CUE Vectors: Modular Training of Language Models Conditioned on Diverse Contextual Signals

Scott Novotney and **Sreeparna Mukherjee** and **Zeeshan Ahmed** and **Andreas Stolcke**

Amazon Alexa

Seattle, WA, USA

{snovotne,sreepar,ahzee,stolcke}@amazon.com

Abstract

We propose a framework to modularize the training of neural language models that use diverse forms of sentence-external context (including metadata) by eliminating the need to jointly train sentence-external and within-sentence encoders. Our approach, contextual universal embeddings (CUE), trains LMs on one set of context, such as date and author, and adapts to novel metadata types, such as article title, or previous sentence. The model consists of a pretrained neural sentence LM, a BERT-based context encoder, and a masked transformer decoder that estimates LM probabilities using sentence-internal and sentence-external information. When context or metadata are unavailable, our model learns to combine contextual and sentence-internal information using noisy oracle unigram embeddings as a proxy. Real contextual information can be introduced later and used to adapt a small number of parameters that map contextual data into the decoder's embedding space. We validate the CUE framework on a NYTimes text corpus with multiple metadata types, for which the LM perplexity can be lowered from 36.6 to 27.4 by conditioning on context. Bootstrapping a contextual LM with only a subset of the context/metadata during training retains 85% of the achievable gain. Training the model initially with proxy context retains 67% of the perplexity gain after adapting to real context. Furthermore, we can swap one type of pretrained sentence LM for another without retraining the context encoders, by only adapting the decoder model. Overall, we obtain a modular framework that allows incremental, scalable training of context-enhanced LMs.

1 Introduction

Language models (LMs) estimate the prior probabilities of token sequences and are key probabilistic modeling components in a variety of applications, such as speech recognition, machine translation, or software keyboards. When modeling linguistic token sequences, typical LMs model one sentence or utterance at a time, reflecting the fact that the strongest predictors of words are syntactic constraints and semantic associations within the sentence. However, it has long been recognized that *context* beyond the sentence has substantial influence on the word probabilities within a sentence. Context literally means the surrounding text (or preceding text, when predicting words in temporal order), but can also refer to any extra-linguistic information, such as metadata (e.g., authorship, time, location) or associated other modalities (e.g., visual cues associated with a spoken utterance).

There is a large literature on leveraging such contextual information for language modeling, some of which we review below (Section 2). However, including context in language modeling presents major challenges for operational settings, especially when LMs need to be trained and deployed at scale:

- Context data is hard to come by. Many language corpora have no or very limited metadata, or contain unordered sentences that do not provide sequential context.

- Context types are specific to a given source. A newspaper corpus has metadata that is very different from spoken language data.

- Use of context renders models context-specific, and therefore, less universally applicable. With each type of context, a new model, or even model architecture, is required.

- Context modeling requires more parameters, compute complexity and more training data.

All these difficulties lead to context being used sparingly in most practical settings, and only when it yields substantial benefits (such as in using a user's personal contact list in voice dialing).

In this paper, we propose a modular modeling framework for contextual language models, called

contextual universal embeddings (CUE). The fundamental idea is to separate the modeling of (1) sentence-internal LM, (2) context embedding and (3) combination of sentence-internal and contextual information each into their own modules. First of all, we show that this architecture is an effective way to bring context to bear on the LM task, achieving 25% relative perplexity reduction over a sentence-internal model, on a corpus of newspaper articles with rich metadata. More importantly, each module can be trained separately, as opposed to jointly with the other modules. Through experimentation we show that, for the practically important use-cases, training modules separately or incrementally preserves most of the achievable gain from contextual information.

Specifically, we can replace one type of context with another, while only adapting the context encoders to the new context, and retain 85% of the best-case perplexity gain. Maybe more surprisingly, we can train the decoder that combines context and sentence-internal information *without any actual context*, instead using noisy oracle unigram embeddings as a proxy. This recovers 67% of the best-case gain after adapting to real context. (Adapting the context encoders affects much fewer parameters, and takes much less data, than the model overall.)

Finally, we show that context encoders can be frozen and a whole different sentence-LM architecture swapped into the model ensemble. After adapting only the combiner-decoder we obtain perplexity gains close to the optimum that would have been achieved by joint training of combiner and context embedding.

2   Prior Work

Longer text history is the most commonly used context in language models (LMs) (Mikolov and Zweig, 2012; Jaech and Ostendorf, 2018a; Ji et al., 2015; Lin et al., 2015). A naive way to bias a LM over text history is to ignore the sentence boundaries and train the contextual LM as the standard neural LM (Ji et al., 2015). However, recurrent neural networks suffer training difficulties on longer sequences (Bengio et al., 1994) while transformer-style models are effective at incorporating this extra information (Dai et al., 2019; Brown et al., 2020).

Another approach is to summarize context into a single context embedding using a separate model. For example, Mikolov and Zweig (2012) and Le

and Mikolov (2014) use topic information extracted from the context. Mikolov and Zweig (2012) use a pretrained Latent Dirichlet Allocation (LDA) model while Le and Mikolov (2014) use paragraph embeddings learned during LM training. Wang and Cho (2016) on the other hand, use a bag-of-words of whole text or individual sentences in the context to build the context vector. Roh et al. (2020) and Lin et al. (2015) further extend sentence-based contextual models by using hierarchical embedding techniques. This approach learns a representation of the context that is directly used as input to a neural LM.

Other sequence tasks in natural language processing (NLP) also leverage contextual information. Neural machine translation (NMT) capitalizes on the availability of previous sentences on the source and target sides when translating documents (Yun et al., 2020; Sugiyama and Yoshinaga, 2021; Zhang et al., 2018). The only difference in the approaches is how the context is encoded into a representation optimized for NMT.

Automatic speech recognition (ASR) and conversational dialog systems also use contextual information, such as recent advances in shallow or deep fusion of end-to-end neural architectures (Zhao et al., 2019; Williams et al., 2018; Kim and Metze, 2018; Munkhdalai et al., 2021; Jain et al., 2020). Recent papers have also considered biasing LMs with context beyond the previous sentence and incorporate additional signals such as date-time, geolocation or gender (Ma et al., 2018; Diehl Martinez et al., 2021) or application metadata like dialog act or intent (Masumura et al., 2019; Shenoy et al., 2021; Liu and Lane, 2017). Other sources of context used to bias LMs are personalized content (Jaech and Ostendorf, 2018b; Fiorini and Lu, 2018); conversational turn-taking (Xiong et al., 2018); multi-modal sources (Moriya and Jones, 2018); or even user demographics to suggest fashion suggestions (Denk and Peleteiro Ramallo, 2020).

3   Architecture

Our task is to estimate an auto-regressive language model conditioned not only on the previous words in the sentence, $W = w_1, w_2, \ldots, w_n$, but also on several contextual signals,

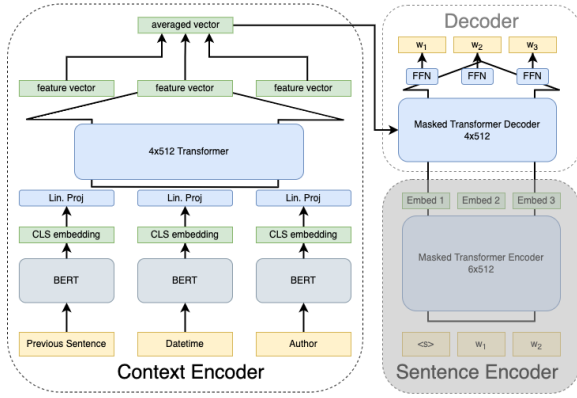$$P(W|C) = \prod_{i=1}^{n} P(w_i | w_{i-1}, w_{i-2}, \ldots, C) \quad (1)$$

Figure 1: *Overview of CUE architecture.* Pretrained sentence encoder and DistilBERT modules are frozen.

where $C = [c_1, ..., c_K]$ represents a set of K contextual inputs that may vary with each sentence, and where each c_k is a sequence of tokens from the same vocabulary as the target sentence.

Adapting recent work in hierarchical contextual embeddings (Yun et al., 2020), our CUE architecture has three components (see Figure 1):

- An auto-regressive transformer **sentence encoder** conditioned on within-sentence history.

- A transformer **context encoder** with a BERT-based embedding combines multiple context signals into one CUE vector.

- An auto-regressive transformer **decoder** to predict the current word based on the context and within-sentence embeddings.

Our goal is to train a separate context encoder that can be updated without modifying the sentence encoder or decoder and run inference independently of the other modules. This is important for operational practicalities, to precompute context embeddings and update them incrementally without requiring downstream components to change.

3.1 Context Encoder

We present all the contextual signals to the context encoder as strings. Non-textual signals, like datetime, are converted into English, such as "Wednesday 29 May 1985". Similarly, we represent any categorical or symbolic context by its English text string.

The encoder projects the set C of K context types into one compact embedding. These may consist of the previous K sentences, K different metadata types such as datetime, or a mix of both.

We then encode each context string c_k with DistilBERT (Sanh et al., 2019) and represent each context by its CLS embedding to generate the intermediate representation

$$g_k = FFN(BERTEncoder(c_k)). \quad (2)$$

We do not fine-tune DistilBERT since empirically it gave negligible gains on our experimental corpora.

The set of intermediate representations $G = [g_1; \ldots; g_K]$ is then passed through transformer blocks to learn dependencies between the context types and to generate the self-attended embeddings $E = [e_1; \ldots; e_K]$, where

$$E = TransformerEncoder(G). \quad (3)$$

The contextual encoder is invariant to the ordering of context types since we treat context as a "bag of sequences" and do not add positional embedding to the CLS embeddings. Additionally, we do not use query values from the within-sentence encoder, so as to preserve the modularity of our architecture; our goal is to use one context encoder with multiple sentence encoders or modeling tasks. The empirical gain was small for conditioning the context attention on the history at each word position (thus giving a different context vector for each token).

Finally, the per-context embeddings e_k are averaged to produce our compact representation,

$$e_{\text{cue}} = \frac{1}{K} \sum_k e_k. \quad (4)$$

3.2 Sentence Encoder

The sentence encoder is a familiar auto-regressive masked language model with a transformer encoder and a final softmax layer to generate a distribution over the vocabulary (Vaswani et al., 2017). We used six layers of 512 dimensions each with 4 attention heads and used a standard language modeling task to fit the parameters; no context was used to train this module. In our experiments, the sentence-encoder parameters are frozen and never fine-tuned when biasing the decoder with context. We assume that the sentence encoder was trained on a very large general text corpus. It uses the same DistilBERT tokenizer as the context encoder, but do not use DistilBERT for word embeddings since our model is causally auto-regressive.

3.3 Decoder

The decoder is a masked transformer decoder as described in Vaswani et al. (2017) with six layers of

Module	# Params	Train	Adapt.
Sentence encoder	24.5M	N	N
DistilBERT	65M	N	N
Decoder	40M	Y	N
Context encoder	6.6M	Y	Y
Total	136M	-	-

Table 1: *CUE components.* Some modules are updated or frozen depending on context training or adaptation.

512 dimensions and 4 heads for multi-headed attention. The sentence-internal embeddings (before the softmax layer) are passed as the shifted outputs to the decoder; along with the contextual CUE vector e_{cue} as input to the multi-headed attention module in the decoder.

4 Adapting to Evolving Context

We now no longer assume that the set of context types is static between training and test. For example, an API providing context may be retired; or business rules improving customer privacy may remove geographic information. The set of contexts may evolve over the life-cycle of our CUE encoder and we now introduce an adaptation step.[1]

Ideally, we would jointly fine-tune the entire model architecture (context encoder, sentence encoder and decoder) on annotated data that contains the new context types. However, this creates an operational burden since different downstream decoders that use context embeddings would each need retraining. Our goal is to adapt the CUE context encoder while leaving the decoder parameters frozen. This will minimize the number of parameters to be retrained and simplify model deployment by factoring the context encoder from the decoder.

We break the training process into two phases: *Training* constructs the initial set of model parameters and is not constrained by operational needs. *Adaptation* happens at some later point in time after the set of context types changes. Section 4.1 considers the scenario where new context types are added to or replace the initial training types. Section 4.2 assumes *no* context is available during model training, only at adaptation time.

4.1 Adapting with annotated data

Our adaptation strategy is to fine-tune only the context encoder, leaving the other parameters unchanged. Since the context encoder consumes sequences of text, our approach benefits from Distil-
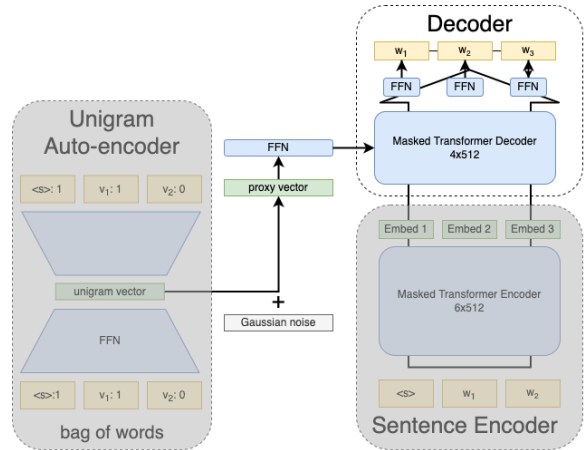

Figure 2: *Priming the decoder with proxy embeddings.* We add noise to an embedding of the target sentence unigram distribution as a proxy for the decoder to learn to attend to context as yet unknown during training. Modules in gray are frozen during decoder training.

BERT projecting sentences into a shared embedding space through the `CLS` token prepended to the beginning of the sentence. We fine-tune the context transformer that operates on the per-context DistilBERT embeddings before averaging (see Figure 1). This component has 6.6M parameters, roughly 5% of the total number of parameters (see Table 1). Given an adaptation corpus of sentences paired with new context types, we take a forward pass for each batch and then backpropagate through the decoder to the contextual encoder. This approach handles any arrangement of new context. Section 6.3 details results evaluating how adaptation benefits over zero-shot approaches with a static model.

4.2 Proxy embeddings

We now consider the scenario where we have no sentences paired with context during training, but want to bias our architecture on context at adaptation time. Since the decoder parameters are frozen during adaptation, we must prime it during training to pay attention to a context embedding, even though we lack the context to generate such an embedding. We tackle this problem by hypothesizing that context plays a role similar to a topic model: it mostly affects the unigram distribution, with small higher-order effects. Thus, we generate proxy embeddings from an oracle encoding the unigrams in the target sentences, described below.

4.2.1 Generate unigram embeddings

We first transform each sentence $W = w_1, \ldots, w_n$ in our training corpus $\mathcal{D}$ into an empirical unigram

distribution ("bag of words") over the vocab $\mathcal{V}$,

$$\tilde{P}(W) = \left\{ w \in \mathcal{V} : \frac{\sum_{i=1}^{n} \mathbb{1}(w_i = w)}{n} \right\}. \quad (5)$$

Next, a feed-forward auto-encoder, $\mathcal{F}_\Theta$, reconstructs $\tilde{P}(W)$ through a low-dimensional hidden layer fitted by minimizing reconstruction loss with Kullback–Leibler (KL) divergence,

$$\mathcal{L}(\Theta; \mathcal{D}) = \sum_{i=1}^{N} \text{KL}(\mathcal{F}_\Theta(\tilde{P}_i) || \tilde{P}_i). \quad (6)$$

The layers were $28996 \times 128 \times 16 \times 128 \times 28996$ with ReLU non-linearities, and a final softmax layer to generate a distribution over the vocabulary. We swept multiple architecture sizes and saw no gain for more parameters. Reconstruction loss on the test set improved from 5.59 to 1.94 after ten epochs.

4.2.2 Train decoder with proxy embeddings

We then replace the context encoder with this auto-encoder; freeze the sentence encoder and fit the parameters of the decoder on training data that do not have context annotations (see Figure 2). In place of the context embedding, we construct a proxy embedding $\hat{a}$ by adding Gaussian noise to the embedding of the *entire* sentence and re-normalizing.

$$\hat{a} = \mathcal{F}_\Theta(\tilde{P}) + \mathcal{N}(0, \sigma^2) \quad (7)$$

$$\hat{a} = \frac{\hat{a}}{\|\hat{a}\|^2}, \quad (8)$$

As we increase σ, the information content in $\hat{a}$ decreases, calibrating the information content of the proxy embeddings to match the expected strength of the actual context. Section 6.3 details the importance of this hyperparameter. We then project this low-dimensional embedding to the target contextual embedding (512 in our experiments) through a linear projection and pass it as input to the decoder.

4.2.3 Adapting the context encoder

Once annotated sentences with context are available for adaptation, we train only the context encoder to project available context into an embedding space tuned to the decoder. The decoder was "primed" to attend to an external embedding and the sentence-internal embeddings. We freeze the decoder and sentence encoder weights; backpropagate; and update only the weights of the transformer in the context encoder and the linear projections that scale from DistilBERT embeddings to

Purpose	#Articles	#Sentence	#Words
Word LM training	250K	8.5M	215M
Context training	55K	1.8M	47M
Context adaptation	60K	2M	41M
Validation	5K	170K	4.3M
Test	5K	166K	4.2M

Table 2: *NYTimes corpora used in this work.* We randomly shuffle all articles before partitioning and use 20% of the entire corpus to reduce experiment turnaround time.

the context embedding dimension. As mentioned above, our encoder is agnostic to the ordering of the context types and transforms text into an embedding through DistilBERT. Section 6 demonstrates that this approach successfully adapts to unseen context data.

5 Corpus

We used the New York Times Annotated Corpus (Sandhaus, 2008) released through Linguistic Data Consortium (catalog number LDC2008T19) containing over 1.8M English articles spanning 1987 to 2007. This corpus includes a rich collection of contextual annotations for each article, ideal for evaluating our CUE framework. Each article contains up to 47 different metadata types that were labeled by humans (author, title, desk) or algorithmically (locations, topic). We down-selected from 47 to 11 distinct metadata signals after removing redundant or uninformative context. All context types were character sequences and include previous sentence, title, author, entities present in the article, section descriptors, date, and topic descriptors (see Appendix A for details). Articles averaged 32 sentences in length and average sentence length (after tokenization) was 26. We trained the sentence encoder on a large subset of articles; used separate training and adaptation corpora and separate validation and test sets (Table 2).

6 Experimental Results

6.1 Hyper-parameters

Sentences were tokenized first with spaCy (Honnibal and Montani, 2017) and then into word pieces using the DistilBERT tokenizer. We evaluated model performance by computing perplexity (PPL) on the heldout test set. We trained all models for ten epochs using the AdamW (Loshchilov and Hutter, 2017) optimizer and One Cycle learning rate scheduler (Smith and Topin, 2017) with a learning rate of 0.0001, maxing at 0.004; and gradient clip-

Contextual features	Test PPL	Rel. PPL
Sentence-internal only	36.6	-
+ article metadata	35.9	-2.0%
+ previous sentence	29.8	-18.6%
+ previous sentence + metadata	27.4	-25.0%

Table 3: *Reduction in PPL by adding context.* We contrast a sentence-internal transformer LM with four variations of added contextual information. Article metadata (e.g., author, title) is mildly informative, the previous sentence is the most useful. Metadata improves PPL *more* when previous sentence is included.

ping of 0.95. We parallelized batches on 8 V100 GPUs and averaged 75k tokens per second with a per-GPU batch size varying between 64 and 256 sentences depending on the architecture size. The parameters of the sentence encoder and DistilBERT are frozen for all the experiments, greatly speeding up training time with negligible impact on PPL.

6.2 Contextual biasing

We first compare our architecture against a sentence-internal auto-regressive language model. The 6x512, 4-head transformer word language model was trained on the separate 200M-word corpus and used as the sentence-encoder in our full CUE framework. As shown in Table 3, contextual signals reduce PPL by 25% for this corpus and nearly three fourths of that gain is due to the previous sentence. Since the remaining contextual features are at the *article* level, they have a smaller impact on within-sentence likelihoods.[2] This 25% relative gain is the upper bound for adaptation methods since context types are consistent between training and test; and context encoder and decoder are trained jointly.

To evaluate the key elements of our architecture, we conducted an ablation study by disabling various components and measured the relative degradation in perplexity, as shown in Table 4. Removing the transformer after DistilBERT embeddings and using a simple average gives an 8% degradation. Removing the transformer decoder and instead concatenating the CUE vector with each step's hidden state before the logit layer gives a 22% degradation. Replacing DistilBERT with a randomly initialized transformer estimated on the contextual training corpus give the biggest loss of 27%. Finally, using only the context to predict each word (a constant vector at each step) is much worse, but still 45x better than random (which would be equal to the

---

[2]See Appendix B for a breakdown of the relative strength of each contextual type.

Module	Test PPL	Rel. PPL
Full architecture	27.4	0%
No context transformer	29.6	+8%
No decoder transformer	33.5	+22%
No DistilBERT	34.8	+27%
No contextual inputs	36.6	+33%
No sentence inputs	643	+2200%

Table 4: *Ablation study on architecture modules.* Refer to Figure 1 for a schematic of the components.
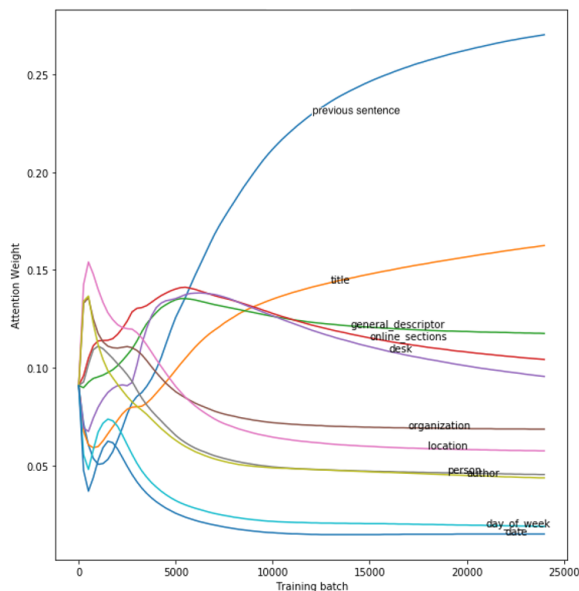

Figure 3: *Change in normalized attention weights over training iterations.* The weights of the self-attention component of the context encoder converge to the relative importance of each contextual category over time, with previous sentence receiving the most weight.

vocab size of 28,996). Context is a useful prior, even though it is constant for all tokens in the sentence.

Figure 3 captures the model's attention converging to the relative importance of each context type. The ordering of context types by attention weights is similar to a ranking by perplexity gain given in Appendix B.

6.3 Adaptation

We next evaluate our framework for interchangeability of different forms of context. We randomly partitioned the eleven context types into two sets **A** and **B** and report the average over five separate trials in Table 5. Set **A** is our training set and we experiment with two adaptation scenarios: **B** replaces **A** or **B** is added to **A**.

When *adding* additional context types ($\mathbf{A} \rightarrow \mathbf{A}+\mathbf{B}$), adapting the context encoder without retraining the decoder captures 85% of the possible gain for jointly training the context encoder and de-

Row	Description	Train	Adapt	Test	PPL	Rel. PPL
1	Word-only baseline	-	-	-	36.6	-
2	Proxy training (cheating)	Proxy	-	Proxy	29.7	-19%
3	Context A always available	A	-	A	29.3 ± 1.3	$-20 \pm 4\%$
4	No training context	Proxy	B	B	31.8 ± 0.8	$-13 \pm 2\%$
5	No adaptation	A	-	B	32.0 ± 0.2	$-13 \pm 1\%$
6	Context A replaced by B	A	B	B	30.9 ± 0.7	$-16 \pm 2\%$
7	Context B always available	B	-	B	29.9 ± 1.4	$-19 \pm 4\%$
8	No training context	Proxy	A+B	A+B	30.4	-20%
9	No adaptation	A	-	A+B	30.1 ± 2.1	$-18 \pm 6\%$
10	Context B added after training	A	A+B	A+B	28.8 ± 1.0	$-21 \pm 3\%$
11	Context A+B always available	A+B	-	A+B	27.4	-25%

Table 5: *Adaptation results.* Results are averaged over five random partitions of context types into training set **A** and adaptation set **B**. Results without std. dev. are based on a single experiment run. Adding metadata with CUE embeddings outperforms a word-only model (row 1) by 25% (row 11). CUE vectors are robust to evolving context, either without any context in training (rows 4, 8); no adaptation (rows 5, 9); or adapting with new annotated sentences (rows 6, 10). Contrast with lower bound of all context available in training and adaptation (rows 7,11).

coder on all context types (compare rows 1, 10 and 11). Starting at a word-only PPL of 36.6, adaptation to **A**+**B** reaches 28.8 versus the lower bound of 27.4. When *replacing* context types (**A** → **B**), adaptation also achieves 85% of the possible gain (36.6 to 30.9 versus the lower bound of 29.9 in rows 1, 6 and 7). Even without *any* adaptation (rows 5 and 9), our architecture generalizes to new context types (approximately 70% of the possible gain), though not as effectively as with adaptation. This is because we transform context into English text and leverage BERT embeddings as a strong initial embedding for context sequences.

When we train the decoder with proxy embeddings (no real context at all) and adapt to context, the PPL is within 6% to 11% (depending on the context subset) of the lower bound of jointly training the context encoder and decoder. This approach recovers 67% of the gain from jointly training context encoder and decoder for the two scenarios (**A** → **B** and **A** → **A**+**B**). We find this quite remarkable given that the decoder knows nothing of context during training; the result validates our hypothesis that context encodes the unigram priors.

We tuned the strength of the proxy embedding by sweeping the variance of Gaussian noise added. The sweet spot is where the information content in the proxy embedding is close to the actual context, as shown in Figure 4. This intuitive result provides a sensible recipe for setting this hyperparameter in a production setting.

6.4 Different sentence encoders

Our CUE architecture factors the context encoder from the sentence encoder and decoder. This approach generates one embedding that can be used with multiple decoders and sentence encoder pairs.
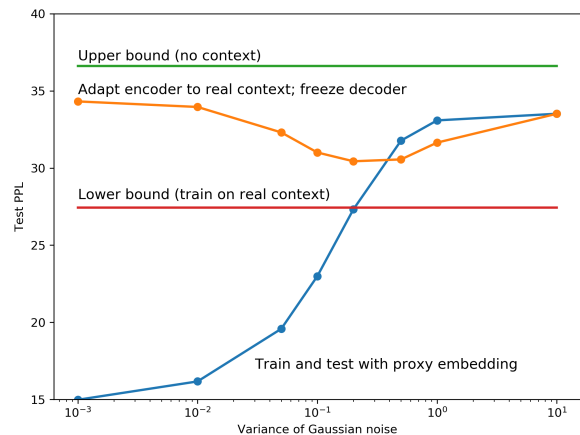

Figure 4: *Varying proxy embedding strength.* The baseline is no context (green line) versus the lower bound of knowing all context in training and test (red line). We sweep the amount of noise added to the oracle unigram vector on the x-axis. When training and testing on only the unigram vector (blue line) the unigram vector is a powerful oracle without noise, but then becomes random as the variance increases. During adaptation (orange line), we discard the unigram embeddings, freeze the decoder parameters, and retrain the context encoder (5% of parameters). The amount of embedding noise is optimal roughly when the proxy embedding is as informative as actual context (where blue and red lines intersect).

To evaluate the generalizability of our CUE framework, we trained a 4x512 LSTM sentence encoder and froze its model parameters for the remaining experiments. We then trained a new decoder using the LSTM sentence encoder and evaluated two different context encoders: 1) randomly initialized and jointly trained with the decoder *or* 2) the pretrained encoder jointly trained with the old, *transformer*-based sentence encoder and frozen parameters. Table 6 shows that the CUE vectors trained with one sentence-LM architecture

are useful to the other, with a relative degradation when swapping of 7% and 1%, respectively, between LSTM and transformer sentence encoders.

Row	Sentence Encoder	Context Encoder	PPL
1	Transformer, frozen	jointly trained	27.4
2	Transformer, frozen	frozen from (3)	29.4
3	LSTM, frozen	jointly trained	28.0
4	LSTM, frozen	frozen from (1)	28.2

Table 6: *Swapping context and sentence encoders.* Without fine tuning, frozen context encoders generalize to new sentence encoders and decoders with <7% relative degradation (compare rows 1 and 2; 3 and 4).

These results suggest that our CUE framework can factor the context encoder and decoder training and generalize to multiple decoder architectures. This frequently occurs in operational settings such as the first and second pass LMs in speech recognition or compressed parameter sizes due to memory and latency constraints.

7 Discussion

We analyzed whether a sentence's context behaves like a cache model, since it contains textual data from the previous sentence, title, and other contexts. To better understand this effect, we divided test data tokens into two bins: Those that appeared in the text of the sentence's context and those that did not. We measured the relative gain in log likelihood when conditioning the LM on context versus not. 30% of the tokens appeared in the context (cache) and the relative gain was 74%—there clearly is a strong benefit for recurring tokens and the CUE encoders capture this effect. The 70% of tokens that do not occur in the cache improved their log likelihood by 26% relative. So the cache effect does not explain the entire benefit of CUE vectors and correlations among different token types are captured as well. The top context types that had tokens in the sentence were previous sentence (23%), title (9%) and person (6%).

To verify that the empirical improvements from the previous sections are semantically plausible, we analyzed the context embeddings of the *first sentence* of 5000 heldout articles. These embeddings do not contain information from the previous sentence and thus represent the entire article's metadata. Figure 6 projects these embeddings down to two dimensions with t-SNE. We then clustered the vectors with k-means and aggregated word counts for all articles within a cluster.

Finally, we display the five most salient words (computed with TF-IDF) from the context and, separately, from the article text. Even though the articles were clustered based only on context, the groupings of *article text* are semantically meaningful, with clear clusters related to newspaper sections such as corrections, marriage announcements, sports and other news related topics. Our context embedding is preserving semantic information.

One limitation of the proxy embedding approach is that they may not extend to other NLP tasks, like named entity tagging. Since they are derived from the unigram embedding, they directly encode the targets of the language model task. This may not prove useful for higher-order annotations and further work should look into a multi-task proxy embedding that directly optimizes an "interface" embedding space instead of a unigram distribution.

8 Conclusions

We introduced the CUE framework to factor context encoding and next word prediction of context-aware neural language models. Unlike previous work, we do not assume that the set of context signals is constant between training and test. We optimize the model architecture to reduce the operational burden of managing and retraining of large neural LMs over their life cycle.

Our approach is robust to changing context types; by adapting only 5% of the parameters, we recover 85% of the possible gain from jointly training all components. Furthermore, we introduce *proxy embeddings* to pretrain a decoder to be attuned to external context embeddings even when those are not known at training time. This approach is 67% as good as jointly training with all context.

In future work, we would like to handle missing context at inference time through data imputation or dropout approaches. Furthermore, we plan to extend the proxy embedding approach such that the context encoders can be trained fully independent of the decoder.

References

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

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

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

Timo I. Denk and Ana Peleteiro Ramallo. 2020. Contextual BERT: Conditioning the language model using a global state. In *Proceedings of the Graph-based Methods for Natural Language Processing (TextGraphs)*, pages 46–50, Barcelona, Spain (Online). Association for Computational Linguistics.

Richard Diehl Martinez, Scott Novotney, Ivan Bulyko, Ariya Rastrow, Andreas Stolcke, and Ankur Gandhe. 2021. Attention-based contextual language model adaptation for speech recognition. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 1994–2003, Online. Association for Computational Linguistics.

Nicolas Fiorini and Zhiyong Lu. 2018. Personalized neural language models for real-world query auto completion. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 3 (Industry Papers)*, pages 208–215, New Orleans - Louisiana. Association for Computational Linguistics.

Matthew Honnibal and Ines Montani. 2017. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. https://sentometrics-research.com/publication/72/.

Aaron Jaech and Mari Ostendorf. 2018a. Low-rank RNN adaptation for context-aware language modeling. *Transactions of the Association for Computational Linguistics*, 6:497–510.

Aaron Jaech and Mari Ostendorf. 2018b. Personalized language model for query auto-completion. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 700–705, Melbourne, Australia. Association for Computational Linguistics.

Mahaveer Jain, Gil Keren, Jay Mahadeokar, Geoffrey Zweig, Florian Metze, and Yatharth Saraf. 2020. Contextual RNN-T for open domain ASR. In *Proc. Interspeech*, pages 11–15, Shanghai.

Yangfeng Ji, Trevor Cohn, Lingpeng Kong, Chris Dyer, and Jacob Eisenstein. 2015. Document context language models. arXiv preprint arXiv:1511.03962.

Suyoun Kim and Florian Metze. 2018. Dialog-context aware end-to-end speech recognition. In *2018 IEEE Spoken Language Technology Workshop (SLT)*, pages 434–440. IEEE.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 1188–1196, Bejing, China. PMLR.

Rui Lin, Shujie Liu, Muyun Yang, Mu Li, Ming Zhou, and Sheng Li. 2015. Hierarchical recurrent neural network for document modeling. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 899–907.

Bing Liu and Ian Lane. 2017. Dialog context language modeling with recurrent neural networks. In *Proc. IEEE ICASSP*, pages 5715–5719, New Orleans.

Ilya Loshchilov and Frank Hutter. 2017. Fixing weight decay regularization in Adam. *CoRR*, abs/1711.05101.

Min Ma, Shankar Kumar, Fadi Biadsy, Michael Nirschl, Tomas Vykruta, and Pedro Moreno. 2018. Modeling non-linguistic contextual signals in lstm language models via domain adaptation. In *Proc. IEEE ICASSP*, pages 6094–6098, Calgary, Alberta.

Ryo Masumura, Tomohiro Tanaka, Atsushi Ando, Hosana Kamiyama, Takanobu Oba, Satoshi Kobashikawa, and Yushi Aono. 2019. Improving Conversation-Context Language Models with Multiple Spoken Language Understanding Models. In *Proc. Interspeech*, pages 834–838, Graz.

Tomas Mikolov and Geoffrey Zweig. 2012. Context dependent recurrent neural network language model. In *2012 IEEE Spoken Language Technology Workshop (SLT)*, pages 234–239. IEEE.

Yasufumi Moriya and Gareth J. F. Jones. 2018. Lstm language model adaptation with images and titles for multimedia automatic speech recognition. In *2018 IEEE Spoken Language Technology Workshop (SLT)*, pages 219–226.

Tsendsuren Munkhdalai, Khe Chai Sim, Angad Chandorkar, Fan Gao, Mason Chua, Trevor Strohman, and Françoise Beaufays. 2021. Fast contextual adaptation with neural associative memory for on-device personalized speech recognition. arXiv preprint arXiv:2110.02220.

Jihyeon Roh, Huiseong Gim, and Soo-Young Lee. 2020. Hierarchical GPT with congruent transformers for multi-sentence language models. arXiv preprint arXiv:2009.08636.

Evan Sandhaus. 2008. The New York Times annotated corpus. Linguistic Data Consortium. Catalog No. LDC2008T19.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. arXiv preprint arXiv:1910.01108.

Ashish Shenoy, Sravan Bodapati, and Katrin Kirchhoff. 2021. Contextual biasing of language models for speech recognition in goal-oriented conversational agents. arXiv preprint arXiv:2103.10325.

Leslie N. Smith and Nicholay Topin. 2017. Super-convergence: Very fast training of residual networks using large learning rates. CoRR, abs/1708.07120.

Amane Sugiyama and Naoki Yoshinaga. 2021. Context-aware decoder for neural machine translation using a target-side document-level language model. In Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 5781–5791, Online. Association for Computational Linguistics.

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 Proceedings of the 31st International Conference on Neural Information Processing Systems, NIPS'17, page 6000–6010, Red Hook, NY, USA. Curran Associates Inc.

Tian Wang and Kyunghyun Cho. 2016. Larger-context language modelling with recurrent neural network. In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 1319–1329, Berlin, Germany. Association for Computational Linguistics.

Ian Williams, Anjuli Kannan, Petar Aleksic, David Rybach, and Tara Sainath. 2018. Contextual speech recognition in end-to-end neural network systems using beam search. In Proc. Interspeech, pages 2227–2231, Hyderabad.

Wayne Xiong, Lingfeng Wu, Jun Zhang, and Andreas Stolcke. 2018. Session-level language modeling for conversational speech. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 2764–2768, Brussels, Belgium. Association for Computational Linguistics.

Hyeongu Yun, Yongkeun Hwang, and Kyomin Jung. 2020. Improving context-aware neural machine translation using self-attentive sentence embedding. Proceedings of the AAAI Conference on Artificial Intelligence, 34(05):9498–9506.

Jiacheng Zhang, Huanbo Luan, Maosong Sun, Feifei Zhai, Jingfang Xu, Min Zhang, and Yang Liu. 2018. Improving the transformer translation model with document-level context. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 533–542, Brussels, Belgium. Association for Computational Linguistics.

Ding Zhao, Tara N. Sainath, David Rybach, Pat Rondon, Deepti Bhatia, Bo Li, and Ruoming Pang. 2019. Shallow-Fusion End-to-End Contextual Biasing. In Proc. Interspeech, pages 1418–1422, Graz.

A   Appendix: Examples of Context Types

Type	Example	Avg. # tokens	%Articles
Author	Michael T. Kaufman	6	84.0%
Date	May 25 1985	4.6	99.9%
Day of Week	Monday	1	99.9%
Descriptor	Computers And The Internet	5.1	83.4%
Desk	Business/Financial Desk	12	99.5%
General Descriptor	Surfing, Ranching	4.5	100%
Location	New York, NY	4.5	42.1%
Online Section	Business; Technology	4.5	99.0%
Organization	Linguistic Data Consortium (LDC)	5	42.4%
Person	Bloomberg, Michael	8	83.7%
Previous Sentence	beloved wife of the late freddy pomerantz.	26	97.0%
Title	Voice Recognition Is Improving, but Don't Stop the Elocution Lessons	8	100%

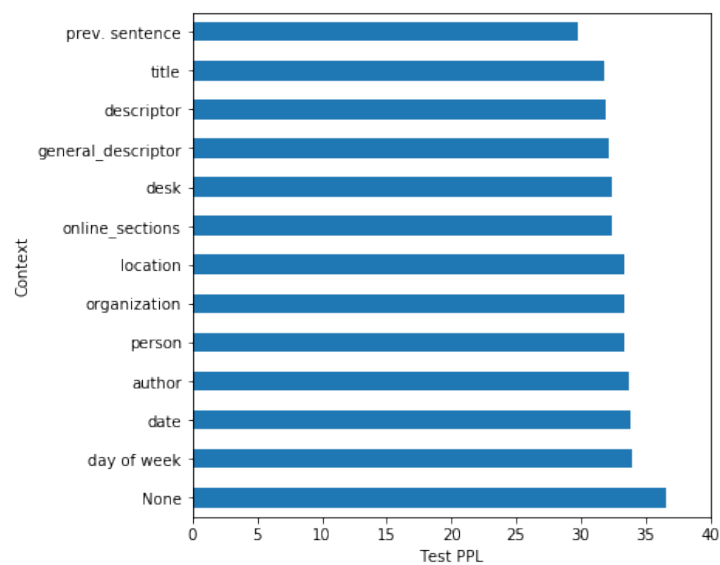# B   Appendix: Relative strength of context types

Figure 5: *Relative strength of each contextual type*. We trained the CUE model with only one contextual signal at a time and measured perplexity on the same heldout test set. Textual context types (previous sentence, title, descriptor) are the most powerful.

C   Appendix: Qualitative Visualization

Figure 6: *T-SNE plot of context embeddings*. We cluster the *first sentence* embedding of 5000 articles and project the 512-d context vectors to two dimensions with t-SNE. We group context vectors into clusters with k-means and compute TF-IDF scores separately for context (green) and sentence (blue) words and show the top 5 for each. Notice how the set of five green words cohere with the five blue words, indicating the CUE embeddings project context and metadata to a similar space as the article contents. The clustering recovers meaningful news topics, such as company earning reports, obituaries, sports, books and art.