essential context, we count the occurrence of dependency paths based on the **disjoint** essential words. That is, given an input sentence, we only count the dependency paths of



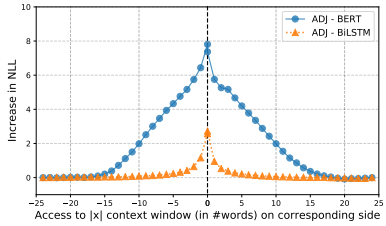
(a) BERT looking at context window size [-16, 16]; biLSTM looking at context window size [-7, 7]



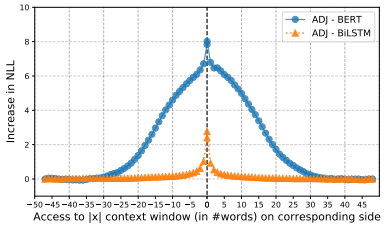
(b) BERT looking at context window size [-29, 32]; biLSTM looking at context window size [-12, 12]



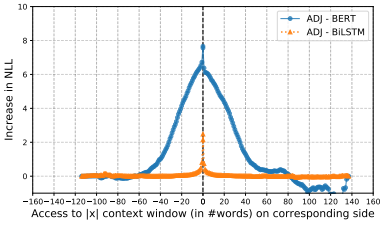
(c) BERT looking at context window size [-135, 72]; biLSTM looking at context window size [-19, 5]



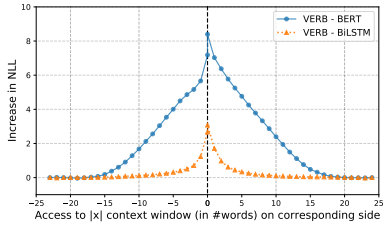
(d) BERT looking at context window size [-15, 16]; biLSTM looking at context window size [-5, 5]



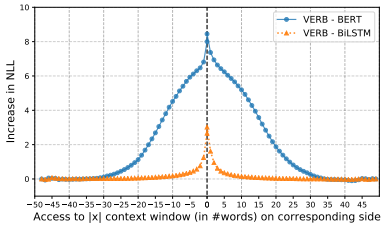
(e) BERT looking at context window size [-28, 30]; biLSTM looking at context window size [-7, 6]



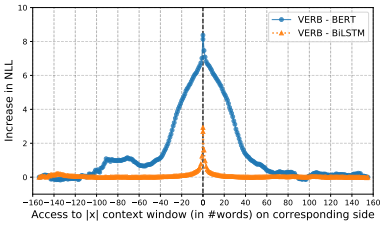
(f) BERT looking at context window size [-54, 77]; biLSTM looking at context window size [-6, 4]



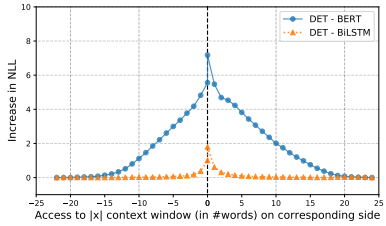
(g) BERT looking at context window size [-14, 16]; biLSTM looking at context window size [-7, 6]



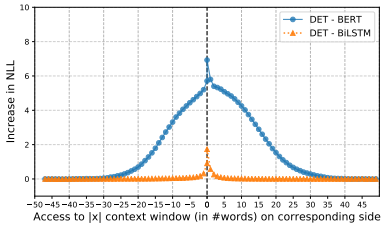
(h) BERT looking at context window size [-28, 30]; biLSTM looking at context window size [-9, 8]



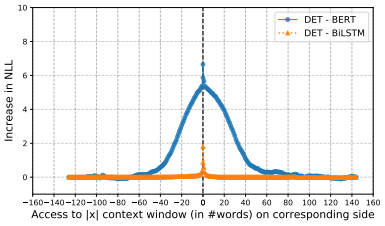
(i) BERT looking at context window size [-102, 148]; biLSTM looking at context window size [-10, 9]



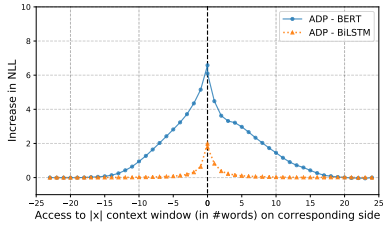
(j) BERT looking at context window size [-14, 18]; biLSTM looking at context window size [-1, 3]



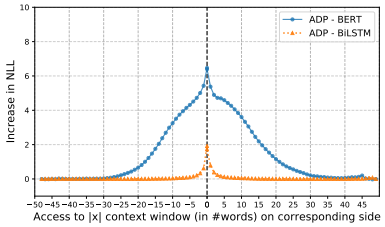
(k) BERT looking at context window size [-25, 31]; biLSTM looking at context window size [-1, 2]



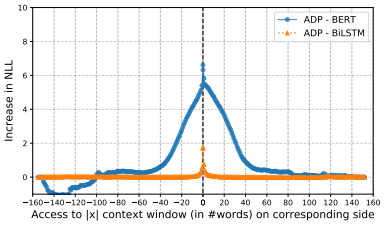
(l) BERT looking at context window size [-50, 75]; biLSTM looking at context window size [-2, 2]



(m) BERT looking at context window size [-13, 16]; biLSTM looking at context window size [-2, 3]



(n) BERT looking at context window size [-25, 30]; biLSTM looking at context window size [-3, 3]



(o) BERT looking at context window size [-99, 113]; biLSTM looking at context window size [-3, 3]

Figure 10: Symmetricity analysis of context window size for different syntactic categories of target word belonging to sentences from buckets of different lengths; along the rows, we consider sentences of different lengths for a given syntactic category: **(a) - (c)** analysis for NOUN; **(d) - (f)** analysis for ADJ; **(g) - (i)** analysis for VERB; **(j) - (l)** analysis for DET; **(m) - (o)** analysis for ADP; along the columns, we consider different syntactic categories for given bucket ranging from short (first column), medium (second column) to long (third column)

BERT	BiLSTM
DET =(up:det)⇒ NOUN	DET =(up:det)⇒ NOUN
ADP =(up:case)⇒ NOUN	ADP =(up:case)⇒ NOUN
ADJ =(up:amod)⇒ NOUN	ADJ =(up:amod)⇒ NOUN
VERB =(down:obj)⇒ NOUN	VERB =(down:obj)⇒ NOUN
ADP =(up:case)⇒ NOUN =(up:nmod)⇒ NOUN	VERB =(down:obl)⇒ NOUN
NOUN =(down:compound)⇒ NOUN	ADP =(up:case)⇒ NOUN =(up:nmod)⇒ NOUN
NOUN =(up:compound)⇒ NOUN	NOUN =(up:nmod)⇒ NOUN
NOUN =(up:nmod)⇒ NOUN	NOUN =(down:nmod)⇒ NOUN
NOUN =(down:nmod)⇒ NOUN	NOUN =(down:compound)⇒ NOUN
VERB =(down:obl)⇒ NOUN	NOUN =(up:compound)⇒ NOUN

Table 5: Top 10 most frequent dependency paths when the target words are NOUN.

BERT	BiLSTM
NOUN =(down:amod)⇒ ADJ	NOUN =(down:amod)⇒ ADJ
DET =(up:det)⇒ NOUN =(down:amod)⇒ ADJ	DET =(up:det)⇒ NOUN =(down:amod)⇒ ADJ
ADP =(up:case)⇒ NOUN =(down:amod)⇒ ADJ	ADP =(up:case)⇒ NOUN =(down:amod)⇒ ADJ
AUX =(up:cop)⇒ ADJ	AUX =(up:cop)⇒ ADJ
VERB =(down:obj)⇒ NOUN =(down:amod)⇒ ADJ	ADV =(up:advmod)⇒ ADJ
ADV =(up:advmod)⇒ ADJ	VERB =(down:obj)⇒ NOUN =(down:amod)⇒ ADJ
ADJ =(up:amod)⇒ NOUN =(down:amod)⇒ ADJ	ADJ =(up:amod)⇒ NOUN =(down:amod)⇒ ADJ
ADP =(up:case)⇒ NOUN =(up:nmod)⇒ NOUN =(down:amod)⇒ ADJ	PUNCT =(up:punct)⇒ ADJ
PUNCT =(up:punct)⇒ NOUN =(down:amod)⇒ ADJ	VERB =(down:obl)⇒ NOUN =(down:amod)⇒ ADJ
PUNCT =(up:punct)⇒ ADJ	PUNCT =(up:punct)⇒ NOUN =(down:amod)⇒ ADJ

Table 6: Top 10 most frequent dependency paths when the target words are ADJ.

BERT	BiLSTM
PRON =(up:nsubj)⇒ VERB	PRON =(up:nsubj)⇒ VERB
NOUN =(up:obj)⇒ VERB	NOUN =(up:obj)⇒ VERB
PUNCT =(up:punct)⇒ VERB	PUNCT =(up:punct)⇒ VERB
AUX =(up:aux)⇒ VERB	AUX =(up:aux)⇒ VERB
ADV =(up:advmod)⇒ VERB	ADV =(up:advmod)⇒ VERB
ADP =(up:case)⇒ NOUN =(up:obl)⇒ VERB	ADP =(up:case)⇒ NOUN =(up:obl)⇒ VERB
NOUN =(up:obl)⇒ VERB	NOUN =(up:obl)⇒ VERB
PART =(up:mark)⇒ VERB	PART =(up:mark)⇒ VERB
DET =(up:det)⇒ NOUN =(up:obj)⇒ VERB	DET =(up:det)⇒ NOUN =(up:obj)⇒ VERB
SCONJ =(up:mark)⇒ VERB	SCONJ =(up:mark)⇒ VERB

Table 7: Top 10 most frequent dependency paths when the target words are VERB.

BERT	BiLSTM
NOUN =(down:det)⇒ DET	NOUN =(down:det)⇒ DET
ADP =(up:case)⇒ NOUN =(down:det)⇒ DET	ADP =(up:case)⇒ NOUN =(down:det)⇒ DET
ADJ =(up:amod)⇒ NOUN =(down:det)⇒ DET	ADJ =(up:amod)⇒ NOUN =(down:det)⇒ DET
VERB =(down:obj)⇒ NOUN =(down:det)⇒ DET	VERB =(down:obj)⇒ NOUN =(down:det)⇒ DET
ADP =(up:case)⇒ NOUN =(up:nmod)⇒ NOUN =(down:det)⇒ DET	ADP =(up:case)⇒ NOUN =(up:nmod)⇒ NOUN =(down:det)⇒ DET
VERB =(down:obl)⇒ NOUN =(down:det)⇒ DET	VERB =(down:obl)⇒ NOUN =(down:det)⇒ DET
NOUN =(up:compound)⇒ NOUN =(down:det)⇒ DET	NOUN =(down:nmod)⇒ NOUN =(down:det)⇒ DET
PROPN =(down:det)⇒ DET	NOUN =(up:compound)⇒ NOUN =(down:det)⇒ DET
NOUN =(down:nmod)⇒ NOUN =(down:det)⇒ DET	PROPN =(down:det)⇒ DET
NOUN =(up:nmod)⇒ NOUN =(down:det)⇒ DET	NOUN =(up:nmod)⇒ NOUN =(down:det)⇒ DET

Table 8: Top 10 most frequent dependency paths when the target words are DET.

BERT	BiLSTM
NOUN =(down:case)⇒ ADP	NOUN =(down:case)⇒ ADP
DET =(up:det)⇒ NOUN =(down:case)⇒ ADP	DET =(up:det)⇒ NOUN =(down:case)⇒ ADP
VERB =(down:obl)⇒ NOUN =(down:case)⇒ ADP	VERB =(down:obl)⇒ NOUN =(down:case)⇒ ADP
NOUN =(down:nmod)⇒ NOUN =(down:case)⇒ ADP	NOUN =(down:nmod)⇒ NOUN =(down:case)⇒ ADP
PROPN =(down:case)⇒ ADP	PROPN =(down:case)⇒ ADP
ADJ =(up:amod)⇒ NOUN =(down:case)⇒ ADP	ADJ =(up:amod)⇒ NOUN =(down:case)⇒ ADP
DET =(up:det)⇒ NOUN =(down:nmod)⇒ NOUN =(down:case)⇒ ADP	DET =(up:det)⇒ NOUN =(down:nmod)⇒ NOUN =(down:case)⇒ ADP
PUNCT =(up:punct)⇒ VERB =(down:obl)⇒ NOUN =(down:case)⇒ ADP	PRON =(up:nmod:poss)⇒ NOUN =(down:case)⇒ ADP
NOUN =(down:nmod)⇒ PROPN =(down:case)⇒ ADP	NOUN =(down:nmod)⇒ PROPN =(down:case)⇒ ADP
PRON =(up:nmod:poss)⇒ NOUN =(down:case)⇒ ADP	PRON =(down:case)⇒ ADP

Table 9: Top 10 most frequent dependency paths when the target words are ADP.

essential words which are unique to each model, e.g., words essential to BERT but not essential to BiLSTM. Our goal is to see for these essential words unique to a model, whether some special dependency paths are captured by the model. Table 10, 11, 12, 13, 14 show the results for NOUN, ADJ, VERB, DET, and ADP, respectively. We observe that around top 5 dependency paths for essential words unique to BERT or BiLSTM are mostly overlapping with each other as well as the results in Table 5, 6, 7, 8, 9. This implies that sometimes words of common dependency paths can be identified by BERT as essential while BiLSTM fails to do so and sometimes it's another way around. In other words, there is a room to make models to be more consistently aware of syntactic structures of an input. The observation suggests that explicitly incorporating dependency relations into pre-training could potentially benefit masked language models.

BERT	BiLSTM
DET =(up:det)⇒ NOUN	ADP =(up:case)⇒ NOUN
ADP =(up:case)⇒ NOUN	DET =(up:det)⇒ NOUN
ADJ =(up:amod)⇒ NOUN	VERB =(down:obl)⇒ NOUN
PUNCT =(up:punct)⇒ NOUN	VERB =(down:obj)⇒ NOUN
VERB =(down:obl)⇒ NOUN	PUNCT =(up:punct)⇒ NOUN
VERB =(down:obj)⇒ NOUN	NOUN =(up:nmod)⇒ NOUN
ADP =(up:case)⇒ NOUN =(up:nmod)⇒ NOUN	PUNCT =(up:punct)⇒ VERB =(down:obj)⇒ NOUN
NOUN =(up:nmod)⇒ NOUN	NOUN =(down:nmod)⇒ NOUN
NOUN =(down:nmod)⇒ NOUN	PUNCT =(up:punct)⇒ VERB =(down:obl)⇒ NOUN
NOUN =(up:compound)⇒ NOUN	PRON =(up:nsubj)⇒ VERB =(down:obj)⇒ NOUN

Table 10: Top 10 dependency paths from essential words unique to each model to the target words that are NOUN.

BERT	BiLSTM
NOUN =(down:amod)⇒ ADJ	NOUN =(down:amod)⇒ ADJ
DET =(up:det)⇒ NOUN =(down:amod)⇒ ADJ	PUNCT =(up:punct)⇒ ADJ
ADP =(up:case)⇒ NOUN =(down:amod)⇒ ADJ	ADP =(up:case)⇒ NOUN =(down:amod)⇒ ADJ
VERB =(down:obj)⇒ NOUN =(down:amod)⇒ ADJ	VERB =(down:obl)⇒ NOUN =(down:amod)⇒ ADJ
PUNCT =(up:punct)⇒ NOUN =(down:amod)⇒ ADJ	PUNCT =(up:punct)⇒ NOUN =(down:amod)⇒ ADJ
AUX =(up:cop)⇒ ADJ	NOUN =(up:nmod)⇒ NOUN =(down:amod)⇒ ADJ
ADJ =(up:amod)⇒ NOUN =(down:amod)⇒ ADJ	DET =(up:det)⇒ NOUN =(down:amod)⇒ ADJ
PUNCT =(up:punct)⇒ ADJ	VERB =(down:obj)⇒ NOUN =(down:amod)⇒ ADJ
ADP =(up:case)⇒ NOUN =(up:nmod)⇒ NOUN =(down:amod)⇒ ADJ	PUNCT =(up:punct)⇒ VERB =(down:obj)⇒ NOUN =(down:amod)⇒ ADJ
VERB =(down:obl)⇒ NOUN =(down:amod)⇒ ADJ	PRON =(up:nsubj)⇒ ADJ

Table 11: Top 10 dependency paths from essential words unique to each model to the target words that are ADJ.

BERT	BiLSTM
PUNCT =(up:punct)⇒ VERB	PUNCT =(up:punct)⇒ VERB
ADP =(up:case)⇒ NOUN =(up:obl)⇒ VERB	NOUN =(up:obl)⇒ VERB
NOUN =(up:obj)⇒ VERB	NOUN =(up:obj)⇒ VERB
NOUN =(up:obl)⇒ VERB	PRON =(up:nsubj)⇒ VERB
DET =(up:det)⇒ NOUN =(up:obj)⇒ VERB	DET =(up:det)⇒ NOUN =(up:obl)⇒ VERB
PRON =(up:nsubj)⇒ VERB	VERB =(up:advcl)⇒ VERB
ADV =(up:advmod)⇒ VERB	ADP =(up:case)⇒ NOUN =(up:obl)⇒ VERB
NOUN =(up:nsubj)⇒ VERB	VERB =(down:advcl)⇒ VERB
CCONJ =(up:cc)⇒ VERB	VERB =(up:conj)⇒ VERB
SCONJ =(up:mark)⇒ VERB	VERB =(down:conj)⇒ VERB

Table 12: Top 10 dependency paths from essential words unique to each model to the target words that are VERB.

BERT	BiLSTM
NOUN =(down:det)⇒ DET	NOUN =(down:det)⇒ DET
ADP =(up:case)⇒ NOUN =(down:det)⇒ DET	VERB =(down:obl)⇒ NOUN =(down:det)⇒ DET
VERB =(down:obj)⇒ NOUN =(down:det)⇒ DET	NOUN =(up:nmod)⇒ NOUN =(down:det)⇒ DET
NOUN =(up:nmod)⇒ NOUN =(down:det)⇒ DET	NOUN =(down:nmod)⇒ NOUN =(down:det)⇒ DET
VERB =(down:obl)⇒ NOUN =(down:det)⇒ DET	PUNCT =(up:punct)⇒ VERB =(down:obl)⇒ NOUN =(down:det)⇒ DET
ADJ =(up:amod)⇒ NOUN =(down:det)⇒ DET	PUNCT =(up:punct)⇒ NOUN =(down:det)⇒ DET
ADP =(up:case)⇒ NOUN =(up:nmod)⇒ NOUN =(down:det)⇒ DET	ADP =(up:case)⇒ NOUN =(down:det)⇒ DET
PRON =(up:nsubj)⇒ VERB =(down:obj)⇒ NOUN =(down:det)⇒ DET	ADP =(up:case)⇒ NOUN =(up:nmod)⇒ NOUN =(down:det)⇒ DET
NOUN =(up:compound)⇒ NOUN =(down:det)⇒ DET	VERB =(down:obj)⇒ NOUN =(down:det)⇒ DET
DET =(up:det)⇒ NOUN =(down:nmod)⇒ NOUN =(down:det)⇒ DET	PUNCT =(up:punct)⇒ VERB =(down:obj)⇒ NOUN =(down:det)⇒ DET

Table 13: Top 10 dependency paths from essential words unique to each model to the target words that are DET.

BERT	BiLSTM
NOUN =(down:case)⇒ ADP	NOUN =(down:case)⇒ ADP
DET =(up:det)⇒ NOUN =(down:case)⇒ ADP	VERB =(down:obl)⇒ NOUN =(down:case)⇒ ADP
VERB =(down:obl)⇒ NOUN =(down:case)⇒ ADP	PUNCT =(up:punct)⇒ VERB =(down:obl)⇒ NOUN =(down:case)⇒ ADP
ADJ =(up:amod)⇒ NOUN =(down:case)⇒ ADP	DET =(up:det)⇒ NOUN =(down:case)⇒ ADP
PUNCT =(up:punct)⇒ VERB =(down:obl)⇒ NOUN =(down:case)⇒ ADP	ADJ =(up:amod)⇒ NOUN =(down:case)⇒ ADP
PROPN =(down:case)⇒ ADP	AUX =(up:aux)⇒ VERB =(down:obl)⇒ NOUN =(down:case)⇒ ADP
NOUN =(down:nmod)⇒ NOUN =(down:case)⇒ ADP	PROPN =(down:case)⇒ ADP
DET =(up:det)⇒ NOUN =(down:nmod)⇒ NOUN =(down:case)⇒ ADP	PRON =(up:nsubj)⇒ VERB =(down:obl)⇒ NOUN =(down:case)⇒ ADP
ADP =(up:case)⇒ NOUN =(down:nmod)⇒ NOUN =(down:case)⇒ ADP	NOUN =(up:nmod)⇒ NOUN =(down:case)⇒ ADP
NOUN =(up:compound)⇒ NOUN =(down:case)⇒ ADP	ADP =(up:case)⇒ NOUN =(up:nmod)⇒ NOUN =(down:case)⇒ ADP

Table 14: Top 10 dependency paths from essential words unique to each model to the target words that are ADP.