Syntax-driven Iterative Expansion Language Models for Controllable Text Generation

Noe Casas[†*], **Jose A. R. Fonollosa**[*], **Marta R. Costa-jussà**[*]
[†] Lucy Software, United Language Group
[*] TALP Research Center, Universitat Politècnica de Catalunya
{noe.casas,jose.fonollosa,marta.ruiz}@upc.edu

Abstract

The dominant language modeling paradigm handles text as a sequence of discrete tokens. While that approach can capture the latent structure of the text, it is inherently constrained to sequential dynamics for text generation. We propose a new paradigm for introducing a syntactic inductive bias into neural text generation, where the dependency parse tree is used to drive the Transformer model to generate sentences iteratively.

Our experiments show that this paradigm is effective at text generation, with quality between LSTMs and Transformers, and comparable diversity, requiring less than half their decoding steps, and its generation process allows direct control over the syntactic constructions of the generated text, enabling the induction of stylistic variations.

1 Introduction

The currently dominant text generation paradigm is based on generating a sequence of discrete tokens in a left-to-right autoregressive way. Most neural language models (LMs) fall into this autoregressive generation category. Some neural architectures are sequential in nature, such as those based on recurrent neural networks (RNNs), lending themselves naturally to the autoregressive approach when used together with teacher forcing (Williams and Zipser, 1989). Other architectures, such as Transformer (Vaswani et al., 2017), while not intrinsically sequential, have also been targeted for sequential generation. On the other hand, some recent lines of research have focused on nonsequential generation. In this work, we propose a new paradigm for text generation and language modeling called Iterative Expansion Language Model, which generates the final sequence following a token ordering defined by the sentence dependency parse by iteratively expanding each level of the tree.

2 Related Work

In this section, we provide an overview of works related to ours, including dependency tree-driven LMs (§2.1), syntax-driven generation (§2.2), insertion-based approaches (§2.3) and iterative refinement approaches (§2.4).

2.1 Dependency LMs

The use of dependency parse trees to drive a language model was first proposed by Chelba et al. (1997), with a similar structure to an n-gram LM, but where the context of a word is its preceding bigram plus a list of preceding words whose parent does not precede it. Shen et al. (2008) make use of the dependency tree in a probabilistic LM, computing the probability of each word conditioned on its parent and the sibling words between both.

Mirowski and Vlachos (2015) propose a dependency LM based on RNNs, where the dependency tree is decomposed into a collection of unrolls, that is, paths from the root to one of the leaves, and where the probability of a word can be predicted from these unrolls. Buys and Blunsom (2018) propose a shift-reduce transition-based LSTM (Hochreiter and Schmidhuber, 1997) dependency LM that can be used for language modeling and generation by means of dynamic programming.

2.2 Syntax-driven Generation

Recurrent neural network grammars (Dyer et al., 2016) are recursive models that operate with a stack of symbols that can be populated with terminals or nonterminals, or "reduced" to generate a syntactic constituent, obtaining as a result a sentence and its associated constituency parse tree.

Shen et al. (2018) use skip-connections to integrate constituent relations with RNNs, learning the underlying dependency structures by leveraging a syntactic distance together with structured

Proceedings of 4th Workshop on Structured Prediction for NLP, pages 1–10
November 20, 2020.

attention.

Akoury et al. (2019) use a simplified constituency tree as latent variables, modeling it autoregressively to later use it as input for a non-autoregressive transformer that generates the output sentence.

Ordered neurons (Shen et al., 2019) are modified LSTMs where the latent sentence tree structure is used to control the dependencies between recurrent units with a special "master" input and forget gates.

2.3 Insertion-based Generation

Stern et al. (2019) propose a conditional generative model that iteratively generates tokens plus the position at which they should be inserted within the sequence. Emelianenko et al. (2019) further propose to optimize the generation order by sampling from the ordering permutations. Instead, Chan et al. (2019) optimize a lower bound of the marginalized probability over every possible ordering.

Gu et al. (2019a) handle the generation order as a latent variable that is captured as the relative position through self-attention, optimizing the ELBO to train the model.

Levenshtein Transformer (Gu et al., 2019b) is a non-autoregressive approach trained with reinforcement learning (RL) to generate token insertion and deletion actions. While it benefits from the same generation speed-ups over autoregressive models as our model, it has the added difficulty of learning an insertion/deletion policy using RL without any linguistically or empirically motivated priors, which can be slow or difficult to obtain convergence in practice. By comparison, our approachmakes uses a linguistically motivated prior for word insertion in a fully supervised way, avoiding the optimization difficulties of RL.

Welleck et al. (2019) use cost minimization imitation learning to learn a policy to generate a binary tree that is used to drive the token generation.

2.4 Iterative Refinement

Lee et al. (2018) propose a latent variable non-autoregressive machine translation model where first the target length is predicted by the model, and then, the decoder is iteratively applied to its own output to refine it.

Mask-predict (Ghazvininejad et al., 2019) also predicts the target sentence length and then non-autoregressively predicts the sentence itself, iteratively refining it a fixed number of times, masking out and regenerating the tokens it is least confident about. Lawrence et al. (2019) follow a similar approach and start with a sequence of placeholder tokens (all the same) of a specified length, and they iteratively replace them with normal tokens via masked LM-style inference. As the masking strategy for the training data, the authors propose different stochastic processes to randomly select which placeholders are to be uncovered.

3 Iterative Expansion LMs

Our proposal is to train a new kind of language model where the token generation order is driven by the dependency parse tree of the sentence and where the generation process is iterative.

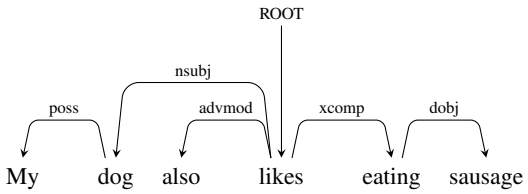

Figure 1: Example of dependency parse tree.

The input vocabulary contains terminal tokens as well as non-terminal special tokens called dependency placeholders, each of which is associated with one of the possible dependency relations to the heads. For the dependency tree in Figure 1, the dependency placeholders are `[poss]`, `[nsubj]`, `[advmod]`, `[xcomp]`, `[dobj]` and `[ROOT]`.

The input of the first iteration is the sequence with the `[ROOT]` element. At each iteration, the model receives as input a sequence I_{tok} with tokens from the input vocabulary and non-autoregressively generates two new sequences, each with the same length as the input.

The first output sequence, O_{tok} , contains tokens from a vocabulary with all possible textual tokens (terminal tokens). The second output, O_{exp} , is a sequence of tokens called expansion placeholders, which are taken from a separate vocabulary. Each expansion placeholder is associated with a pattern describing the left and right dependencies of the token at that position in the O_{tok} sequence. An example of dependency expansion could be `[nsubj-advmod-HEAD-xcomp]` for the word "likes" in the dependency parse tree from Figure 1.

After each iteration, the output of the model is expanded.[1] This consists of creating a new sequence

[1]The expansion of the output to be fed as input in the next iteration occurs in the CPU outside of the neural model itself.

by combining the tokens from I_{tok} , O_{tok} and O_{exp} . This process is illustrated in Figure 2, making use of the dependency tree from Figure 1.

When there is a padding token `[pad]` in the output (either O_{tok} or O_{exp}), this means that the output at that position is ignored when computing the loss function. This occurs when the terminal token has already been computed in previous iterations and has therefore been received as part of I_{tok} , and the model does not need to compute it again.

Note also that an empty dependencies token `[HEAD]` marks the end of a branch and that there is no need for an end of sequence token `<eos>`. As shown in the example from Figure 1, the generation of different branches occurs in parallel, needing only 3 iterations to generate a 6-token sentence.

Iteration 1						
I_{tok} :			[ROOT]			
O_{tok} :			likes			
O_{exp} :	[nsubj-advmod-HEAD-xcomp]					
Iteration 2						
I_{tok} :	[nsubj]	[advmod]	likes	[xcomp]		
O_{tok} :	dog	also	[pad]	eating		
O_{exp} :	[poss-HEAD]	[HEAD]	[pad]	[HEAD-dobj]		
Iteration 3						
I_{tok} :	[poss]	dog	also	likes	eating	[dobj]
O_{tok} :	my	[pad]	[pad]	[pad]	[pad]	sausage
O_{exp} :	[HEAD]	[pad]	[pad]	[pad]	[pad]	[HEAD]

Figure 2: Example of iterative text generation.

The strategy for composing tree expansion tokens (e.g., `[nsubj-advmod-HEAD-xcomp]`) may not scale well when single words have many direct dependencies. To alleviate this, we introduce a preprocessing step to modify the dependency tree so that every word has at most one dependency to the left and one to the right. For each word with more than one dependency on any of its sides, we rearrange the tree to force left-to-right dependencies. Although this **tree binarization** reduces the degree of parallelism, it reduces data sparsity and allows handling constructions with a number of dependencies may otherwise be too large for the model to properly capture, such as enumerations (e.g., "I bought a pair of shoes, an umbrella, a beautiful jacket and a bracelet").

Iterative expansion LMs can be naturally extended to subword vocabularies, like byte-pair encoding (BPE; Sennrich et al., 2016): for each word, we decompose its node in the tree into as many nodes as subwords in the word, rearranging the tree so that the head of the old word is now the head of the first subword, and each subsequent subword depends on the previous one, while every dependency of the old word node now depends on the last subword.

3.1 Neural Architecture

The neural architecture proposed is based on a Transformer decoder (Vaswani et al., 2017). To generate the dual output (terminal tokens and expansion placeholders) we condition the generation of terminals on the expansions: the probability distribution over the expansion token space is generated first by projecting from one of the intermediate layers' hidden states. We sample from it and use the resulting expansion IDs as an index to a trainable expansion embedding layer; the embedded vectors are added to the hidden state used to generate them for use as input to subsequent layers.

As described in Section 3, the input and output token vocabularies are different: the latter only contains terminal tokens (plus some special tokens such as `[PAD]`); the former also contains dependency placeholders. However, for practical purposes, at the model level, we define both vocabularies to be the same, both with terminal tokens and dependency placeholders, and we mask the entries of dependency placeholders in the final softmax.

To inject the syntactic dependency information as input into the model, we add a layer of learned positional embeddings containing the position of the head of each token, and we refer to this embedding layer as head position embedding.

The self-attention mask used in Transformer to force causality is not used in our proposal. The input is therefore not masked at all, and the token predictions have access to the full input sequence.

3.2 Training

For training iterative expansion LMs, the main input of the model is the tokens at one of the levels of the dependency parse tree (I_{tok}), while the output is the following level tokens (O_{tok}) and expansion placeholders (O_{exp}). A secondary input to the model are the dependency indexes, which are used in the head position embedding.

The model is trained with the categorical cross-entropy for both tokens and expansion placeholders, then adding both sublosses into the final loss (with equal weights). Tokens generated in previous

iterations appear as [PAD] tokens in the expected output and are ignored when computing the loss.

Training takes place in batches; as the trainable unit is a level transition, a training batch is composed of level transitions from different sentences.

3.3 Inference and Text Generation

In iterative expansion LMs, inference takes place iteratively. The initial state is a batch of [ROOT] tokens, together with the head positions initialized to the special value representing the root node and, in constrained attention variants, a mask with the self-dependency of the single node in each sentence in the batch. At each iteration, the model generates the probability distributions for terminal tokens and expansion tokens. We use nucleus sampling (Holtzman et al., 2020) to sample from them. The terminal token sequences are expanded according to the expansion tokens (see §3), and these are the inputs for the following iteration if there are still unfinished branches. Before sampling from the token and expansion probability distributions, we mask the <unk> token and the dependency placeholders to avoid generating them.

Although iterative expansion LMs could be subject to beam search across iterations, we have not covered such a possibility as part of this work.

4 Experimental Setup

4.1 Unconditional Text Generation

We conducted experiments on unconditional text generation following the methodology used by Caccia et al. (2020). The goal is to assess both the quality and diversity of the text generated by the model and the baselines. For the quality evaluation, we use the BLEU score (Papineni et al., 2002) over the test set, where each generated sentence is evaluated against the whole test set as a reference. For diversity, we used the self-BLEU score (Zhu et al., 2018), computed using as references the rest of the generated sentences. For each model, the temperature of the final softmax τ is tuned to generate text in the closest quality/diversity regime to the training data.

Iterative expansion LMs are compared against a standard LM baselines, namely, AWD-LSTM[2] (Merity et al., 2018) and a Transformer LM (Vaswani et al., 2017), both with word (w) and BPE subword (sw) vocabularies. The models were trained on the EMNLP2017 News dataset, which contains news in English, enriched with dependency annotations by corenlp, an automatic annotation tool that provides pre-trained models. Syntax-driven generation baseline models were not included because the only model with an available implementation that is able to do unsupervised text generation are RNNGs, but they proved not to scale even to medium-sized datasets like EMNLP2017 News. When sampling from models, we use nucleus sampling (Holtzman et al., 2020), a form of ancestral sampling that constrains the candidate pool by discarding the distribution tail. Samples from the training and validation data are included for reference. Full hyperparameters and data processing details are described in Appendices D and B.

[2]Abbreviation of ASGD weight-dropped LSTM, where ASGD stands for averaged stochastic gradient descent.

4.2 Style Variation

Iterative expansion LMs drive the generation of text with the dependency parse tree. It is possible to influence the generated trees by altering artificially the probability of the different expansion tokens. To demonstrate this, we modified the decoding process of iterative expansion LMs to force the probability of generating adjectival constructions to be higher than normal, aiming at generating a more descriptive style: during decoding, we multiply the probabilities of the expansion placeholders that express adjectival dependencies (i.e. those containing adjectival modifier "amod" relations), and renormalize the probabilities by dividing by the sum.

We conducted this experiment with the word-level models trained on EMNLP2017 News data. We compute the ratio of adjectives per sentence to verify the increased presence of adjectives, while controlling quality and diversity measures over the generated text for potential degradation.

5 Results and Analysis

We assess the ability of iterative expansion LMs to unconditionally generate text in terms quality (BLEU-5) vs. diversity (self BLEU-5), comparing against sequential baselines, each with a softmax temperature τ tuned separately.

In order to tune the output softmax termperature τ, we generated text with each model at different temperatures and chose the value of τ that was the most similar to a sample from the training data in terms of BLEU-5 against a sample from the validation set (proxy for quality) and self BLEU-5

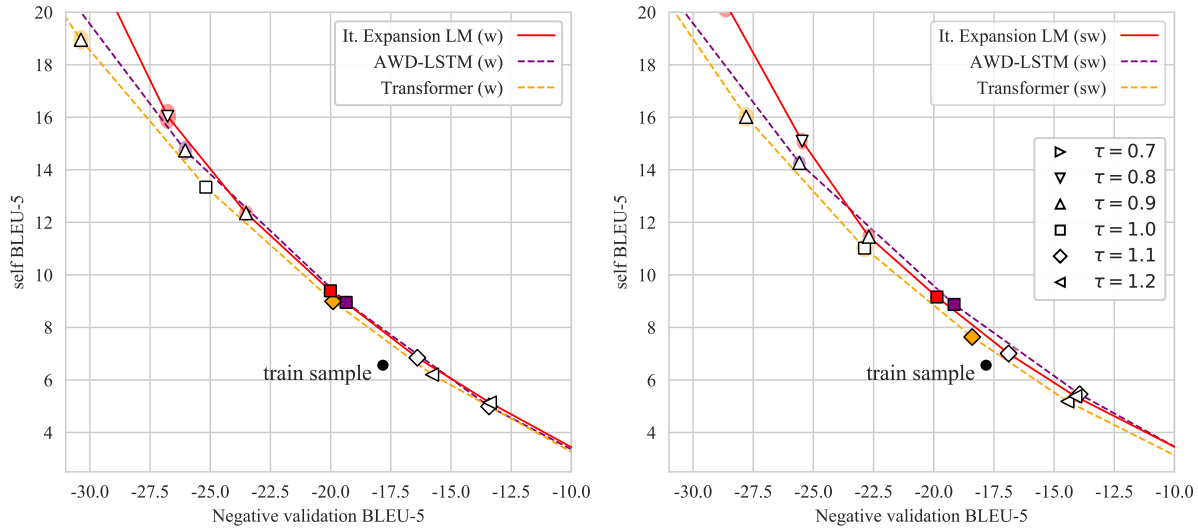

Figure 3: Quality vs. diversity on EMNLP2017 News (BLEU-5). Models with **word-level vocabulary on the left** and **subword-level on the right**. The point marker is color-filled for the chosen value of τ. Each point represents the average over 20 generated text samples, and is surrounded by a small colored ellipse representing the standard deviation.

τ	ITEXP (w)		AWD-LSTM (w)		Transformer (w)	
	valid ↑	**self** ↓	**valid** ↑	**self** ↓	**valid** ↑	**self** ↓
0.70	30.1 ± 0.8	22.3 ± 1.0	39.2 ± 0.9	33.4 ± 1.1	40.5 ± 0.6	35.0 ± 1.1
0.80	26.8 ± 0.8	16.0 ± 1.0	33.0 ± 0.7	23.2 ± 1.0	35.8 ± 0.7	26.3 ± 0.8
0.90	23.5 ± 0.7	12.4 ± 0.7	26.0 ± 0.6	14.7 ± 0.8	30.4 ± 0.7	19.0 ± 0.8
1.00	$\mathbf{20.0 \pm 0.6}$	$\mathbf{9.4 \pm 0.5}$	$\mathbf{19.4 \pm 0.6}$	$\mathbf{9.0 \pm 0.6}$	25.2 ± 0.5	13.3 ± 0.5
1.10	16.4 ± 0.5	6.8 ± 0.5	13.4 ± 0.4	5.0 ± 0.4	$\mathbf{19.9 \pm 0.6}$	$\mathbf{9.0 \pm 0.6}$
1.20	13.4 ± 0.6	5.1 ± 0.4	9.0 ± 0.5	2.9 ± 0.3	15.8 ± 0.5	6.2 ± 0.5

τ	ITEXP (sw)		AWD-LSTM (sw)		Transformer (sw)	
	valid ↑	**self** ↓	**valid** ↑	**self** ↓	**valid** ↑	**self** ↓
0.70	28.6 ± 0.9	20.3 ± 1.1	39.0 ± 0.8	33.5 ± 1.1	36.9 ± 0.7	30.6 ± 1.2
0.80	25.5 ± 0.5	15.1 ± 0.7	32.3 ± 0.7	22.4 ± 0.7	32.5 ± 0.7	22.4 ± 1.0
0.90	22.7 ± 0.6	11.5 ± 0.7	25.6 ± 0.6	14.3 ± 0.6	27.8 ± 0.7	16.0 ± 0.8
1.00	$\mathbf{19.9 \pm 0.6}$	$\mathbf{9.2 \pm 0.5}$	$\mathbf{19.2 \pm 0.5}$	$\mathbf{8.9 \pm 0.5}$	22.9 ± 0.8	11.0 ± 0.7
1.10	16.9 ± 0.8	7.0 ± 0.6	13.9 ± 0.5	5.5 ± 0.4	$\mathbf{18.4 \pm 0.7}$	$\mathbf{7.6 \pm 0.6}$
1.20	14.1 ± 0.6	5.4 ± 0.5	9.7 ± 0.4	3.3 ± 0.3	14.5 ± 0.5	5.2 ± 0.5

Table 1: Validation and self BLEU-5 scores of the text generated by the **word-level (top)** and **subword-level (bottom)** models under study at different temperatures τ, showing the average and standard deviation over 20 different generated text samples. The selected generation regime is highlighted for each model, being the closest to the training sample, which has a validation BLEU-5 of 17.8 and a self BLEU-5 of 6.6.

(proxy for diversity). Each model was used to generate 20 samples of 400 sentences, and self-BLEU5 and validation-BLEU5 were computed over each of them, taking the average and the standard deviation. Figure 3 and Table 1 show these BLEU values, highlighting the chosen τ for each model. Given the low values for the standard deviation, we decided not to include it in subsequent tables to avoid unnecessary clutter. Note that in all BLEU vs. self-BLEU figures, each model is shown as a different line (each with its own color and/or dashed pattern) and that the data points computed for each temperature value are plotted with a specific marker shape (square, diamond, triangle, or flipped triangle).

Apart from BLEU scores, we also include extra quality measures, namely the perplexity obtained

	τ	Test BLEU-5 (quality ↑)	Self BLEU-5 (diversity ↓)	AWD-LSTM perplex. ↓	Transformer perplex. ↓	GPT-2 perplex. ↓
AWD-LSTM (w)	1.0	22.9	8.9	37.0	47.9	99.5
Transformer (w)	1.1	23.8	9.0	33.6	18.6	66.5
ITEXP (w)	1.0	23.7	9.4	40.8	40.7	85.2
AWD-LSTM (sw)	1.0	22.7	8.9	43.5	56.9	113.5
Transformer (sw)	1.1	22.1	7.6	37.5	31.6	77.1
ITEXP (sw)	1.0	23.6	9.2	45.2	49.2	97.1
Train sample	-	21.5	6.6	49.5	29.1	37.7
Valid sample	-	21.2	7.2	53.3	44.7	36.7

Table 2: Quality and diversity on EMNLP2017, with τ generating the closest text to the validation data.

by other language models: an AWD-LSTM word-level LM and a Transformer word-level LM, both trained on EMNLP2017 News, plus OpenAI GPT-2 (1.5 B parameters) (Radford et al., 2019). The results are shown in Table 2.

These results show how the generated text improves over AWD-LSTM in terms of quality by all measures, with a comparable level of diversity. In comparison to the Transformer, while the quality measured with BLEU-5 is better for ITEXP, the rest of the quality measures indicate that the text generated by the Transformer is of better quality.

Adjective probability	Adjs. per sentence	Test BLEU-5	Self BLEU-5
×1	1.2	23.7	9.4
×10	3.4	21.3	8.4
×20	4.2	20.6	8.8
×50	5.2	19.8	8.9

Table 3: ITEXP (w, $\tau = 1.0$) with increased adjectives.

The results of the styled text generation experiments, shown in Table 3, confirm that the style of the resulting text can be successfully modulated to the desired degree and that the quality and diversity are only slightly degraded at moderate increases of the probability of adjectival clause generation.

5.1 Human Evaluation

In order to better assess the quality of the generated text, we also include a human evaluation. For this, we took a sample of 60 sentences of each model under study, including also a sample of the same size from the validation data, to serve as reference. The sentences were evaluated by a pool of annotators, who were requested to rate the sentence in an integer scale from 1 to 5, taking into account its fluency and correctness.

The pack of sentences rated by each annotator contained 10 sentences from each of the models under evaluation. Each sentence under evaluation was part of the packs of 3 evaluators; this redundancy was used to measure the discrepancies in the rating of each sentence among annotators, which was quantified by means of the average per-sentence standard deviation.

Model	Average rating	Per sentence avg. stddev
AWD-LSTM (w)	3.08	0.74
Transformer (w)	3.43	0.78
ITEXP (w)	3.28	0.73
AWD-LSTM (sw)	2.66	0.68
Transformer (sw)	3.33	0.83
ITEXP (sw)	3.09	0.70
Valid sample	4.49	0.61

Table 4: Human evaluation for the different models.

Table 4 shows the statistics of the obtained ratings, were we can see the average rating of the sentences generated by each model, together with the average per-sentence standard deviation, to understand how different the ratings for each sentence were among the different evaluator ratings. We can see that the highest human ratings were obtained by the Transformer, both with word and subword-level vocabularies, followed by ITEXP and then AWD-LSTM.

Table 5 shows the human evaluation for the models from the style variation experiments presented in Table 3. As we can see, there is a small degradation in quality as we force high levels of adjectival presence.

Adjective probability	Average rating	Per sentence avg. stddev
×1	3.28	0.73
×10	3.16	0.79
×20	2.98	0.84
×50	3.19	0.70

Table 5: Human evaluation for ITEXP (w) models with increased adjectival construction probability.

6 Further Comparison with Real Text

Given that the generation process in iterative expansion LMs is not sequential, we studied the distribution of the sentence lengths it generates. This is shown in Figure 4 for the text generated by a word-level iterative expansion LM trained on EMNLP2017 News, along with the lengths of a sample from the training data.

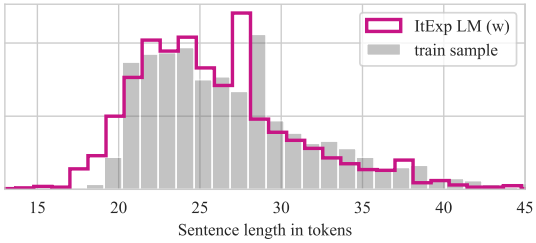

Figure 4: Distribution of generated text length.

Iterative expansion LMs generate the dependency parse tree as they generate text. We studied the depths of the dependency trees of generated text in relation to those parsed from the training data, as shown in Figure 5.

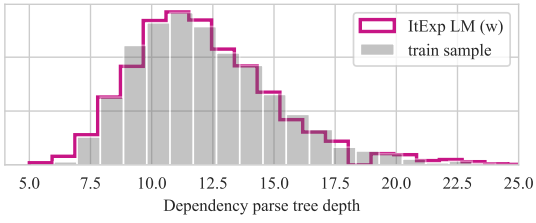

Figure 5: Histogram of generated text tree depth.

We also measured the degree to which the generated trees adhere to the trees obtained by parsing their lexicalized representation. Specifically, we computed the labeled and unlabeled attachment scores between both for the text generated at different softmax temperatures τ. Attachment scores are the standard performance measure in dependency parsing and are computed as the percentage of words that have been assigned the same head as the reference tree, over a test set. The attachment score is "labeled" if the dependency label is taken into account or "unlabeled" otherwise. As shown in Table 6, the obtained labeled attachment scores (LAS) and unlabeled attachment scores (UAS) are very high across the different values of the generation temperature τ.

τ	0.7	0.8	0.9	1.0	1.2
LAS	96.4	95.3	94.2	92.3	86.2
UAS	98.0	97.3	96.5	95.2	90.7

Table 6: Attachment scores of the generated trees.

6.1 Quantification of the Generation Speedup

Text generation with autoregressive models like LSTM or Transformer models offers a linear computational complexity with respect to the length of the generated sequence. In comparison, the dependency tree-driven decoding used by iterative expansion LMs generates text in parallel for each branch in the tree. If the tree was a perfectly balanced binary tree, then the computational complexity would be logarithmic. However, dependency trees in general are not balanced and, given the tree binarization postprocessing that we introduce, the parallelization is slightly reduced.

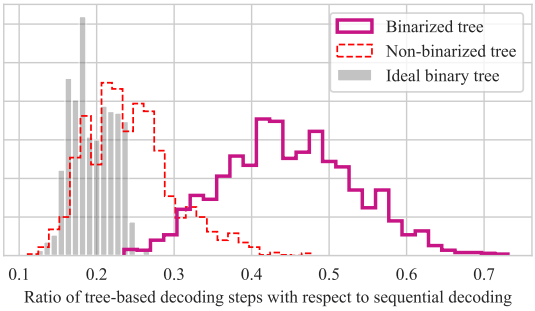

Figure 6: Histogram of the ratio of the decoding steps needed to generate a sentence with tree-based decoding with respect to sequential generation.

Figure 6 shows the speedup of the needed decoding steps of tree-based decoding with respect of auto-regressive decoding, taking a sample of the training data and computing the needed steps to decode them should the sentences have an idealized binary dependency parse tree, a normal parse tree, and a binarized parse tree. On average, the binarized parse tree, which is the decoding used by iterative expansion LMS, needs only 45% of the decoding steps needed by autoregressive decoding.

American students were 62 percent more likely to die in a heart attack during the first week of 2004, according to the study.
For 150 days, Hillary Clinton will do more to improve access to affordable quality care, support and education funding for millions of Americans, she says.
For those on this list, it's likely that I would rather be able to train them up, she said.
He made it clear the SNP repeated on Friday as a response, saying they discussed a contract getting the extra cost here.
He'll pay $25, 000 for rent and more buses and bring his collection to The Academy on Channel 31.
Six years later, at least eight people died as a result of the shooting.
The health prime minister told CNN Thursday that he was willing to back up against the US and remove all of the relevant items at the end of the transition.
Then, another man told police that was a friend's friend, and as a child, he made the decision to call his mother.
They are 40 - 60 among the top 50, 000 women in the last year in that group since 2014 - 15.
They've worked hard on Twitter and they think they've tried to focus on our sport, she said.
We like to think that if you try to get this game done, we can get a lower success rate out of 15.

Table 7: Samples of text generated by iterative expansion LMs with word vocabulary.

I feel that they're going to Syria because we had this explanation, that they have an indication of their advance.
The girl's mother told the group of three she needed treatment and the family said her daughter would still be alive with another child.
But she added: "The data is important to the EU that the UK can attract more businesses.
Though he also spoke to Mr Wilson on Saturday morning at the Netherlands Police trial, Johnson referred it to the No. 1 commission.
It's a collective belief and it's a statement to us, he said.
It's just the first thing we're feeling now and I don't like it.
So if you want to be sitting in a garden, you have to wait for something to make sure that this does not end.
So, for example, we need to argue about what the president did, but I'm just interested in having any talk.
The British defence ministry confirmed action had been taken at the hospital but could not confirm the details until now.
We'll ask for a fair share of Russia to stop border security, particularly for people of color, he added.

Table 8: Samples of text generated by iterative expansion LMs with subword vocabulary.

6.2 Generation Examples

Table 7 shows a selection of text samples generated by iterative expansion LMs with a word-level vocabulary, while Table 8 shows samples generated with a subword-level vocabulary. We can see that, despite being generated non-sequentially and each branch of the dependency parse tree being generated in parallel, the resulting sentences maintain coherence and syntactic agreement, confirming that conditioning on the token dependencies in the parse tree provides enough information to generate it while speeding up the decoding process.

7 Conclusion

In this work, we presented iterative expansion LMs, which are iterative non-autoregressive text generation models that rely on syntactic dependency trees to generate sentence tokens in parallel. As opposed to other syntax-driven generation mechanisms, the training of iterative expansion LMs can be naturally computed in batches and they are amenable to subword-level vocabularies.

We showed that our proposed method generates text with quality between LSTMs and Transformers, with comparable diversity, both regarding automatic measurements and human judgement, while generating text in half of the decoding steps needed by sequential LMs, and also allowing direct control over the generation process at the syntactic level, enabling the induction of stylistic variations in the generated text.

Our code is available as open source at https://github.com/noe/iterative_expansion_lms .

Acknowledgments

This work is partially supported by Lucy Software / United Language Group (ULG) and the Catalan Agency for Management of University and Research Grants (AGAUR) through an Industrial Ph.D. Grant. This work also is supported in part by the Spanish Ministerio de Economía y Competitividad, the European Regional Development Fund through the postdoctoral senior grant Ramón y Cajal and by the Agencia Estatal de Investigación through the projects EUR2019-103819, PCIN-2017-079 and PID2019-107579RB-I00 / AEI / 10.13039/501100011033

References

Nader Akoury, Kalpesh Krishna, and Mohit Iyyer. 2019. Syntactically supervised transformers for faster neural machine translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1269–1281, Florence, Italy. Association for Computational Linguistics.

Jan Buys and Phil Blunsom. 2018. Neural syntactic generative models with exact marginalization. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 942–952, New Orleans, Louisiana. Association for Computational Linguistics.

Massimo Caccia, Lucas Caccia, William Fedus, Hugo Larochelle, Joelle Pineau, and Laurent Charlin. 2020. Language gans falling short. In *International Conference on Learning Representations*.

William Chan, Nikita Kitaev, Kelvin Guu, Mitchell Stern, and Jakob Uszkoreit. 2019. KERMIT: Generative insertion-based modeling for sequences. *arXiv preprint arXiv:1906.01604*.

Ciprian Chelba, David Engle, Frederick Jelinek, Victor Jimenez, Sanjeev Khudanpur, Lidia Mangu, Harry Printz, Eric Ristad, Ronald Rosenfeld, Andreas Stolcke, and Dekai Wu. 1997. Structure and performance of a dependency language model. In *In Proceedings of Eurospeech*, pages 2775–2778.

Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A. Smith. 2016. Recurrent neural network grammars. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 199–209, San Diego, California. Association for Computational Linguistics.

Dmitrii Emelianenko, Elena Voita, and Pavel Serdyukov. 2019. Sequence modeling with unconstrained generation order. In *Advances in Neural Information Processing Systems 32*, pages 7698–7709. Curran Associates, Inc.

Marjan Ghazvininejad, Omer Levy, Yinhan Liu, and Luke Zettlemoyer. 2019. Mask-predict: Parallel decoding of conditional masked language models. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6114–6123, Hong Kong, China. Association for Computational Linguistics.

Jiatao Gu, Qi Liu, and Kyunghyun Cho. 2019a. Insertion-based decoding with automatically inferred generation order. *Transactions of the Association for Computational Linguistics*, 7:661–676.

Jiatao Gu, Changhan Wang, and Junbo Zhao. 2019b. Levenshtein transformer. In *Advances in Neural Information Processing Systems 32*, pages 11179–11189. Curran Associates, Inc.

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

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2020. The curious case of neural text degeneration. In *International Conference on Learning Representations*.

Carolin Lawrence, Bhushan Kotnis, and Mathias Niepert. 2019. Attending to future tokens for bidirectional sequence generation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1–10, Hong Kong, China. Association for Computational Linguistics.

Jason Lee, Elman Mansimov, and Kyunghyun Cho. 2018. Deterministic non-autoregressive neural sequence modeling by iterative refinement. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1173–1182, Brussels, Belgium. Association for Computational Linguistics.

Edward Loper and Steven Bird. 2002. Nltk: The natural language toolkit. In *In Proceedings of the ACL Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics. Philadelphia: Association for Computational Linguistics*.

Stephen Merity, Nitish Shirish Keskar, and Richard Socher. 2018. Regularizing and optimizing LSTM language models. In *International Conference on Learning Representations*.

Piotr Mirowski and Andreas Vlachos. 2015. Dependency recurrent neural language models for sentence completion. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference*

on Natural Language Processing (Volume 2: Short Papers), pages 511–517, Beijing, China. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Libin Shen, Jinxi Xu, and Ralph Weischedel. 2008. A new string-to-dependency machine translation algorithm with a target dependency language model. In *Proceedings of ACL-08: HLT*, pages 577–585.

Yikang Shen, Zhouhan Lin, Chin wei Huang, and Aaron Courville. 2018. Neural language modeling by jointly learning syntax and lexicon. In *International Conference on Learning Representations*.

Yikang Shen, Shawn Tan, Alessandro Sordoni, and Aaron Courville. 2019. Ordered neurons: Integrating tree structures into recurrent neural networks. In *International Conference on Learning Representations*.

Mitchell Stern, William Chan, Jamie Kiros, and Jakob Uszkoreit. 2019. Insertion transformer: Flexible sequence generation via insertion operations. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, pages 5976–5985.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Sean Welleck, Kianté Brantley, Hal Daumé III, and Kyunghyun Cho. 2019. Non-monotonic sequential text generation. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 6716–6726, Long Beach, California, USA. PMLR.

Ronald J Williams and David Zipser. 1989. A learning algorithm for continually running fully recurrent neural networks. *Neural computation*, 1(2):270–280.

Yaoming Zhu, Sidi Lu, Lei Zheng, Jiaxian Guo, Weinan Zhang, Jun Wang, and Yong Yu. 2018. Texygen: A benchmarking platform for text generation models. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 1097–1100. ACM.