Sharp Nearby, Fuzzy Far Away: How Neural Language Models Use Context

Urvashi Khandelwal, He He, Peng Qi, Dan Jurafsky
Computer Science Department
Stanford University
{urvashik,hehe,pengqi,jurafsky}@stanford.edu

Abstract

We know very little about how neural language models (LM) use prior linguistic context. In this paper, we investigate the role of context in an LSTM LM, through ablation studies. Specifically, we analyze the increase in perplexity when prior context words are shuffled, replaced, or dropped. On two standard datasets, Penn Treebank and WikiText-2, we find that the model is capable of using about 200 tokens of context on average, but sharply distinguishes nearby context (recent 50 tokens) from the distant history. The model is highly sensitive to the order of words within the most recent sentence, but ignores word order in the long-range context (beyond 50 tokens), suggesting the distant past is modeled only as a rough semantic field or topic. We further find that the neural caching model (Grave et al., 2017b) especially helps the LSTM to copy words from within this distant context. Overall, our analysis not only provides a better understanding of how neural LMs use their context, but also sheds light on recent success from cache-based models.

1 Introduction

Language models are an important component of natural language generation tasks, such as machine translation and summarization. They use context (a sequence of words) to estimate a probability distribution of the upcoming word. For several years now, neural language models (NLMs) (Graves, 2013; Jozefowicz et al., 2016; Grave et al., 2017a; Dauphin et al., 2017; Melis et al., 2018; Yang et al., 2018) have consistently outperformed classical n-gram models, an improvement often attributed to their ability to model long-range dependencies in faraway context. Yet, how these NLMs use the context is largely unexplained.

Recent studies have begun to shed light on the information encoded by Long Short-Term Memory (LSTM) networks. They can remember sentence lengths, word identity, and word order (Adi et al., 2017), can capture some syntactic structures such as subject-verb agreement (Linzen et al., 2016), and can model certain kinds of semantic compositionality such as negation and intensification (Li et al., 2016).

However, all of the previous work studies LSTMs at the sentence level, even though they can potentially encode longer context. Our goal is to complement the prior work to provide a richer understanding of the role of context, in particular, long-range context beyond a sentence. We aim to answer the following questions: (i) How much context is used by NLMs, in terms of the number of tokens? (ii) Within this range, are nearby and long-range contexts represented differently? (iii) How do copy mechanisms help the model use different regions of context?

We investigate these questions via ablation studies on a standard LSTM language model (Merity et al., 2018) on two benchmark language modeling datasets: Penn Treebank and WikiText-2. Given a pretrained language model, we perturb the prior context in various ways *at test time*, to study how much the perturbed information affects model performance. Specifically, we alter the context length to study how many tokens are used, permute tokens to see if LSTMs care about word order in both local and global contexts, and drop and replace target words to test the copying abilities of LSTMs with and without an external copy mechanism, such as the neural cache (Grave et al., 2017b). The cache operates by first recording target words and their context representations seen in the history, and then encouraging the model to copy a word from the past when the current context representation matches that word's recorded context vector.

We find that the LSTM is capable of using about 200 tokens of context on average, with no observable differences from changing the hyperparameter settings. Within this context range, word order is only relevant within the 20 most recent tokens or about a sentence. In the long-range context, order has almost no effect on performance, suggesting that the model maintains a high-level, rough semantic representation of faraway words. Finally, we find that LSTMs can regenerate some words seen in the nearby context, but heavily rely on the cache to help them copy words from the long-range context.

2 Language Modeling

Language models assign probabilities to sequences of words. In practice, the probability can be factorized using the chain rule

$$P(w_1, \ldots, w_t) = \prod_{i=1}^{t} P(w_i | w_{i-1}, \ldots, w_1),$$

and language models compute the conditional probability of a *target word* w_t given its preceding context, $w_1, \ldots, w_{t-1}$.

Language models are trained to minimize the negative log likelihood of the training corpus:

$$\text{NLL} = -\frac{1}{T} \sum_{t=1}^{T} \log P(w_t | w_{t-1}, \ldots, w_1),$$

and the model's performance is usually evaluated by perplexity (PP) on a held-out set:

$$\text{PP} = \exp(\text{NLL}).$$

When testing the effect of ablations, we focus on comparing differences in the language model's losses (NLL) on the dev set, which is equivalent to relative improvements in perplexity.

3 Approach

Our goal is to investigate the effect of contextual features such as the length of context, word order and more, on LSTM performance. Thus, we use ablation analysis, during evaluation, to measure changes in model performance in the absence of certain contextual information.

	PTB		**Wiki**	
	Dev	Test	Dev	Test
# Tokens	73,760	82,430	217,646	245,569
Perplexity (no cache)	59.07	56.89	67.29	64.51
Avg. Sent. Len.	20.9	20.9	23.7	22.6

Table 1: Dataset statistics and performance relevant to our experiments.

Typically, when testing the language model on a held-out sequence of words, all tokens prior to the target word are fed to the model; we call this the *infinite-context* setting. In this study, we observe the change in perplexity or NLL when the model is fed a perturbed context $\delta(w_{t-1}, \ldots, w_1)$, at test time. δ refers to the perturbation function, and we experiment with perturbations such as dropping tokens, shuffling/reversing tokens, and replacing tokens with other words from the vocabulary.[1] It is important to note that we do not train the model with these perturbations. This is because the aim is to start with an LSTM that has been trained in the standard fashion, and discover how much context it uses and which features in nearby vs. long-range context are important. Hence, the mismatch in training and test is a necessary part of experiment design, and all measured losses are upper bounds which would likely be lower, were the model also trained to handle such perturbations.

We use a standard LSTM language model, trained and finetuned using the Averaging SGD optimizer (Merity et al., 2018).[2] We also augment the model with a cache *only* for Section 6.2, in order to investigate why an external copy mechanism is helpful. A short description of the architecture and a detailed list of hyperparameters is listed in Appendix A, and we refer the reader to the original paper for additional details.

We analyze two datasets commonly used for language modeling, Penn Treebank (PTB) (Marcus et al., 1993; Mikolov et al., 2010) and Wikitext-2 (Wiki) (Merity et al., 2017). PTB consists of Wall Street Journal news articles with 0.9M tokens for training and a 10K vocabulary. Wiki is a larger and more diverse dataset, containing Wikipedia articles across many topics with 2.1M tokens for training and a 33K vocabulary. Additional dataset statistics are provided in Ta-

[1]Code for our experiments available at `https://github.com/urvashik/lm-context-analysis`

[2]Public release of their code at `https://github.com/salesforce/awd-lstm-lm`

ble 1.

In this paper, we present results only on the dev sets, in order to avoid revealing details about the test sets. However, we have confirmed that all results are consistent with those on the test sets. In addition, for all experiments we report averaged results from three models trained with different random seeds. Some of the figures provided contain trends from only one of the two datasets and the corresponding figures for the other dataset are provided in Appendix B.

4 How much context is used?

LSTMs are designed to capture long-range dependencies in sequences (Hochreiter and Schmidhuber, 1997). In practice, LSTM language models are provided an infinite amount of prior context, which is as long as the test sequence goes. However, it is unclear how much of this history has a direct impact on model performance. In this section, we investigate how many tokens of context achieve a similar loss (or 1-2% difference in model perplexity) to providing the model infinite context. We consider this the *effective context size*.

LSTM language models have an effective context size of about 200 tokens on average. We determine the effective context size by varying the number of tokens fed to the model. In particular, at test time, we feed the model the most recent n tokens:

$$\delta_{\text{truncate}}(w_{t-1}, \ldots, w_1) = (w_{t-1}, \ldots, w_{t-n}), \quad (1)$$

where $n > 0$ and all tokens farther away from the target w_t are dropped.[3] We compare the dev loss (NLL) from truncated context, to that of the infinite-context setting where all previous words are fed to the model. The resulting increase in loss indicates how important the dropped tokens are for the model.

Figure 1a shows that the difference in dev loss, between truncated- and infinite-context variants of the test setting, gradually diminishes as we increase n from 5 tokens to 1000 tokens. In particular, we only see a 1% increase in perplexity as we move beyond a context of 150 tokens on PTB and 250 tokens on Wiki. Hence, we provide empirical evidence to show that LSTM language models do, in fact, model long-range dependencies, without help from extra context vectors or caches.

Changing hyperparameters does not change the effective context size. NLM performance has been shown to be sensitive to hyperparameters such as the dropout rate and model size (Melis et al., 2018). To investigate if these hyperparameters affect the effective context size as well, we train separate models by varying the following hyperparameters one at a time: (1) number of timesteps for truncated back-propogation (2) dropout rate, (3) model size (hidden state size, number of layers, and word embedding size). In Figure 1b, we show that while different hyperparameter settings result in different perplexities in the infinite-context setting, the trend of how perplexity changes as we reduce the context size remains the same.

4.1 Do different types of words need different amounts of context?

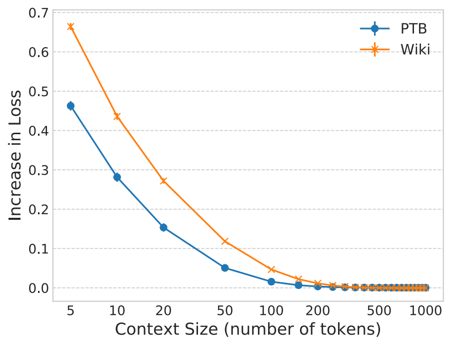
The effective context size determined in the previous section is aggregated over the entire corpus, which ignores the type of the upcoming word. Boyd-Graber and Blei (2009) have previously investigated the differences in context used by different types of words and found that function words rely on less context than content words. We investigate whether the effective context size varies across different types of words, by categorizing them based on either frequency or parts-of-speech. Specifically, we vary the number of context tokens in the same way as the previous section, and aggregate loss over words within each class separately.

Infrequent words need more context than frequent words. We categorize words that appear at least 800 times in the training set as *frequent*, and the rest as *infrequent*. Figure 1c shows that the loss of frequent words is insensitive to missing context beyond the 50 most recent tokens, which holds across the two datasets. Infrequent words, on the other hand, require more than 200 tokens.

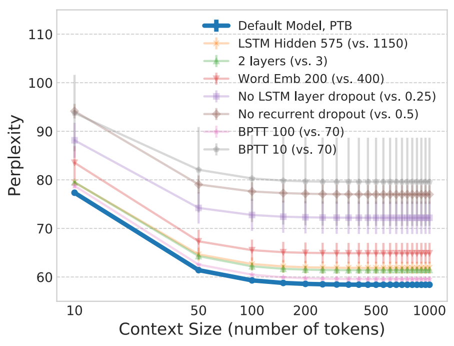
Content words need more context than function words. Given the parts-of-speech of each word, we define *content words* as nouns, verbs and adjectives, and *function words* as prepositions and determiners.[4] Figure 1d shows that the loss of nouns and verbs is affected by distant context, whereas when the target word is a determiner, the model only relies on words within the last 10 tokens.

[3]Words at the beginning of the test sequence with fewer than n tokens in the context are ignored for loss computation.

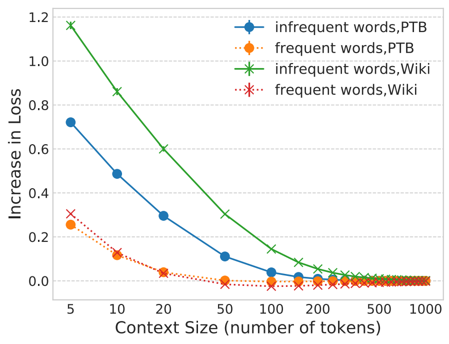
[4]We obtain part-of-speech tags using Stanford CoreNLP (Manning et al., 2014).



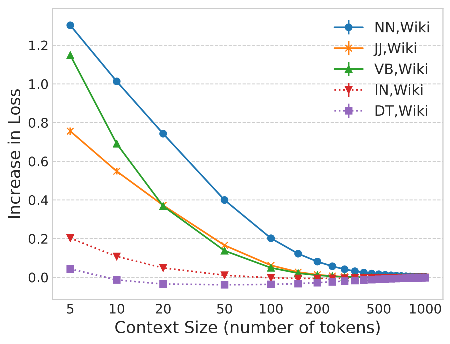
(a) Varying context size.



(b) Changing model hyperparameters.



(c) Frequent vs. infrequent words.



(d) Different parts-of-speech.

Figure 1: Effects of varying the number of tokens provided in the context, as compared to the same model provided with infinite context. Increase in loss represents an absolute increase in NLL over the entire corpus, due to restricted context. All curves are averaged over three random seeds, and error bars represent the standard deviation. **(a)** The model has an effective context size of 150 on PTB and 250 on Wiki. **(b)** Changing model hyperparameters does not change the context usage trend, but does change model performance. We report perplexities to highlight the consistent trend. **(c)** Infrequent words need more context than frequent words. **(d)** Content words need more context than function words.

Discussion. Overall, we find that the model's effective context size is dynamic. It depends on the target word, which is consistent with what we know about language, e.g., determiners require less context than nouns (Boyd-Graber and Blei, 2009). In addition, these findings are consistent with those previously reported for different language models and datasets (Hill et al., 2016; Wang and Cho, 2016).

5 Nearby vs. long-range context

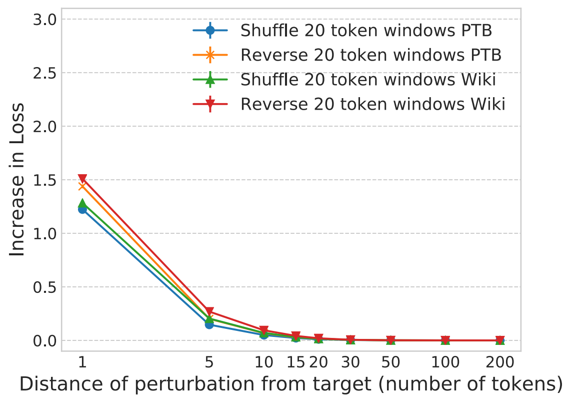
An effective context size of 200 tokens allows for representing linguistic information at many levels of abstraction, such as words, sentences, topics, etc. In this section, we investigate the importance of contextual information such as word order and word identity. Unlike prior work that studies LSTM embeddings at the sentence level, we look at both nearby and faraway context, and analyze how the language model treats contextual information presented in different regions of the context.

5.1 Does word order matter?

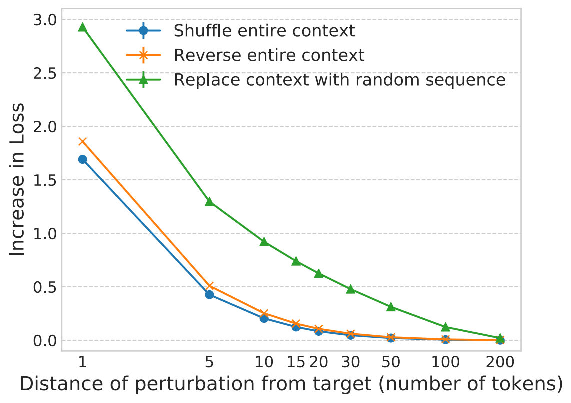
Adi et al. (2017) have shown that LSTMs are aware of word order within a sentence. We investigate whether LSTM language models are sensitive to word order within a larger context window. To determine the range in which word order affects model performance, we permute substrings in the context to observe their effect on dev loss compared to the unperturbed baseline. In particular, we perturb the context as follows,

$$\delta_{\text{permute}}(w_{t-1}, \ldots, w_{t-n}) = (w_{t-1}, .., \rho(w_{t-s_1-1}, .., w_{t-s_2}), .., w_{t-n}) \quad (2)$$

where $\rho \in \{\text{shuffle}, \text{reverse}\}$ and $(s_1, s_2]$ denotes the range of the substring to be permuted. We refer to this substring as the *permutable span*. For



(a) Perturb order locally, within 20 tokens of each point.



(b) Perturb global order, i.e. all tokens in the context before a given point, in Wiki.

Figure 2: Effects of shuffling and reversing the order of words in 300 tokens of context, relative to an unperturbed baseline. All curves are averages from three random seeds, where error bars represent the standard deviation. **(a)** Changing the order of words within a 20-token window has negligible effect on the loss after the first 20 tokens. **(b)** Changing the global order of words within the context does not affect loss beyond 50 tokens.

the following analysis, we distinguish *local word order*, within 20-token permutable spans which are the length of an average sentence, from *global word order*, which extends beyond local spans to include all the farthest tokens in the history. We consider selecting permutable spans within a context of $n = 300$ tokens, which is greater than the effective context size.

Local word order only matters for the most recent 20 tokens. We can locate the region of context beyond which the local word order has no relevance, by permuting word order locally at various points within the context. We accomplish this by varying s_1 and setting $s_2 = s_1 + 20$. Figure 2a shows that local word order matters very much within the most recent 20 tokens, and far less beyond that.

Global order of words only matters for the most recent 50 tokens. Similar to the local word order experiment, we locate the point beyond which the general location of words within the context is irrelevant, by permuting global word order. We achieve this by varying s_1 and fixing $s_2 = n$. Figure 2b demonstrates that after 50 tokens, shuffling or reversing the remaining words in the context has no effect on the model performance.

In order to determine whether this is due to insensitivity to word order or whether the language model is simply not sensitive to any changes in the long-range context, we further replace words in the permutable span with a randomly sampled sequence of the same length from the training set. The gap between the permutation and replacement curves in Figure 2b illustrates that the identity of words in the far away context is still relevant, and only the order of the words is not.

Discussion. These results suggest that word order matters only within the most recent sentence, beyond which the order of sentences matters for 2-3 sentences (determined by our experiments on global word order). After 50 tokens, word order has almost no effect, but the identity of those words is still relevant, suggesting a high-level, rough semantic representation for these faraway words. In light of these observations, we define 50 tokens as the boundary between nearby and long-range context, for the rest of this study. Next, we investigate the importance of different word types in the different regions of context.

5.2 Types of words and the region of context

Open-class or *content words* such as nouns, verbs, adjectives and adverbs, contribute more to the semantic context of natural language than *function words* such as determiners and prepositions. Given our observation that the language model represents long-range context as a rough semantic representation, a natural question to ask is how important are function words in the long-range

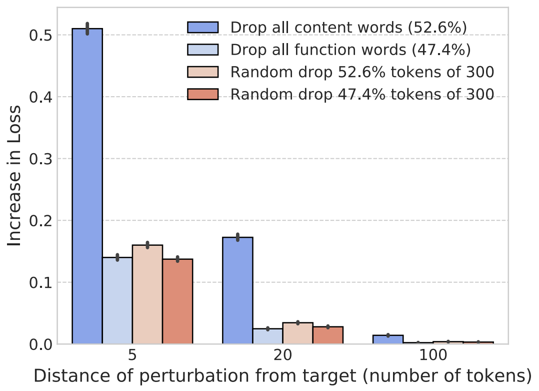


Figure 3: Effect of dropping content and function words from 300 tokens of context relative to an unperturbed baseline, on PTB. Error bars represent 95% confidence intervals. Dropping both content and function words 5 tokens away from the target results in a nontrivial increase in loss, whereas beyond 20 tokens, only content words are relevant.

context? Below, we study the effect of these two classes of words on the model's performance. Function words are defined as all words that are not nouns, verbs, adjectives or adverbs.

Content words matter more than function words. To study the effect of content and function words on model perplexity, we drop them from different regions of the context and compare the resulting change in loss. Specifically, we perturb the context as follows,

$$\delta_{\text{drop}}(w_{t-1}, \ldots, w_{t-n}) = (w_{t-1}, .., w_{t-s_1}, f_{\text{pos}}(y, (w_{t-s_1-1}, .., w_{t-n}))) \quad (3)$$

where $f_{\text{pos}}(y, \text{span})$ is a function that drops all words with POS tag y in a given span. s_1 denotes the starting offset of the perturbed subsequence. For these experiments, we set $s_1 \in \{5, 20, 100\}$. On average, there are slightly more content words than function words in any given text. As shown in Section 4, dropping more words results in higher loss. To eliminate the effect of dropping different fractions of words, for each experiment where we drop a specific word type, we add a control experiment where the same number of tokens are sampled randomly from the context, and dropped.

Figure 3 shows that dropping content words as close as 5 tokens from the target word increases model perplexity by about 65%, whereas dropping the same proportion of tokens at random, results in a much smaller 17% increase. Dropping all function words, on the other hand, is not very different from dropping the same proportion of words at random, but still increases loss by about 15%. This suggests that within the most recent sentence, content words are extremely important but function words are also relevant since they help maintain grammaticality and syntactic structure. On the other hand, beyond a sentence, only content words have a sizeable influence on model performance.

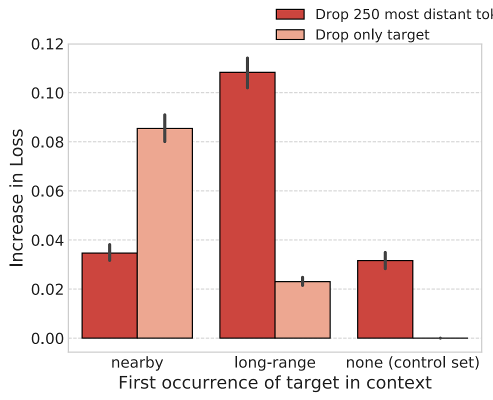
6 To cache or not to cache?

As shown in Section 5.1, LSTM language models use a high-level, rough semantic representation for long-range context, suggesting that they might not be using information from any specific words located far away. Adi et al. (2017) have also shown that while LSTMs are aware of which words appear in their context, this awareness degrades with increasing length of the sequence. However, the success of copy mechanisms such as attention and caching (Bahdanau et al., 2015; Hill et al., 2016; Merity et al., 2017; Grave et al., 2017a,b) suggests that information in the distant context is very useful. Given this fact, can LSTMs copy any words from context without relying on external copy mechanisms? Do they copy words from nearby and long-range context equally? How does the caching model help? In this section, we investigate these questions by studying how LSTMs copy words from different regions of context. More specifically, we look at two regions of context, nearby (within 50 most recent tokens) and long-range (beyond 50 tokens), and study three categories of target words: those that can be copied from nearby context (C_{near}), those that can *only* be copied from long-range context (C_{far}), and those that cannot be copied at all given a limited context (C_{none}).

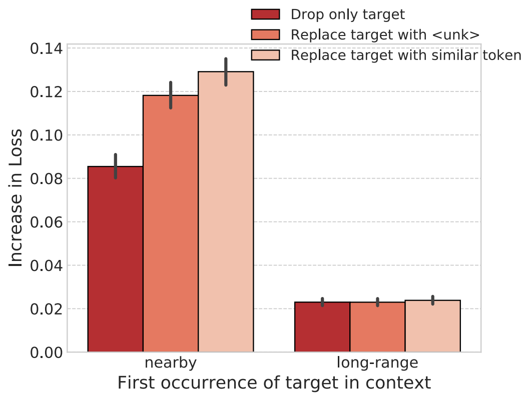
6.1 Can LSTMs copy words without caches?

Even without a cache, LSTMs often regenerate words that have already appeared in prior context. We investigate how much the model relies on the previous occurrences of the upcoming target word, by analyzing the change in loss after dropping and replacing this target word in the context.

LSTMs can regenerate words seen in nearby context. In order to demonstrate the usefulness



(a) Dropping tokens



(b) Perturbing occurrences of target word in context.

Figure 4: Effects of perturbing the target word in the context compared to dropping long-range context altogether, on PTB. Error bars represent 95% confidence intervals. **(a)** Words that can only be copied from long-range context are more sensitive to dropping all the distant words than to dropping the target. For words that can be copied from nearby context, dropping only the target has a much larger effect on loss compared to dropping the long-range context. **(b)** Replacing the target word with other tokens from vocabulary hurts more than dropping it from the context, for words that can be copied from nearby context, but has no effect on words that can only be copied from far away.

of target word occurrences in context, we experiment with dropping all the distant context versus dropping only occurrences of the target word from the context. In particular, we compare removing all tokens after the 50 most recent tokens, (Equation 1 with $n = 50$), versus removing only the target word, in context of size $n = 300$:

$$\delta_{\text{drop}}(w_{t-1}, \ldots, w_{t-n}) = f_{\text{word}}(w_t, (w_{t-1}, \ldots, w_{t-n})), \tag{4}$$

where $f_{\text{word}}(w, \text{span})$ drops words equal to w in a given span. We compare applying both perturbations to a baseline model with unperturbed context restricted to $n = 300$. We also include the target words that never appear in the context (C_{none}) as a control set for this experiment.

The results show that LSTMs rely on the rough semantic representation of the faraway context to generate C_{far}, but direclty copy C_{near} from the nearby context. In Figure 4a, the long-range context bars show that for words that can only be copied from long-range context (C_{far}), removing all distant context is far more disruptive than removing only occurrences of the target word (12% and 2% increase in perplexity, respectively). This suggests that the model relies more on the rough semantic representation of faraway context to predict these C_{far} tokens, rather than directly copying them from the distant context. On the other hand, for words that can be copied from nearby context (C_{near}), removing all long-range context has a smaller effect (about 3.5% increase in perplexity) as seen in Figure 4a, compared to removing the target word which increases perplexity by almost 9%. This suggests that these C_{near} tokens are more often copied from nearby context, than inferred from information found in the rough semantic representation of long-range context.

However, is it possible that dropping the target tokens altogether, hurts the model too much by adversely affecting grammaticality of the context? We test this theory by replacing target words in the context with other words from the vocabulary. This perturbation is similar to Equation 4, except instead of dropping the token, we replace it with a different one. In particular, we experiment with replacing the target with `<unk>`, to see if having the generic word is better than not having any word. We also replace it with a word that has the same part-of-speech tag and a similar frequency in the dataset, to observe how much this change confuses the model. Figure 4b shows that replacing the target with other words results in up to a 14% increase in perplexity for C_{near}, which suggests that the replacement token seems to confuse the model far more than when the token is simply dropped. However, the words that rely on the long-range context, C_{far}, are largely unaffected by these changes, which confirms our conclusion from dropping the target tokens: C_{far}

</s> snack-food UNK increased a strong NUM NUM in the third quarter while domestic profit increased in double UNK mr. calloway said </s> excluding the british snack-food business acquired in july snack-food international UNK jumped NUM NUM with sales strong in spain mexico and brazil </s> total snack-food profit rose NUM NUM </s> led by pizza hut and UNK bell restaurant earnings increased about NUM NUM in the third quarter on a NUM NUM sales increase </s> UNK sales for pizza hut rose about NUM NUM while UNK bell 's increased NUM NUM as the chain continues to benefit from its UNK strategy </s> UNK bell has turned around declining customer counts by permanently lowering the price of its UNK </s> same UNK for kentucky fried chicken which has struggled with increased competition in the fast-food chicken market and a lack of new products rose only NUM NUM </s> the operation which has been slow to respond to consumers ' shifting UNK away from fried foods has been developing a UNK product that may be introduced nationally at the end of next year </s> the new product has performed well in a market test in las vegas nev. mr. **calloway**

Figure 5: Success of neural cache on PTB. Brightly shaded region shows peaky distribution.

offerings outside the u.s. </s> goldman sachs & co. will manage the offering </s> macmillan said berlitz intends to pay quarterly dividends on the stock </s> the company said it expects to pay the first dividend of NUM cents a share in the NUM first quarter </s> berlitz will borrow an amount equal to its expected net proceeds from the offerings plus $ NUM million in connection with a credit agreement with lenders </s> the total borrowing will be about $ NUM million the company said </s> proceeds from the borrowings under the credit agreement will be used to pay an $ NUM million cash dividend to macmillan and to lend the remainder of about $ NUM million to maxwell communications in connection with a UNK note </s> proceeds from the offering will be used to repay borrowings under the short-term parts of a credit agreement </s> berlitz which is based in princeton n.j. provides language instruction and translation services through more than NUM language centers in NUM countries </s> in the past five years more than NUM NUM of its sales have been outside the u.s. </s> macmillan has owned berlitz since NUM </s> in the first six **months**

Figure 6: Failure of neural cache on PTB. Lightly shaded regions show flat distribution.

words are predicted from the rough representation of faraway context instead of specific occurrences of certain words.

6.2 How does the cache help?

If LSTMs can already regenerate words from nearby context, how are copy mechanisms helping the model? We answer this question by analyzing how the neural cache model (Grave et al., 2017b) helps with improving model performance. The cache records the hidden state h_t at each timestep t, and computes a cache distribution over the words in the history as follows:

$$P_{\text{cache}}(w_t | w_{t-1}, \dots, w_1; h_t, \dots, h_1) \propto \sum_{i=1}^{t-1} \mathbb{1}[w_i = w_t] \exp(\theta h_i^T h_t), \tag{5}$$

where θ controls the flatness of the distribution. This cache distribution is then interpolated with the model's output distribution over the vocabulary. Consequently, certain words from the history are upweighted, encouraging the model to copy them.

Caches help words that can be copied from long-range context the most. In order to study the effectiveness of the cache for the three classes of words ($C_{\text{near}}, C_{\text{far}}, C_{\text{none}}$), we evaluate an LSTM language model with and without a cache, and measure the difference in perplexity for these words. In both settings, the model is provided all prior context (not just 300 tokens) in order to replicate the Grave et al. (2017b) setup. The amount of history recorded, known as the cache size, is a hyperparameter set to 500 past timesteps for PTB and 3,875 for Wiki, both values very similar to the average document lengths in the respective datasets.

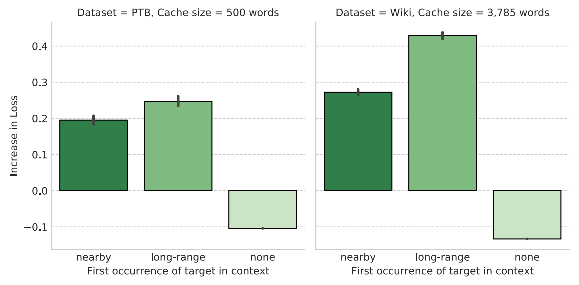


Figure 7: Model performance relative to using a cache. Error bars represent 95% confidence intervals. Words that can only be copied from the distant context benefit the most from using a cache.

We find that the cache helps words that can only be copied from long-range context (C_{far}) more than words that can be copied from nearby (C_{near}). This is illustrated by Figure 7 where without caching, C_{near} words see a 22% increase in perplexity for PTB, and a 32% increase for Wiki, whereas C_{far} see a 28% increase in perplexity for PTB, and a whopping 53% increase for Wiki. Thus, the cache is, in a sense, complementary to the standard model, since it especially helps regenerate words from the long-range context where the latter falls short.

However, the cache also hurts about 36% of the words in PTB and 20% in Wiki, which are words that cannot be copied from context (C_{none}), as illustrated by bars for "none" in Figure 7. We also provide some case studies showing success (Fig. 5) and failure (Fig. 6) modes for the cache. We find that for the successful case, the cache distribution is concentrated on a single word that it wants to copy. However, when the target is not present in the history, the cache distribution is more flat, illustrating the model's confusion, shown in Figure 6. This suggests that the neural cache model might benefit from having the option to ignore the cache when it cannot make a confident choice.

7 Discussion

The findings presented in this paper provide a great deal of insight into how LSTMs model context. This information can prove extremely useful for improving language models. For instance, the discovery that some word types are more important than others can help refine word dropout strategies by making them adaptive to the different word types. Results on the cache also show that we can further improve performance by allowing the model to ignore the cache distribution when it is extremely uncertain, such as in Figure 6. Differences in nearby vs. long-range context suggest that memory models, which feed explicit context representations to the LSTM (Ghosh et al., 2016; Lau et al., 2017), could benefit from representations that specifically capture information orthogonal to that modeled by the LSTM.

In addition, the empirical methods used in this study are model-agnostic and can generalize to models other than the standard LSTM. This opens the path to generating a stronger understanding of model classes beyond test set perplexities, by comparing them across additional axes of information such as how much context they use on average, or how robust they are to shuffled contexts.

Given the empirical nature of this study and the fact that the model and data are tightly coupled, separating model behavior from language characteristics, has proved challenging. More specifically, a number of confounding factors such as vocabulary size, dataset size etc. make this separation difficult. In an attempt to address this, we have chosen PTB and Wiki - two standard language modeling datasets which are diverse in content (news vs. factual articles) and writing style, and are structured differently (eg: Wiki articles are 4-6x longer on average and contain extra information such as titles and paragraph/section markers). Making the data sources diverse in nature, has provided the opportunity to somewhat isolate effects of the model, while ensuring consistency in results. An interesting extension to further study this separation would lie in experimenting with different model classes and even different languages.

Recently, Chelba et al. (2017), in proposing a new model, showed that on PTB, an LSTM language model with 13 tokens of context is similar to the infinite-context LSTM performance, with close to an 8% [5] increase in perplexity. This is compared to a 25% increase at 13 tokens of context in our setup. We believe this difference is attributed to the fact that their model was trained with restricted context and a different error propagation scheme, while ours is not. Further investigation would be an interesting direction for future work.

8 Conclusion

In this analytic study, we have empirically shown that a standard LSTM language model can effectively use about 200 tokens of context on two benchmark datasets, regardless of hyperparameter settings such as model size. It is sensitive to word order in the nearby context, but less so in the long-range context. In addition, the model is able to regenerate words from nearby context, but heavily relies on caches to copy words from far away. These findings not only help us better understand these models but also suggest ways for improving them, as discussed in Section 7. While observations in this paper are reported at the token level, deeper understanding of sentence-level interactions warrants further investigation, which we leave to future work.

Acknowledgments

We thank Arun Chaganty, Kevin Clark, Reid Pryzant, Yuhao Zhang and our anonymous reviewers for their thoughtful comments and suggestions. We gratefully acknowledge support of the DARPA Communicating with Computers (CwC) program under ARO prime contract no. W911NF15-1-0462 and the NSF via grant IIS-1514268.

[5]Table 3, 91 perplexity for the 13-gram vs. 84 for the infinite context model.

References

Yossi Adi, Einat Kermany, Yonatan Belinkov, Ofer Lavi, and Yoav Goldberg. 2017. Fine-grained analysis of sentence embeddings using auxiliary prediction tasks. *International Conference on Learning Representations (ICLR)* https://openreview.net/pdf?id=BJh6Ztuxl.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. *International Conference on Learning Representations (ICLR)* https://arxiv.org/pdf/1409.0473.pdf.

Jordan Boyd-Graber and David Blei. 2009. Syntactic topic models. In *Advances in neural information processing systems*. pages 185–192. https://papers.nips.cc/paper/3398-syntactic-topic-models.pdf.

Ciprian Chelba, Mohammad Norouzi, and Samy Bengio. 2017. N-gram language modeling using recurrent neural network estimation. *arXiv preprint arXiv:1703.10724* https://arxiv.org/pdf/1703.10724.pdf.

Yann N Dauphin, Angela Fan, Michael Auli, and David Grangier. 2017. Language modeling with gated convolutional networks. *International Conference on Machine Learning (ICML)* https://arxiv.org/pdf/1612.08083.pdf.

Yarin Gal and Zoubin Ghahramani. 2016. A theoretically grounded application of dropout in recurrent neural networks. In *Advances in neural information processing systems (NIPS)*. pages 1019–1027. https://arxiv.org/pdf/1512.05287.pdf.

Shalini Ghosh, Oriol Vinyals, Brian Strope, Scott Roy, Tom Dean, and Larry Heck. 2016. Contextual lstm (clstm) models for large scale nlp tasks. *Workshop on Large-scale Deep Learning for Data Mining, KDD* https://arxiv.org/pdf/1602.06291.pdf.

Edouard Grave, Moustapha M Cisse, and Armand Joulin. 2017a. Unbounded cache model for online language modeling with open vocabulary. In *Advances in Neural Information Processing Systems (NIPS)*. pages 6044–6054. https://papers.nips.cc/paper/7185-unbounded-cache-model-for-online-language-modeling-with-open-vocabulary.pdf.

Edouard Grave, Armand Joulin, and Nicolas Usunier. 2017b. Improving Neural Language Models with a Continuous Cache. *International Conference on Learning Representations (ICLR)* https://openreview.net/pdf?id=B184E5qee.

Alex Graves. 2013. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850* https://arxiv.org/pdf/1308.0850.pdf.

Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. 2016. The goldilocks principle: Reading children's books with explicit memory representations. *International Conference on Learning Representations (ICLR)* https://arxiv.org/pdf/1511.02301.pdf.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9(8):1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735.

Hakan Inan, Khashayar Khosravi, and Richard Socher. 2017. Tying word vectors and word classifiers: A loss framework for language modeling. *International Conference on Learning Representations (ICLR)* https://openreview.net/pdf?id=r1aPbsFle.

Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. 2016. Exploring the limits of language modeling. *arXiv preprint arXiv:1602.02410* https://arxiv.org/pdf/1602.02410.pdf.

Jey Han Lau, Timothy Baldwin, and Trevor Cohn. 2017. Topically Driven Neural Language Model. *Association for Computational Linguistics (ACL)* https://doi.org/10.18653/v1/P17-1033.

Jiwei Li, Xinlei Chen, Eduard Hovy, and Dan Jurafsky. 2016. Visualizing and understanding neural models in nlp. *North American Association of Computational Linguistics (NAACL)* http://www.aclweb.org/anthology/N16-1082.

Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. 2016. Assessing the ability of lstms to learn syntax-sensitive dependencies. *Transactions of the Association for Computational Linguistics (TACL)* http://aclweb.org/anthology/Q16-1037.

Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*. pages 55–60. https://doi.org/10.3115/v1/P14-5010.

Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of english: The penn treebank. *Computational linguistics* 19(2):313–330. http://aclweb.org/anthology/J93-2004.

Gabor Melis, Chris Dyer, and Phil Blunsom. 2018. On the State of the Art of Evaluation in Neural Language Models. *International Conference on Learning Representations (ICLR)* https://openreview.net/pdf?id=ByJHuTgA-.

Stephen Merity, Nitish Shirish Keskar, and Richard Socher. 2018. Regularizing and Optimizing LSTM Language Models. *International Conference on Learning Representations (ICLR)* https://openreview.net/pdf?id=SyyGPP0TZ.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2017. Pointer Sentinel Mixture Models. *International Conference on Learning Representations (ICLR)* https://openreview.net/pdf?id=Byj72udxe.

Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Eleventh Annual Conference of the International Speech Communication Association*.

Ofir Press and Lior Wolf. 2017. Using the output embedding to improve language models. *European Chapter of the Association for Computational Linguistics* http://aclweb.org/anthology/E17-2025.

Li Wan, Matthew Zeiler, Sixin Zhang, Yann Le Cun, and Rob Fergus. 2013. Regularization of neural networks using dropconnect. In *International Conference on Machine Learning (ICML)*. pages 1058–1066.

Tian Wang and Kyunghyun Cho. 2016. Larger-Context Language Modelling with Recurrent Neural Network. *Association for Computational Linguistics (ACL)* https://doi.org/10.18653/v1/P16-1125.

Zhilin Yang, Zihang Dai, Ruslan Salakhutdinov, and William W Cohen. 2018. Breaking the softmax bottleneck: a high-rank rnn language model. *International Conference on Learning Representations (ICLR)* https://openreview.net/pdf?id=HkwZSG-CZ.