DOCUMENT CONTEXT LANGUAGE MODELS

Yangfeng Ji[1], **Trevor Cohn**[2], **Lingpeng Kong**[3], **Chris Dyer**[3] **& Jacob Eisenstein**[1]
[1] School of Interactive Computing, Georgia Institute of Technology
[2] Department of Computing and Information Systems, University of Melbourne
[3] School of Computer Science, Carnegie Mellon University

ABSTRACT

Text documents are structured on multiple levels of detail: individual words are related by syntax, and larger units of text are related by discourse structure. Existing language models generally fail to account for discourse structure, but it is crucial if we are to have language models that reward coherence and generate coherent texts. We present and empirically evaluate a set of multi-level recurrent neural network language models, called Document-Context Language Models (DCLMs), which incorporate contextual information both within and beyond the sentence. In comparison with sentence-level recurrent neural network language models, the DCLMs obtain slightly better predictive likelihoods, and considerably better assessments of document coherence.

1 INTRODUCTION

Statistical language models are essential components of natural language processing systems, such as machine translation (Koehn, 2009), automatic speech recognition (Jurafsky & Martin, 2000), text generation (Sordoni et al., 2015) and information retrieval (Manning et al., 2008). Language models estimate the probability of a word for a given context. In conventional language models, context is represented by n-grams, so these models condition on a fixed number of preceding words. Recurrent Neural Network Language Models (RNNLMs; Mikolov et al., 2010) use a dense vector representation to summarize context across all preceding words within the same sentence. But context operates on multiple levels of detail: on the syntactic level, a word's immediate neighbors are most predictive; but on the level of discourse and topic, all words in the document lend contextual information.

Recent research has developed a variety of ways to incorporate document-level contextual information. For example, both Mikolov & Zweig (2012) and Le & Mikolov (2014) use topic information extracted from the entire document to help predict words in each sentence; Lin et al. (2015) propose to construct contextual information by predicting the bag-of-words representation of the previous sentence with a separate model; Wang & Cho (2015) build a bag-of-words context from the previous sentence and integrate it into the Long Short-Term Memory (LSTM) generating the current sentence. These models are all *hybrid* architectures in that they are recurrent at the sentence level, but use a different architecture to summarize the context outside the sentence.

In this paper, we explore multi-level recurrent architectures for combining local and global information in language modeling. The simplest such model would be to train a single RNN, ignoring sentence boundaries: as shown in Figure 1, the last hidden state from the previous sentence $t-1$ is used to initialize the first hidden state in sentence t. In such an architecture, the length of the RNN is equal to the number of tokens in the document; in typical genres such as news texts, this means training RNNs from sequences of several hundred tokens, which introduces two problems:

Information decay In a sentence with thirty tokens (not unusual in news text), the contextual information from the previous sentence must be propagated through the recurrent dynamics thirty times before it can reach the last token of the current sentence. Meaningful document-level information is unlikely to survive such a long pipeline.

Learning It is notoriously difficult to train recurrent architectures that involve many time steps (Bengio et al., 1994). In the case of an RNN trained on an entire document, back-propagation would have to run over hundreds of steps, posing severe numerical challenges.

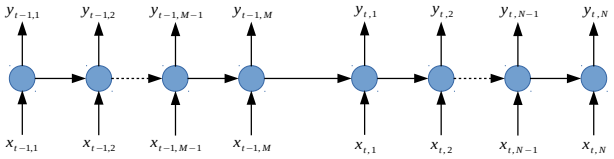


Figure 1: A fragment of document-level recurrent neural network language model (DRNNLM). It is also an extension of sentence-level RNNLM to the document level by ignoring sentence boundaries.

In this paper, we use multi-level recurrent structures to solve both of these problems, thereby successfully efficiently leveraging document-level context in language modeling. We present several variant Document-Context Language Models (DCLMs), and evaluate them on predictive likelihood and their ability to capture document coherence.

2 MODELING FRAMEWORK

The core modeling idea of this work is to integrate contextual information from the RNN language model of the previous sentence into the language model of the current sentence. We present three alternative models, each with various practical or theoretical merits, and then evaluate them in section 4.

2.1 RECURRENT NEURAL NETWORK LANGUAGE MODELS

We start from a recurrent neural network language model (RNNLM) to explain some necessary terms. Given a sentence $\{\boldsymbol{x}_n\}_{n=1}^N$, a recurrent neural network language model is defined as

$$\boldsymbol{h}_n = \boldsymbol{g}\left(\boldsymbol{h}_{n-1}, \boldsymbol{x}_n\right) \tag{1}$$

$$\boldsymbol{y}_n = \text{softmax}\left(\mathbf{W}_o \boldsymbol{h}_n + \boldsymbol{b}\right), \tag{2}$$

where $\boldsymbol{x}_n \in \mathbb{R}^K$ is the distributed representation of the n-th word, $\boldsymbol{h}_n \in \mathbb{R}^H$ is the corresponding hidden state computed from the word representation and the previous hidden state $\boldsymbol{h}_{n-1}$, and $\boldsymbol{b}$ is the bias term. K and H are the input and hidden dimension respectively. As in the original RNNLM (Mikolov et al., 2010), $\boldsymbol{y}_n$ is a prediction of the $(n+1)$-th word in the sequence.

The transition function $\boldsymbol{g}\left(\cdot\right)$ could be any nonlinear function used in neural networks, such as the elementwise sigmoid function, or more complex recurrent functions such as the LSTM (Hochreiter & Schmidhuber, 1997) or GRU (Chung et al., 2014). In this work, we use LSTM, as it consistently gives the best performance in our experiments. By stacking two LSTM together, we are able obtain a even more powerful transition function, called multi-layer LSTM (Sutskever et al., 2014). In a multi-layer LSTM, the hidden state from a lower-layer LSTM cell is used as the input to the upper-layer, and the hidden state from the final-layer is used for prediction. In our following models, we fix the number of layers as two.

In the rest of this section, we will consider different ways to employ the contextual information for document-level language modeling. All models obtain the contextual representation from the hidden states of the previous sentence, but they use this information in different ways.

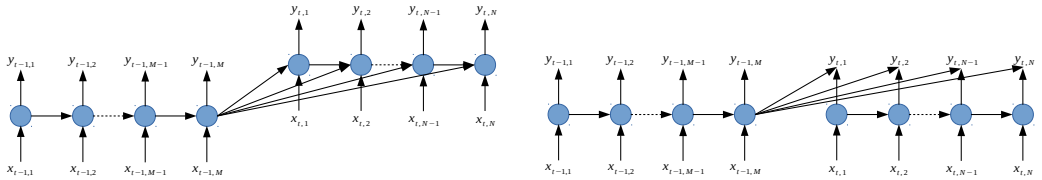
2.2 MODEL I: CONTEXT-TO-CONTEXT DCLM

The underlying assumption of this work is that contextual information from previous sentences needs to be able to "short-circuit" the standard RNN, so as to more directly impact the generation of words across longer spans of text. We first consider the relevant contextual information to be the final hidden representation from the previous sentence $t-1$, so that,

$$\boldsymbol{c}_{t-1} = \boldsymbol{h}_{t-1,M} \tag{3}$$

where M is the length of sentence $t-1$. We then create additional paths for this information to impact each hidden representation in the current sentence t. Writing $\boldsymbol{x}_{t,n}$ for the word representation of the n-th word in the t-th sentence, we have,

$$\boldsymbol{h}_{t,n} = \boldsymbol{g}_{\boldsymbol{\theta}}\left(\boldsymbol{h}_{t,n-1}, \boldsymbol{s}\left(\boldsymbol{x}_{t,n}, \boldsymbol{c}_{t-1}\right)\right) \tag{4}$$



(a) ccDCLM (b) coDCLM

Figure 2: Context-to-context and context-to-output DCLMs

where $\boldsymbol{g_\theta}(\cdot)$ is the activation function parameterized by θ and $\boldsymbol{s}(\cdot)$ is a function that combines the context vector with the input $\boldsymbol{x}_{t,n}$ for the hidden state. In future work we may consider a variety of forms for this function, but here we simply concatenate the representations,

$$\boldsymbol{s}(\boldsymbol{x}_{t,n}, \boldsymbol{c}_{t-1}) = [\boldsymbol{x}_{t,n}, \boldsymbol{c}_{t-1}]. \tag{5}$$

The emission probability for $\boldsymbol{y}_{t,n}$ is then computed from $\boldsymbol{h}_{t,n}$ as in the standard RNNLM (Equation 2). The underlying assumption of this model is that contextual information should impact the generation of each word in the current sentence. The model therefore introduces computational "short-circuits" for cross-sentence information, as illustrated in Figure 2(a). Because information flows from one hidden vector to another, we call this the **context-to-context Document Context Language Model**, abbreviated ccDCLM.

With this specific architecture, the number of parameters is $H(16H + 3K + 6) + V(H + K + 1)$, where H is the size of the hidden representation, K is the size of the word representation, and V is the vocabulary size. The constant factors come with the weight matrices within a two-layer LSTM unit. This is in the same complexity class as the standard RNNLM. Special handling is necessary for the first sentence of the document. Inspired by the idea of sentence-level language modeling, we introduce a *dummy* contextual representation $\boldsymbol{c}_0$ as a START symbol for a document. This is another parameter to be learned jointly with the other parameters in this model.

The training procedure of ccDCLM is similar to a conventional RNNLM: we move from left to right through the document and compute a softmax loss on each output $\boldsymbol{y}_{t,n}$. We then backpropagate this loss through the entire sequences.

2.2.1 Model II: Context-to-Output DCLM

Rather than incorporating the document context into the recurrent definition of the hidden state, we can push it directly to the output, as illustrated in Figure 2(b). Let $\boldsymbol{h}_{t,n}$ be the hidden state from a conventional RNNLM of sentence t,

$$\boldsymbol{h}_{t,n} = \boldsymbol{g}_\theta(\boldsymbol{h}_{t,n-1}, \boldsymbol{x}_{t,n}). \tag{6}$$

Then, the context vector $\boldsymbol{c}_{t-1}$ is directly used in the output layer as

$$\boldsymbol{y}_{t,n} \sim \text{softmax}(\mathbf{W}_h \boldsymbol{h}_{t,n} + \mathbf{W}_c \boldsymbol{c}_{t-1} + \boldsymbol{b}) \tag{7}$$

where $\boldsymbol{c}_{t-1}$ is defined in Equation 3. Because the document context impacts the output directly, we call this model the **context-to-output DCLM** (coDCLM). The modification on the model architecture from ccDCLM to coDCLM leads to a notable change on the number of parameters. The total number of parameters of coDCLM is $H(13H + 3K + 6) + V(2H + K + 1)$. The difference of the parameter numbers between these coDCLM and ccDCLM is $VH - 3H^2$. Recall that V is the vocabulary size and H is the size of latent representation, in most cases we have $V \geq 10^4$ and $H \approx 10^2$. Therefore $V \gg H$ in all reasonable cases, and coDCLM includes more parameters than ccDCLM in general.

While the coDCLM has more parameters that must be learned, it has a potentially important computational advantage. By shifting $\boldsymbol{c}_{t-1}$ from hidden layer to output layer, the relationship of any two hidden vectors $\boldsymbol{h}_t$ and $\boldsymbol{h}_{t'}$ from different sentences is decoupled, so that each can be computed in isolation. In a guided language generation scenario such as machine translation or speech recognition — the most common use case of neural language models — this means that decoding decisions are only *pairwise dependent* across sentences. This is in contrast with the ccDCLM, where the

tying between each $\boldsymbol{h}_t$ and $\boldsymbol{h}_{t+1}$ means that decoding decisions are *jointly dependent* across the entire document. This joint dependence may have important advantages, as it propagates contextual information further across the document; the CCDCLM and CODCLM thereby offer two points on a tradeoff between accuracy and decoding complexity.

2.3 ATTENTIONAL DCLM

One potential shortcoming of CCDCLM and CODCLM is the limited capacity of the context vector, $\boldsymbol{c}_{t-1}$, which is a fixed dimensional representation of the context. While this might suffice for short sentences, as sentences grow longer, the amount of information needing to be carried forward will also grow, and therefore a fixed size embedding may be insufficient. For this reason, we now consider an *attentional* mechanism, based on conditional language models for translation (Sutskever et al., 2014; Bahdanau et al., 2015) which allows for a dynamic capacity representation of the context.

Central to the attentional mechanism is the context representation, which is defined separately for each word position in the output sentence,

$$\boldsymbol{c}_{t-1,n} = \sum_{m=1}^{M} \alpha_{n,m} \boldsymbol{h}_{t-1,m} \tag{8}$$

$$\boldsymbol{\alpha}_n = \text{softmax}\left(\boldsymbol{a}_n\right) \tag{9}$$

$$a_{n,m} = \boldsymbol{w}_a^\top \tanh\left(\mathbf{W}_{a1}\boldsymbol{h}_{t,n} + \mathbf{W}_{a2}\boldsymbol{h}_{t-1,m}\right) \tag{10}$$

where $\boldsymbol{c}_{t-1,n}$ is formulated as a weighted linear combination of all the hidden states in the previous sentence, with weights α constrained to lie on the simplex using the softmax transformation. Each weight $\alpha_{n,m}$ encodes the importance of the context at position m for generating the current word at n, defined as a neural network with a hidden layer and a single scalar output. Consequently each position in the generated output can 'attend' to different elements of the context sentence, which would arguably be useful to shift the focus to make best use of the context vector during generation.

The revised definition of the context in Equation 8 requires some minor changes in the generating components. We include this as an additional input to both the recurrent function (similar to CCDCLM), and output generating function (akin to CODCLM), as follows

$$\boldsymbol{h}_{t,n} = \boldsymbol{g}_{\boldsymbol{\theta}}\left(\boldsymbol{h}_{t,n-1}, \left[\boldsymbol{c}_{t-1,n}^\top, \boldsymbol{x}_{t,n}^\top\right]^\top\right) \tag{11}$$

$$\boldsymbol{y}_{t,n} \sim \text{softmax}\left(\mathbf{W}_o \tanh\left(\mathbf{W}_h\boldsymbol{h}_{t,n} + \mathbf{W}_c\boldsymbol{c}_{t-1,n} + \boldsymbol{b}\right)\right) \tag{12}$$

where the output uses a single hidden layer network to merge the local state and context, before expanding the dimensionality to the size of the output vocabulary, using $\mathbf{W}_o$. The extended model is named as **attentional DCLM** (ADCLM).

3 DATA AND IMPLEMENTATION

We evaluate our models with perplexity and document-level coherence assessment. The first data set used for evaluation is the Penn Treebank (PTB) corpus (Marcus et al., 1993), which is a standard data set used for evaluating language models (e.g., Mikolov et al., 2010). We use the standard split: sections 0-20 for training, 21-22 for development, and 23-24 for test. Following prior work (e.g., Mikolov et al., 2010), we keep the top 10,000 words to construct the vocabulary, and replace lower frequency words with the special token UNKNOWN. The vocabulary also includes two special tokens START and END to indicate the beginning and end of a sentence. In total, the vocabulary size is $10,003$.

To investigate the capacity of modeling documents with larger context, we use a subset of the North American News Text (NANT) corpus (McClosky et al., 2008) to construct another evaluation data set. As shown in Table 1, the average length of the training documents is more than 30 sentences. We follow the same procedure to preprocess the dataset as for the PTB corpus, and keep the top 15,000 words from the training set in the vocabulary. Some basic statistics of both data sets are listed in Table 1.

			Average Document Length	
		# Documents	# Tokens	# Sentences
PTB	Training	2,000	502	21
	Development	155	516	22
	Test	155	577	24
NANT	Training	26,462	783	32
	Development	148	799	33
	Test	2,753	778	32

Table 1: Basic statistics of the Penn Treebank (PTB) and North American News Text (NANT) data sets

3.1 IMPLEMENTATION

We use a two-layer LSTM to build the recurrent architecture of our document language models, which we implement in the CNN package (`https://github.com/clab/cnn`). The rest of this section includes some additional details of our implementation, which is available online at `https://github.com/jiyfeng/dclm`.

Initialization All parameters are initialized with random values drawn from the range $[-\sqrt{6/(d_1+d_2)}, \sqrt{6/(d_1+d_2)}]$, where d_1 and d_2 are the input and output dimensions of the parameter matrix respectively, as suggested by Glorot & Bengio (2010).

Learning Online learning was performed using AdaGrad (Duchi et al., 2011) with the initial learning $\lambda = 0.1$. To avoid the exploding gradient problem, we used the norm clipping trick proposed by Pascanu et al. (2012) and fixed the norm threshold as $\tau = 5.0$.

Hyper-parameters Our models include two tunable hyper-parameters: the dimension of word representation K and the hidden dimension of LSTM unit H. We consider the values $\{32, 48, 64, 96, 128, 256\}$ for both K and H. The best combination of K and H for each model is selected by the development sets via grid search. In all experiments, we fix the hidden dimension of the attentional component in ADCLM as 48.

Document length As shown in Table 1, the average length of documents is more than 500 tokens, with extreme cases having over 1,000 tokens. In practice, we noticed that training on long documents leads to a very slow convergence. We therefore segment documents into several non-overlapping shorter documents, each with at most L sentences, while preserving the original sentence order. The value of L used in most experiments is 5, although we compare with $L = 10$ in subsection 4.1.

4 EXPERIMENTS

We compare the three DCLM-style models (CCDCLM, CODCLM, ADCLM) with the following competitive alternatives:

Recurrent neural network language model (RNNLM) The model is trained on individual sentences without any contextual information (Mikolov et al., 2010). The comparison between our models and this baseline system highlights the contribution of contextual information.

RNNLM w/o sentence boundary (DRNNLM) This is a straightforward extension of sentence-level RNNLM to document-level, as illustrated in Figure 1. It can also be viewed a conventional RNNLM without considering sentence boundaries. The difference between RNNLM and DRNNLM is that DRNNLM is able to consider (a limited amount of) extra-sentential context.

Hierarchical RNNLM (HRNNLM) We also adopt the model architecture of HRNNLM (Lin et al., 2015) as another baseline system, and reimplemented it with several modifications for a fair comparison. Comparing to the original implementation (Lin et al., 2015), we first replace the sigmoid recurrence function with a long short-term memory (LSTM) as used in DCLMs. Furthermore, instead of using pretrained word embedding, we update word representation during training. Finally, we jointly train the language models on both sentence-level and document-level. These changes resulted in substantial improvements over the

Model	PTB		NANT	
	Dev	Test	Dev	Test
Baselines				
1. RNNLM (Mikolov et al., 2010)	69.24	71.88	109.48	194.43
2. RNNLM w/o sentence boundary (dRNNLM)	65.27	69.37	101.42	181.62
3. Hierarchical RNNLM (hRNNLM) (Lin et al., 2015)	66.32	70.62	103.90	175.92
Our models				
4. Attentional DCLM (aDCLM)	64.31	68.32	96.47	**170.99**
5. Context-to-output DCLM (coDCLM)	64.37	68.49	**95.10**	173.52
6. Context-to-context DCLM (ccDCLM)	**62.34**	**66.42**	96.77	172.88

Table 2: Perplexities of the Penn Treebank (PTB) and North American News Text (NANT) data sets.

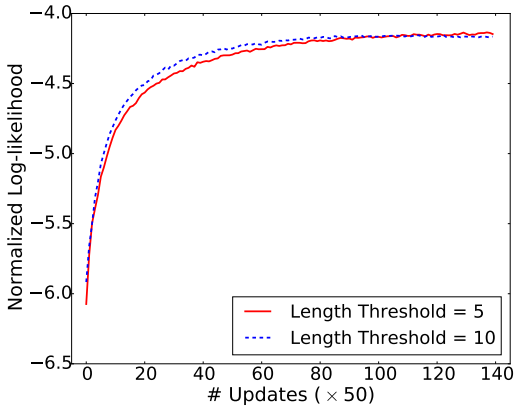


Figure 3: Effect of length thresholds on predictive log-likelihood on the PDTB development set.

original version of the hRNNLM; they allow us to isolate what we view as the most substantive difference between the DCLM and this modeling approach, which is how contextual information is identified and exploited.

4.1 Perplexity

To make a fair comparison across different models, we follow the conventional way to compute perplexity. Particularly, the START and END tokens are only used for notational convenience. The END token from the previous sentence was never used to predict the START token in the current sentence. Therefore, we have the same computation procedure on perplexity for the models with and without contextual information.

Table 2 present the results on language modeling perplexity. The best perplexities are given by the context-to-context DCLM on the PTB data set (line 6 in Table 2), and attentional DCLM on the NANT data set (line 4 in Table 2). All DCLM-based models achieve better perplexity than the prior work. While the improvements on the PTB dataset are small in an absolute sense, they consistently point to the value of including multi-level context information in language modeling. The value of context information is further verified by the model performance on the NANT dataset. Of interest is the behavior of the attentional DCLM on two data sets. This model combines both the context-to-context and context-to-output mechanisms. Theoretically, aDCLMis considerably more expressive then the coDCLM and ccDCLM. On the other hand, it is also complex to learn and innately favors large data sets.

In all the results reported in Table 2, the document length threshold was fixed as $L = 5$, meaning that documents were partitioned into subsequences of five sentences. We were interested to know whether our results depended on this parameter. Taking $L = 1$ would be identical to the standard RNNLM, run separately on each sentence. To test the effect of increasing L, we also did an empirical

comparison between $L = 5$ and $L = 10$ with CCDCLM. Figure 3 shows the two curves on the PTB development set. The x-axis is the number of updates on CCDCLM with the PTB training set. The y-axis is the mean per-token log-likelihood given by Equation 2 on the development set. As shown in this figure, $L = 10$ seems to learn more quickly per iteration in the beginning, although each iteration is more time-consuming, due to the need to backpropagate over longer documents. However, after a sufficient number of updates, the final performance results are nearly identical, with a slight advantage to the $L = 5$ setting. This suggests a tradeoff between the amount of contextual information and the ease of learning.

4.2 LOCAL COHERENCE EVALUATION

The long-term goal of coherence evaluation is to predict which texts are more coherent, and then to optimize for this criterion in multi-sentence generation tasks such as summarization and machine translation. A well-known proxy to this task is to try to automatically distinguish an original document from an alternative form in which the sentences are scrambled (Barzilay & Lapata, 2008; Li & Hovy, 2014). Multi-sentence language models can be applied to this task directly, by determining whether the original document has a higher likelihood; no supervised training is necessary.

We adopt the specific experimental setup proposed by Barzilay & Lapata (2008). To give a robust model comparison with the limited number of documents available in the PTB test set, we employ bootstrapping (Davison & Hinkley, 1997). First, a new test set $\mathcal{D}^{(\ell)}$ is generated by sampling the documents from the original test set with replacement. Then, we shuffled the sentences in each document $d \in \mathcal{D}^{(\ell)}$ to get a pseudo-document d'. The combination of d and d' is a single test example. We repeated the same procedure to produce 1,000 test sets, where each test set includes 155 pairs — one for each document in the PTB test set. Since each test instance is a pairwise choice, a random baseline will have expected accuracy of 50%.

To evaluate the models proposed in this paper, we use the configuration with the best development set perplexity, as shown in Table 2. The results of accuracy and standard deviation are calculated over 1,000 resampled test sets. As shown in Table 3, the best accuracy is 83.26% given by CCDCLM, which also gives the smallest standard deviation 3.77%. Furthermore, all DCLM-based models significantly outperform the RNNLM with $p < 0.01$ given by a two-sample one-side z-test on the bootstrap samples. In addition, the CCDCLM and CODCLM are outperform the HRNNLM with $p < 0.01$ with statistic $z = 36.55$ and 31.26 respectively.

In addition, we also evaluated the models trained on the NANT dataset on this coherence evaluation task. With the same 1,000 test sets, the best accuracy number across different models is 72.85% obtained from CODCLM. Compare to the results in Table 3, we believe the performance drop is due to the domain mismatch. Even though PTB and NANT are both corpora collecting from news articles, they have totally different distributions on words, sentence lengths and even document lengths as shown in Table 1.

Unlike some prior work on coherence evaluation (Li & Hovy, 2014; Li et al., 2015; Lin et al., 2015), our approach is not trained on supervised data. Supervised training might therefore improve performance further. However, we emphasize that the real goal is to make automatically-generated translations and summaries more coherent, and we should therefore avoid overfitting on this artificial proxy task.

5 RELATED WORK

Neural language models (NLMs) learn the distributed representations of words together with the probability function of word sequences. In the NLM proposed by Bengio et al. (2003), a feed-forward neural network with a single hidden layer was used to calculate the language model probabilities. One limitation of this model is only fixed-length context can be used. Recurrent neural network language models (RNNLMs) avoid this problem by recurrently updating a hidden state (Mikolov et al., 2010), thus enabling them to condition on arbitrarily long histories. In this work, we make a further extension to include more context with a recurrent architecture, by allowing multiple pathways for historical information to affect the current word. A comprehensive review of recurrent neural networks language models is offered by De Mulder et al. (2015).

Model	Accuracy	
	Mean (%)	Standard deviation (%)
Baselines		
1. RNNLM w/o sentence boundary (DRNNLM)	72.54	8.46
2. Hierarchical RNNLM (HRNNLM) (Lin et al., 2015)	75.32	4.42
Our models		
3. Attentional DCLM (ADCLM)†	75.51	4.12
4. Context-to-output DCLM (CODCLM)$^{\dagger *}$	81.72	3.81
5. Context-to-context DCLM (CCDCLM)$^{\dagger *}$	83.26	3.77

† significantly better than DRNNLM with p-value < 0.01
* significantly better than HRNNLM with p-value < 0.01

Table 3: Coherence evaluation on the PTB test set. The reported accuracies are calculated from 1,000 bootstrapping test sets (as explained in text).

Conventional language models, including the models with recurrent structures (Mikolov et al., 2010), limit the context scope within a sentence. This ignores potentially important information from preceding text, for example, the previous sentence. Targeting speech recognition, where contextual information may be especially important, Mikolov & Zweig (2012) introduce the *topic-conditioned* RNNLM, which incorporates a separately-trained latent Dirichlet allocation topic model to capture the broad themes of the preceding text. Our focus here is on discriminatively-trained end-to-end models.

Lin et al. (2015) recently introduced a document-level language model, called hierarchical recurrent neural network language model (HRNNLM). As in our approach, there are two channels of information: a RNN for modeling words in a sentence, and another recurrent model for modeling sentences, based on a bag-of-words representation of each sentence. (Contemporaneously to our paper, Wang & Cho (2015) also construct a bag-of-words representation of previous sentences, which they then insert into a sentence-level LSTM.) Our modeling approach is more unified and compact, employing a single recurrent neural network architecture, but with multiple channels for information to feed forward into the prediction of each word. We also go further than this prior work by exploring an attentional architecture (Bahdanau et al., 2015).

Moving away from the specific problem of language modeling, we briefly consider other approaches for modeling document content. Li & Hovy (2014) propose to use a convolution kernel to summarize sentence-level representations for modeling a document. The model is for coherence evaluation, in which the parameters are learned via supervised training. Related convolutional architectures for document modeling are considered by Denil et al. (2014) and Tang et al. (2015). Encoder-decoder architectures provide an alternative perspective, compressing all the information in a sequence into a single vector, and then attempting to decode the target information from this vector; while this idea has notably applied in machine translation (Cho et al., 2014), it can also be employed for coherence modeling (Li et al., 2015). The hierarchical sequence-to-sequence model of Li et al. (2015) conditions the start word of each sentence on contextual information provided by the encoder, but does not apply this idea to language modeling. Different from the models with hierarchical structures, paragraph vector (Le & Mikolov, 2014) encodes a document to a numeric vector by discarding document structure and only retaining topic information.

6 CONCLUSION

Contextual information beyond the sentence boundary is essential to document-level text generation and coherence evaluation. We propose a set of document-context language models (DCLMs), which provide various approaches to incorporate contextual information from preceding texts. Empirical evaluation with perplexity shows that the DCLMs give better word prediction as language models, in comparison with conventional RNNLMs; performance is also good on unsupervised coherence assessment. Future work includes testing the applicability of these models to downstream applications such as summarization and translation.

Acknowledgments This work was initiated during the 2015 Jelinek Memorial Summer Workshop on Speech and Language Technologies at the University of Washington, Seattle, and was supported by Johns Hopkins University via NSF Grant No IIS 1005411, DARPA LORELEI Contract No HR0011-15-2-0027, and gifts from Google, Microsoft Research, Amazon and Mitsubishi Electric Research Laboratory. It was also supported by a Google Faculty Research award to JE.

REFERENCES

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *ICLR*, 2015.

Regina Barzilay and Mirella Lapata. Modeling local coherence: An entity-based approach. *Computational Linguistics*, 34(1):1–34, 2008.

Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *Neural Networks, IEEE Transactions on*, 5(2):157–166, 1994.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. A neural probabilistic language model. *The Journal of Machine Learning Research*, 3:1137–1155, 2003.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *EMNLP*, 2014.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

Anthony Christopher Davison and David Victor Hinkley. *Bootstrap methods and their application*, volume 1. Cambridge university press, 1997.

Wim De Mulder, Steven Bethard, and Marie-Francine Moens. A survey on the application of recurrent neural networks to statistical language modeling. *Computer Speech & Language*, 30(1): 61–98, 2015.

Misha Denil, Alban Demiraj, Nal Kalchbrenner, Phil Blunsom, and Nando de Freitas. Modelling, Visualising and Summarising Documents with a Single Convolutional Neural Network. *arXiv preprint arXiv:1406.3830*, 2014.

John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, 12:2121–2159, 2011.

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *International conference on artificial intelligence and statistics*, pp. 249–256, 2010.

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

Dan Jurafsky and James H Martin. *Speech and language processing*. Pearson Education India, 2000.

Philipp Koehn. *Statistical Machine Translation*. Cambridge University Press, 2009.

Quoc Le and Tomas Mikolov. Distributed Representations of Sentences and Documents. In *ICML*, 2014.

Jiwei Li and Eduard Hovy. A model of coherence based on distributed sentence representation. In *EMNLP*, 2014.

Jiwei Li, Thang Luong, and Dan Jurafsky. A Hierarchical Neural Autoencoder for Paragraphs and Documents. In *ACL-IJCNLP*, pp. 1106–1115, Beijing, China, July 2015. Association for Computational Linguistics.

Rui Lin, Shujie Liu, Muyun Yang, Mu Li, Ming Zhou, and Sheng Li. Hierarchical Recurrent Neural Network for Document Modeling. In *EMNLP*, pp. 899–907, Lisbon, Portugal, September 2015. Association for Computational Linguistics.

Christopher D Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge university press Cambridge, 2008.

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

David McClosky, Eugene Charniak, and Mark Johnson. BLLIP North American News Text, Complete. *Linguistic Data Consortium*, 2008.

Tomas Mikolov and Geoffrey Zweig. Context dependent recurrent neural network language model. In *SLT*, pp. 234–239, 2012.

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. Recurrent neural network based language model. In *INTERSPEECH*, 2010.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. *arXiv preprint arXiv:1211.5063*, 2012.

Alessandro Sordoni, Michel Galley, Michael Auli, Chris Brockett, Yangfeng Ji, Margaret Mitchell, Jian-Yun Nie, Jianfeng Gao, and Bill Dolan. A neural network approach to context-sensitive generation of conversational responses. In *NAACL*, 2015.

Ilya Sutskever, Oriol Vinyals, and Quoc VV Le. Sequence to sequence learning with neural networks. In *NIPS*, 2014.

Duyu Tang, Bing Qin, and Ting Liu. Document Modeling with Gated Recurrent Neural Network for Sentiment Classification. In *EMNLP*, pp. 1422–1432, Lisbon, Portugal, September 2015. Association for Computational Linguistics.

Tian Wang and Kyunghyun Cho. Larger-Context Language Modelling. *arXiv preprint arXiv:1511.03729*, 2015.