Language Models not just for Pre-training: Fast Online Neural Noisy Channel Modeling

Shruti Bhosale[△] **Kyra Yee**[†▽] **Sergey Edunov**[△] **Michael Auli**[△]
[△] Facebook AI Research, Menlo Park, CA, USA
[▽] Twitter Cortex, San Francisco, CA, USA

Abstract

Pre-training models on vast quantities of unlabeled data has emerged as an effective approach to improving accuracy on many NLP tasks. On the other hand, traditional machine translation has a long history of leveraging unlabeled data through noisy channel modeling. The same idea has recently been shown to achieve strong improvements for neural machine translation. Unfortunately, naïve noisy channel modeling with modern sequence to sequence models is up to an order of magnitude slower than alternatives. We address this issue by introducing efficient approximations to make inference with the noisy channel approach as fast as strong ensembles while increasing accuracy. We also show that the noisy channel approach can outperform strong pre-training results by achieving a new state of the art on WMT Romanian-English translation.

1 Introduction

Unlabeled data has been leveraged in many ways in natural language processing including back-translation (Bojar and Tamchyna, 2011; Sennrich et al., 2016b; Edunov et al., 2018), self-training (He et al., 2020), or language model pre-training which led to improvements in many natural language tasks (Devlin et al., 2019). While pre-training has achieved impressive results on tasks where labeled data is limited, improvements in settings with abundant labeled data are modest (Raffel et al., 2020) with controlled studies showing a clear trend of diminishing returns as the amount of training data increases (Edunov et al., 2019).

In this paper, we focus on noisy channel modeling for text generation tasks, a classical technique from the statistical machine translation literature which had been the workhorse of text generation tasks for decades before the arrival of neural sequence to sequence models (Brown et al., 1993; Koehn et al., 2003). Unlike pre-training approaches, this approach is very effective irrespective of the amount of labeled data: since a recent revival (Yu et al., 2017; Yee et al., 2019), it has been an important part in the winning entries of several high resource language pairs at WMT 2019 (Ng et al., 2019), improving over strong ensembles that used 500M back-translated sentences. At the low resource WAT 2019 machine translation competition, noisy channel modeling was also a key factor for the winning entry (Chen et al., 2019).

Noisy channel modeling turns text generation on the head: instead of modeling an output sequence given an input, Bayes' rule is applied to model the input given the output, via a backward sequence to sequence model which is combined with the prior probability of the output, typically a language model. This enables the effective use of strong language models trained on large amounts of unlabeled data. The role of the backward model, or the channel model, is to validate outputs preferred by the language model with respect to the input.

A straightforward way to use language models is to pair them with standard sequence to sequence models (Gülçehre et al., 2015; Stahlberg et al., 2018). However, this does not address *explaining away effects* under which modern neural sequence models still suffer (Klein and Manning, 2001; Li et al., 2019). As a consequence, models are susceptible to producing fluent outputs that are unrelated to the input (Li et al., 2019). The noisy channel approach explicitly addresses this via the channel model.

However, a major obstacle to efficient noisy channel modeling is that generating outputs is much slower than decoding from a standard sequence to sequence model. We address this issue by introducing several simple yet highly ef-

† Work done while at Facebook during a Facebook AI Residency.

fective approximations which increase the speed of noisy channel modeling by an order of magnitude to make it practical. This includes smaller channel models as well as scoring only a subset of the channel model vocabulary. Experiments on WMT English-Romanian translation show that noisy channel modeling can outperform recent pre-training results. Moreover, we show that noisy channel modeling benefits much more from larger beam sizes than strong pre-training methods.

2 The Noisy Channel Approach

We assume a sequence to sequence task that takes the input x to predict the output y. A standard sequence to sequence model directly estimates the probability $p(y|x)$, referred to as a *direct model*. On the other hand, the noisy channel approach applies Bayes' rule to model $p(y|x) = p(x|y)p(y)/p(x)$ where $p(x|y)$ predicts the source x given the target y and is referred to as the *channel model*, $p(y)$ is a language model over the target y, and $p(x)$ is generally not modeled since it is constant for all y.

Yee et al. (2019) use Transformer models to parameterize the direct model, the channel model and the language model. Similar to Yu et al. (2017), they use the following linear combination of the channel model, the language model as well as the direct model for decoding:

$$\frac{1}{t}\log p(y|x) + \frac{\lambda_1}{s}\log p(x|y) + \frac{\lambda_2}{s}\log p(y) \quad (1)$$

where t is the length of the output prefix y, s is the length of the input sequence, and λ_1, λ_2 are hyperparameters.

Exact noisy channel model scoring with neural networks during decoding is prohibitively expensive since it requires a separate forward computation with the channel model for every token in the target vocabulary. To side step this issue, Yu et al. (2017) propose the following approximations to beam search with beam width k_1: determine the k_2 highest scoring extensions of each beam according to the direct model, then score the resulting $k_1 \times k_2$ partial candidates by the direct model, the channel model and the language model using the linear combination in Equation 1. Finally, this set is pruned to beam size k_1.

Despite this approximation, noisy channel decoding is still significantly slower than decoding with the direct model alone as shown in Figure 1.

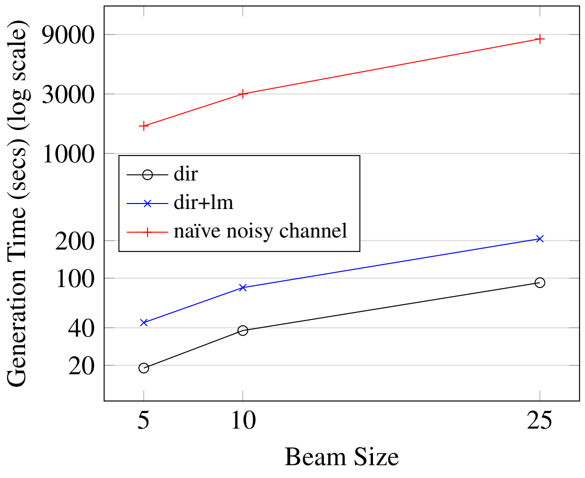

Figure 1: Speed of decoding with a direct model (`dir`), direct model with language model (`dir + lm`) and a naïve noisy channel approach without fast approximations or optimizations. The latter is very slow compared to the direct model. Results are based on generation with the fastest batch size for each setting with beam 5 on newstest2016 De-En (cf. §4.1).

The reason for this is that the channel model repeatedly scores the entire input sequence at each time-step and this is done k_2 times for each beam. Specifically, both the direct model and the language model compute $k_1 \times V$ scores at each time-step in order to make a decoding decision for each target token during beam search, where V denotes the vocabulary size which we assume to be similar between the input and output. In contrast, the channel model computes $k_1 \times k_2 \times S \times V$ scores for each target token, where S is the maximum source sequence length. This adds substantial compute and memory overhead, to the extent that the batch size at decoding often needs to be substantially reduced. This leads to slower inference on GPUs since less computation can be parallelized.

3 Fast Noisy Channel Modeling

Naïve online noisy channel modeling is significantly slower than standard direct models. In this section, we present approximations to make noisy channel modeling substantially faster.

3.1 Reducing Channel Model Size

Prior work on neural noisy channel used channel models which were of the same size as the direct model (Yu et al., 2017; Yee et al., 2019). The most recent work uses standard Transformer models (Yee et al., 2019; Ng et al., 2019; Yu et al., 2020). In this study, we hypothesize that the primary role of the channel model is to avoid *explaining away effects* by the language model. This primarily entails assigning low scores to unrelated outputs, which may not require a very powerful model. In this case, we may be able to substantially decrease the size of the channel model at only a small loss in accuracy.

Recent work demonstrates that direct models with shallow decoders can give comparable accuracy, while being faster at inference time, compared to models with deep decoders (Wu et al., 2019; Elbayad et al., 2020; Kasai et al., 2020; Fan et al., 2020). This is particularly attractive for direct models for which the decoder network accounts for most of the wall time during inference but the dynamics for channel models are different: the channel model repeatedly scores the entire input sequence given progressively larger target prefixes. Unlike for direct models, there is no straightforward way to reuse the encoder output between time-steps, and we opt to recompute the entire encoder and decoder of the channel model at every target time-step. Since the input sequence is given, channel model computation can be batched over all tokens in the target prefix and the input sequence. This implies that we are free to adjust both the encoder and decoder depth.

We pursue two strategies to reduce model size: first, we progressively reduce the model dimension of the `base` Transformer architecture, by first halving the model dimension from 512 to 256, as well as the feed forward dimension from 2048 to 1024 for the `half` model. The smallest configuration uses a model dimension of just 32 and a feed forward dimension of 128 (denoted as `16th` model). Second, we consider models with only a single encoder block and a single decoder block. These models have a postfix `_1_1`, e.g., `16th_1_1`. Table 1 shows the various model sizes as well as accuracy on the development set, newstest2016.

	Parameters (M)	BLEU
big	282.7	38.0
big_1_1	93.8	34.1
base	72.1	36.7
base_1_1	23.6	31.2
half	25.1	33.6
half_1_1	15.8	27.4
quarter	9.8	28.4
quarter_1_1	7.5	22.0
16th	2.8	15.9
16th_1_1	2.7	10.0

Table 1: Smaller channel models in terms of number of total parameters as well as BLEU (avg. over 3 seeds) on the development set. All models have six blocks each in the encoder and the decoder, except for models ending in "_1_1" which have only a single block in the encoder and the decoder.

3.2 Reducing the Output Vocabulary

During online noisy channel decoding, we need to allocate memory for a large number of channel model output probabilites ($k_1 \times k_2 \times S \times V$, as explained in § 2). This substantially reduces the maximum possible batch size in order to prevent running out of memory while decoding on GPUs. A small batch size prevents the full utilization of parallel computation on GPUs, particularly, when the channel model is relatively small: some of our channel models have an embedding dimension of just 32.

To address this issue, we make use of the fact that we know exactly which input tokens need to be scored (since the input sequence is given) instead of computing probabilities for the entire vocabulary. This is similar to vocabulary reduction techniques used for early neural sequence to sequence models, and it is particularly convenient since we know exactly which tokens are in the input sequence (Mi et al., 2016; L'Hostis et al., 2016).

Similar to prior work on vocabulary reduction, we found it useful to not just score the input words but also a subset of the most frequent words in the vocabulary. Specifically, for each batch, we enumerate all input word types, add the 500 most frequent types and then compute output probabilities for this subset with the channel model. The number of output probabilities calculated is typically at least one order of magnitude smaller than the full vocabulary, as shown in § 5.4.1.

This approach substantially reduces the memory footprint of small channel models and enables the use of much larger batch sizes which leads to faster inference as we will see in § 5.

3.3 Reducing the Number of Candidates

We also study the effect of reducing the number of next token candidates k_2 scored for each beam at each step of beam search. This reduces the computation as well as memory overhead of channel model scoring.

4 Experimental Setup

4.1 Datasets

We consider two datasets for our experiments: For German-English (De-En), we train on WMT'19 training data. Following (Ng et al., 2019), we apply language identification filtering (Lui and Baldwin, 2012) and remove sentences longer than 250 tokens as well as sentence pairs with a source/target length ratio exceeding 1.5. This results in 26.8M sentence pairs. We validate on newstest2016 and test on newstest2014, newstest2015, newstest2017, and newstest2018. For all models, the source vocabulary is a 24K byte pair encoding (BPE; Sennrich et al., 2016) learned on the source portion of the bitext. For the target side, we use the vocabulary of the language model (§4.2) so that both models score the exact same units during beam search.

For Romanian-English (Ro-En), we train on WMT'16 training data, comprising 612K sentence pairs, validate on newsdev2016 and test on newstest2016. We learn a joint BPE vocabulary of 18K types on the bitext training data which is used for both the source and target. Different to German-English, we learn a joint BPE vocabulary to enable sharing the source and target embeddings which we found to perform better for Romanian-English in early experiments.

4.2 Language Models

For German-English, we train a sentence-level English Transformer language model with 16 layers and Transformer-Big architecture (Vaswani et al., 2017; Radford et al., 2018). The model is trained on de-duplicated English Newscrawl data from 2007-2018 comprising 186 million sentences or 4.5B words after normalization and tokenization. We use a BPE vocabulary of 24K types learned on this data. For Romanian-English translation, we train a similar English Transformer language model that uses the joint BPE vocabulary learned on the Romanian-English bitext. The latter enables the LM to score the exact same units as the sequence to sequence model during beam search.

We train a sentence-level Romanian Transformer language model with 16 layers and Transformer-Big architecture. The model is trained on de-duplicated Romanian CommonCrawl data consisting of 623M sentences or 21.7B words after normalization and tokenization (Conneau et al., 2019; Wenzek et al., 2020).

The German-English bitext training data as well as the language model training data are preprocessed with the Moses tokenizer (Koehn et al., 2007). We normalize punctuation and remove non-printing characters. Romanian-English data is preprocessed following Sennrich et al. (2016a) by applying Moses tokenization and special normalization for Romanian text.[1]

4.3 Translation Models

For De-En, we use the Transformer-Big architecture for the direct model. We do not share encoder and decoder embeddings since the source and target vocabularies are different. For channel models, operating from English to German, we consider different variants (§3.1, Table 1) to better understand the speed-accuracy trade-off of decreasing the capacity of channel models.

For Ro-En and En-Ro with bitext only, the direct and channel models use a Transformer-Base architecture. For Ro-En with backtranslation, the direct and channel models use a Transformer-Big architecture. We share the encoder and decoder embeddings since the source and target vocabularies are the same and because this improved accuracy.

4.4 Online Noisy Channel Decoding Setup

In order to set weights for the linear combination of model scores (Equation 1), we randomly sample a set of hyperparameters and evaluate each configuration on the development set (Yee et al., 2019; Ng et al., 2019). Hyperparameters are sampled within the interval $[0, 2]$, For direct models (`dir`), we sample ten random weights for the length penalty. For direct models combined with language models (`dir + lm`), we evaluate 100 randomly sampled configurations for the length penalty and the language model weight (λ_2). For direct models combined with language models and channel models (`dir + lm + ch`), we evaluate 1000 configurations of the length penalty, the language model weight (λ_2) and the channel model weight (λ_1). We use 16-bit floating point precision (Ott et al., 2018, 2019) for decoding with the online noisy channel setup.

Accuracy is measured via sacreBLEU (Post, 2018) for WMT German-English. We report the average BLEU of the newstest2014-2015 and newstest2017-2018 test sets, averaged over 3 random seeds for model weight initialization. Speed

[1]https://github.com/rsennrich/wmt16-scripts/tree/master/preprocess

	Total Params (M)	BLEU	Time (s)
Ensembles			
dir	283	38.8	20
2 dir	565	39.3	40
3 dir	848	39.5	59
Ensembles + LMs			
dir + lm	539	39.7	44
2 dir + lm	822	40.2	65
3 dir + lm	1104	40.3	84
Noisy Channel Modeling (Yee et al., 2019)			
dir + lm + big	822	40.5	550
Fast Noisy Channel Modeling (This work)			
dir + lm + 16th_1_1	542	40.2	56
dir + lm + base_1_1	574	40.5	92
2 dir + lm + 16th_1_1	824	40.5	76
2 dir + lm + base_1_1	857	40.8	111
3 dir + lm + 16th_1_1	1107	40.6	93
3 dir + lm + base_1_1	1140	41.0	131

Table 2: Fast noisy channel modeling is more accurate than ensembles at comparable speed and the two methods are additive. All results use beam size 5, batch sizes for each configuration are optimized and BLEU is averaged over news2014, news2015, news2017 and news2018 of WMT German to English.

	Channel Model Params (M)	BLEU	Time (s)
dir + lm + big	283	40.3	472
dir + lm + base	72	40.4	202
dir + lm + half	25	40.5	132
dir + lm + quarter	10	40.4	102
dir + lm + 8th	6	40.3	89
dir + lm + 16th	3	40.2	70
dir + lm + big_1_1	94	40.5	160
dir + lm + base_1_1	24	40.5	92
dir + lm + half_1_1	16	40.4	72
dir + lm + quarter_1_1	8	40.2	63
dir + lm + 8th_1_1	5	40.3	60
dir + lm + 16th_1_1	3	40.2	56

Table 3: Smaller channel models perform similarly for the standard beam size of 5. We exploit this fact to speed up noisy channel decoding.

is measured by the generation time (averaged over 3 trials) in seconds on the German-English newstest2016 test set on a 32GB Volta V100 GPU using 16-bit floating point precision (Ott et al., 2018, 2019). Unless otherwise specified, the beam size is 5, and the number of candidates for noisy channel model scoring per beam is $k_2 = 10$, unless otherwise specified. Generation times are based on a tuned batch size for each model configuration. We select the batch size within $(1, 10, 25, 50, 75, 100, 125, 150, 200, 300)$ that fits in memory and results in the fastest generation time.

5 Results

5.1 Fast Noisy Channel Modeling

In the first experiment, we evaluate the speed and accuracy of fast noisy channel decoding (§ 3) and compare to the naïve version without approximations (Yee et al., 2019). As additional baselines, we consider a single direct model (`dir`), ensembling two direct models (`2 dir`) and three direct models (`3 dir`), as well as adding a language model to each (`lm`). As channel models, we consider a `big` Transformer, a `base` Transformer, as well as a variant with model dimension of only 32 which is 1/16th of the model dimension of a `base` Transformer with a single layer in the encoder and decoder each (`16th_1_1`), totaling just 2.7M parameters. For fast noisy channel decoding, we reduce the channel model output vocabulary (§3.2) and set $k_2 = 3$; we ablate these choices in § 5.4.

Table 2 shows that the approximations we introduce to make noisy channel decoding fast also achieve similar accuracy (40.5 BLEU) to the much slower noisy channel approach of (Yee et al., 2019), while being about six times faster at inference time.

Table 2 also shows that `dir + lm + 16th_1_1` is 0.7 BLEU score better than `3 dir` at a similar decoding speed. Thus, using a small channel model and a language model with online noisy channel decoding is a better strategy than ensembling 3 direct models. Noisy channel decoding is also complementary to ensembling direct models: `3 dir + lm + base_1_1` improves by 0.7 BLEU compared to `3 dir + lm`.

Table 3 compares fast noisy channel decoding with different channel model sizes. Generally, smaller channel models are only slightly less accurate than larger models while being significantly faster than their larger counterparts. For example,

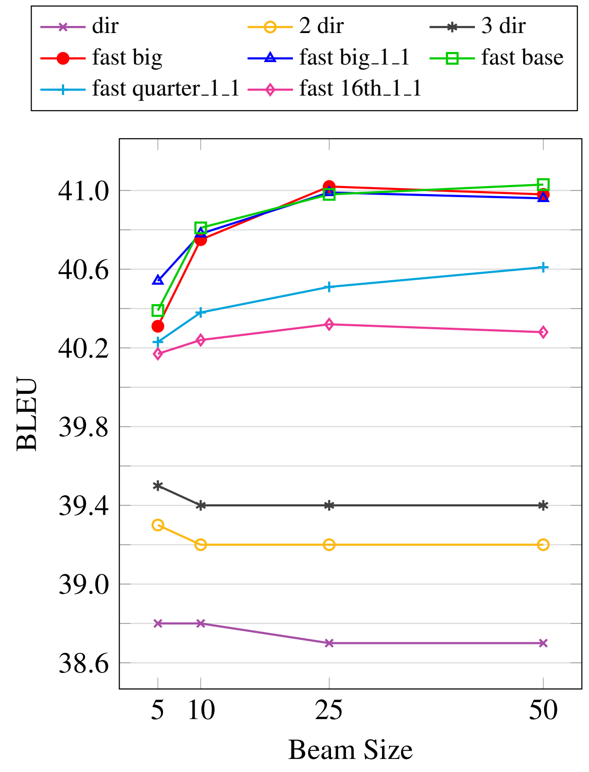

Figure 2: BLEU of fast online noisy channel decoding with different channel models when beam size is increased (compared to ensemble baselines). BLEU is averaged over news2014, news2015, news2017 and news2018 of WMT De-En.

`16th_1_1` is over eight times faster than `big` and achieves nearly the same accuracy.

This observation is in line with the hypothesis that the primary role of the channel model is to tie back the language model generations to the input. We exploit the fact that small channel models work well to make noisy channel decoding very fast.

5.2 Noisy Channel Decoding with Larger Beam Sizes

So far we used a standard beam size of five to enable fast decoding. However, previous work found that noisy channel modeling benefits more from larger beam sizes than other methods (Yee et al., 2019). Next, we evaluate whether our efficiency improvements still enable good performance with larger beam sizes.

Figure 2 shows that for beam size 5, most channel models perform comparably. Larger models are slightly better but overall they are in a similar ball park. As the beam size increases, larger channel models do achieve better accuracy. However, there is no difference between a single layer big model (`big_1_1`) and a six layer version (`big`).

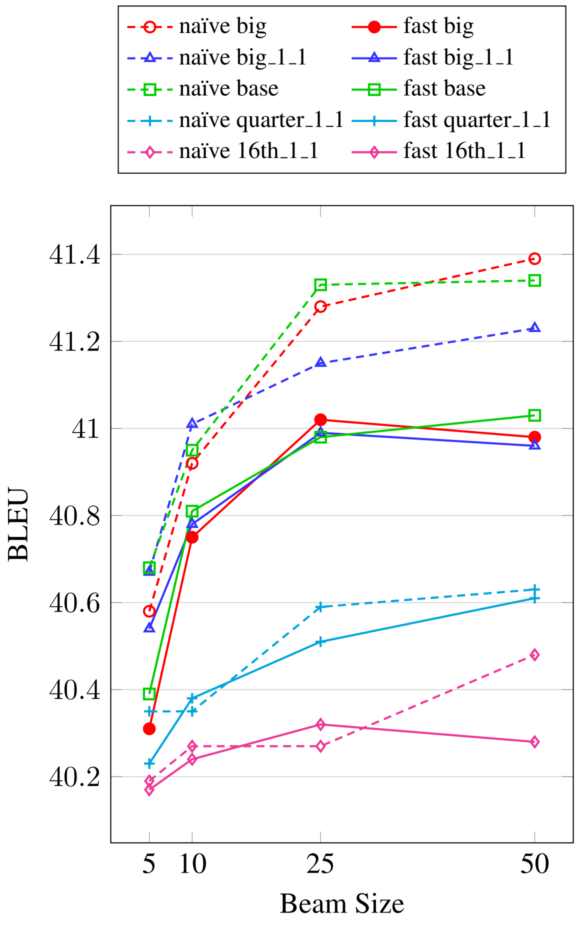

Figure 3: BLEU of fast and naïve online noisy channel decoding with different channel models sizes when beam size is increased. BLEU is averaged over news2014, news2015, news2017 and news2018 of WMT De-En.

As observed in previous work (Yee et al., 2019), the direct model and the direct ensembles (`dir`, `2 dir`, `3 dir`) do not benefit from larger beam sizes.

Next, we compare fast noisy channel decoding and naïve noisy channel decoding at larger beam sizes. As shown in Figure 4, the naïve approach is much slower. Fast approximations to noisy channel decoding scale much better in terms of speed as the beam size increases. Figure 3 compares the accuracy of fast noisy channel decoding at larger beam sizes with that of naïve noisy channel decoding. Using the `big` and `big_1_1` channel models gives the best performance across all beam sizes for naïve noisy channel decoding. With fast noisy channel decoding, we see an average drop of 0.3 BLEU and 0.2 BLEU for `big` and `big_1_1` respectively. On the other hand, for smaller channel models, the difference between naïve and fast noisy

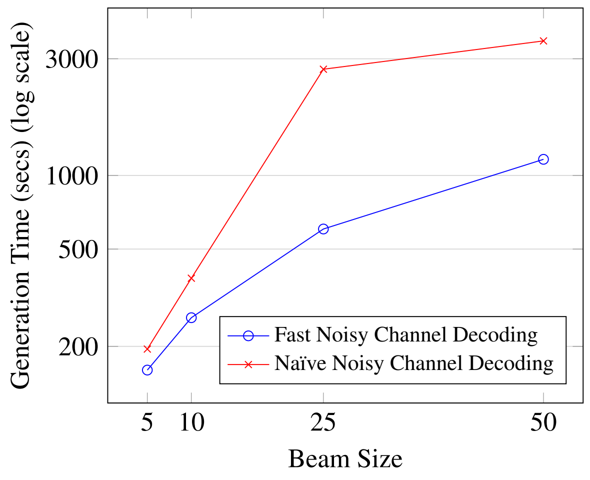

Figure 4: With larger beam sizes, the speed of fast approximations for noisy channel decoding scales much better than that of naïve noisy channel decoding. Results are based on generation using the big_1_1 channel model with the fastest batch size for each setting with beam 5 on newstest2016 De-En.

channel decoding is generally smaller.

5.3 Results on WMT Romanian-English

Next, we evaluate noisy channel modeling on WMT Romanian-English translation (Ro-En and En-Ro) which is a low resource setup compared to WMT German-English. We also compare to a recently introduced pre-training approach, mBART. The mBART model is pre-trained to denoise input sentences in multiple languages, followed by fine-tuning on the bitext (Liu et al., 2020). Following their setup for En-Ro evaluation, we apply Moses tokenization and normalize diacritics for Romanian (Sennrich et al., 2016a), and use tokenized BLEU. For Ro-En, we use SacreBLEU (Post, 2018).

Table 4 shows that noisy channel decoding with a wide beam can outperform multilingual pre-training (mBART) across the board. Large beams are not helpful for generation with mBART. Compared to the direct model, noisy channel decoding improves by 2.7/3.1 BLEU on En-Ro and Ro-En respectively, and increasing the beam size gives gains of 4.5/4.5 BLEU.

We also study the performance of noisy channel decoding on Romanian-English with back-translated data generated using unrestricted sampling (Edunov et al., 2018).[2] As compared to mBART02 (Liu et al., 2020), the previous state-of-the-art result on Romanian-English with back-translation, we achieve a 0.5 BLEU improvement. We use a similar number of total model parameters, but much less monolingual English data. Our English language model is trained on 4.5B tokens, while mBART02 uses 66B tokens of English and Romanian monolingual data.

Finally, Table 5 shows that fast approximations and smaller channel models achieve similar performance but much higher speed compared to naïve noisy channel decoding on WMT Romanian-English with back-translation. Fast noisy channel decoding with base_1_1 achieves comparable accuracy as mBART02 at slightly faster generation time with beam size 5.

5.4 Ablations

In this section we focus on some of the design choices we made to speed up noisy channel decoding. We measure the impact on speed and accuracy when reducing the output vocabulary size of the channel model, and reducing the number of beam candidates scored by the channel model.

5.4.1 Reducing the Output Vocabulary

In the next experiment, we compare the speed of using the full output vocabulary for the channel model to a reduced version. Specifically, we reduce the vocabulary by selecting all source tokens in the batch as well as the most frequent 500 tokens in the training data (see § 3.2). We tune each setup by selecting the fastest batch size based on a sweep over different batch sizes $(1, 10, 25, 50, 75, 100, 125, 150, 200, 300)$.

Table 6 shows that generating channel model scores for a small subset of the source vocabulary results in a small accuracy of up to 0.3 BLEU, but often less, while substantially increasing speed by 40-65% for single layer channel models and by 20-55% for other channel models. base_1_1 with a small vocabulary is nearly ten times faster than the approach proposed in Yee et al. (2019) (channel model size big), with a slight decline in accuracy.

The average vocabulary size used for scoring the channel model is around 1050, as compared to full source vocabulary size of 28,048. This leads to a large reduction in memory consumption and enables fitting larger batches into memory.

[2]The monolingual English data used for backtranslation comes from http://data.statmt.org/rsennrich/wmt16_backtranslations/ (Sennrich et al., 2016c).

	mono tokens Ro-En (B)	mono tokens En-Ro (B)	En-Ro	Ro-En	Ro-En +BT
mBART02	66	66	38.5	38.5	39.9
mBART02 (beam=50)	66	66	-	-	39.9
dir	-	-	34.6	34.6	38.4
dir + lm	4.5	22	35.4	35.9	38.7
dir + lm + big	4.5	22	37.3	37.7	39.6
dir + lm + big (beam=50)	4.5	22	**39.1**	**39.1**	**40.4**

Table 4: BLEU of noisy channel decoding on the Romanian-English newstest2016 test set with bitext-only as well as with backtranslation (BT) compared to mBART (Liu et al., 2020). We also show the total amount of monolingual data used by each method in billions of tokens.

	BLEU	Time (s)
mBART02	39.9	93
mBART02 (beam=50)	39.9	754
dir	38.4	19
Noisy Channel Modeling (Yee et al., 2019)		
dir + lm + big	39.6	1178
dir + lm + big (beam=50)	40.4	12554
Fast Noisy Channel Modeling		
dir + lm + base_1_1	39.8	82
dir + lm + base_1_1 (beam=50)	40.3	631

Table 5: Speed and accuracy on Romanian-English (Ro-En) with backtranslation. Fast noisy channel decoding using `base_1_1` achieves similar accuracy to mBART02 while being faster (beam=5). BLEU is measured on newstest2016 and generation time is measured on newsdev2016.

dir+ch+lm (beam=5)	Full Source Vocab		Small Source Vocab	
	BLEU	Time (s)	BLEU	Time (s)
big	40.6	1656	40.3	1355
base	40.7	854	40.4	516
half	40.6	450	40.5	299
quarter	40.5	359	40.4	212
8th	40.3	324	40.3	178
16th	40.1	264	40.2	118
big_1_1	40.7	543	40.5	339
base_1_1	40.5	336	40.5	169
half_1_1	40.3	264	40.4	117
quarter_1_1	40.4	238	40.2	95
8th_1_1	40.1	223	40.3	87
16th_1_1	40.2	209	40.2	74

Table 6: Comparison of accuracy (BLEU) and speed of online noisy channel decoding with and without the small output vocabulary approximation for different channel model sizes. Note we use $k_2 = 10$ for this ablation. BLEU is averaged over news2014, news2015, news2017 and news2018 of WMT De-En and generation time is on news2016.

5.4.2 Reducing the Number of Candidates

For each beam in each step of beam search, we need to make a choice about how many candidates k_2 we re-score with noisy channel modeling. Yee et al. (2019) re-scored $k_2 = 10$ candidates for each beam at each step. We sweep over different values of k_2 to understand the speed-accuracy trade-off associated with the choice of k_2. Table 7 shows that smaller values for k_2 are as accurate and much faster for beam size 5.

6 Conclusion

We introduced a number of approximations which greatly speed up noisy channel modeling for neural sequence to sequence models. This includes using channel models which are a fraction of the size of commonly used sequence to sequence models, pruning most of the channel model output vocabulary, and reducing the number of beam candidates scored by the channel model.

Our approximations are simple, yet, highly effective and enable comparable inference speed to ensembles of direct models while delivering higher accuracy. Our experiments show that noisy channel modeling can outperform pre-training approaches by being able to better exploit wider beams. Moreover, this is achieved while using a smaller amount of monolingual data.

k_2	BLEU	Time (s)
2	40.4	76
3	40.5	88
5	40.4	124
10	40.5	168

Table 7: Smaller number of rescoring candidates k_2 per beam are as accurate and much faster than larger values of k_2 for fast noisy channel decoding using `base_1_1` with beam 5. BLEU is averaged over news2014, news2015, news2017 and news2018 of WMT De-En and generation time is on news2016.

References

Ondrej Bojar and Ales Tamchyna. 2011. Improving translation model by monolingual data. In *Proc. of WMT*.

Peter F Brown, Vincent J Della Pietra, Stephen A Della Pietra, and Robert L Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational linguistics*.

Peng-Jen Chen, Jiajun Shen, Matt Le, Vishrav Chaudhary, Ahmed El-Kishky, Guillaume Wenzek, Myle Ott, and Marc'Aurelio Ranzato. 2019. Facebook ai's wat19 myanmar-english translation task submission. *WAT 2019*, page 112.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Unsupervised cross-lingual representation learning at scale. *arXiv*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proc. of NAACL*.

Sergey Edunov, Alexei Baevski, and Michael Auli. 2019. Pre-trained language model representations for language generation. In *Proc. of NAACL*.

Sergey Edunov, Myle Ott, Michael Auli, and David Grangier. 2018. Understanding back-translation at scale. In *Proc. of EMNLP*.

Maha Elbayad, Jiatao Gu, Edouard Grave, and Michael Auli. 2020. Depth-adaptive transformer. In *Proc. of ICLR*.

Angela Fan, Edouard Grave, and Armand Joulin. 2020. Reducing transformer depth on demand with structured dropout. In *Proc. of ICLR*.

Çaglar Gülçehre, Orhan Firat, Kelvin Xu, Kyunghyun Cho, Loïc Barrault, Huei-Chi Lin, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2015. On using monolingual corpora in neural machine translation. *arXiv*, abs/1503.03535.

Junxian He, Jiatao Gu, Jiajun Shen, and Marc'Aurelio Ranzato. 2020. Revisiting self-training for neural sequence generation. In *Proc. of ICLR*.

Jungo Kasai, Nikolaos Pappas, Hao Peng, James Cross, and Noah A. Smith. 2020. Deep encoder, shallow decoder: Reevaluating the speed-quality tradeoff in machine translation. *arXiv*, abs/2006.10369.

Dan Klein and Christopher Manning. 2001. Conditional structure versus conditional estimation in nlp. In *Proc. of EMNLP*.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, et al. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th annual meeting of the ACL on interactive poster and demonstration sessions*, pages 177–180. Association for Computational Linguistics.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proc. of NAACL*.

Gurvan L'Hostis, David Grangier, and Michael Auli. 2016. Vocabulary Selection Strategies for Neural Machine Translation. *arXiv*, abs/1610.00072.

Margaret Li, Stephen Roller, Ilia Kulikov, Sean Welleck, Y-Lan Boureau, Kyunghyun Cho, and Jason Weston. 2019. Don't say that! making inconsistent dialogue unlikely with unlikelihood training. *arXiv*.

Yinhan Liu, Jiatao Gu, Naman Goyal, Xian Li, Sergey Edunov, Marjan Ghazvininejad, Mike Lewis, and Luke Zettlemoyer. 2020. Multilingual denoising pre-training for neural machine translation. *arXiv preprint arXiv:2001.08210*.

Marco Lui and Timothy Baldwin. 2012. langid. py: An off-the-shelf language identification tool. In *Proceedings of the ACL 2012 system demonstrations*, pages 25–30.

Haitao Mi, Zhiguo Wang, and Abe Ittycheriah. 2016. Vocabulary manipulation for neural machine translation. In *Proc. of ACL*.

Nathan Ng, Kyra Yee, Alexei Baevski, Myle Ott, Michael Auli, and Sergey Edunov. 2019. Facebook fair's wmt19 news translation task submission. In *Proc. of WMT*.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proc. of NAACL System Demonstrations*.

Myle Ott, Sergey Edunov, David Grangier, and Michael Auli. 2018. Scaling neural machine translation. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 1–9.

Matt Post. 2018. A call for clarity in reporting bleu scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016a. Edinburgh neural machine translation systems for wmt 16. In *Proceedings of the First Conference on Machine Translation: Volume 2, Shared Task Papers*, pages 371–376.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016b. Improving neural machine translation models with monolingual data. In *Proc. of ACL*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016c. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 86–96.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016d. Neural machine translation of rare words with subword units. In *Proc. of ACL*.

Felix Stahlberg, James Cross, and Veselin Stoyanov. 2018. Simple fusion: Return of the language model. In *Proc. of WMT*.

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.

Guillaume Wenzek, Marie-Anne Lachaux, Alexis Conneau, Vishrav Chaudhary, Francisco Guzmán, Armand Joulin, and Édouard Grave. 2020. Ccnet: Extracting high quality monolingual datasets from web crawl data. In *Proceedings of The 12th Language Resources and Evaluation Conference*, pages 4003–4012.

Felix Wu, Angela Fan, Alexei Baevski, Yann N. Dauphin, and Michael Auli. 2019. Pay less attention with lightweight and dynamic convolutions. In *Proc. of ICLR*.

Kyra Yee, Yann N Dauphin, and Michael Auli. 2019. Simple and effective noisy channel modeling for neural machine translation. In *Proc. of NAACL*.

Lei Yu, Phil Blunsom, Chris Dyer, Edward Grefenstette, and Tomás Kociský. 2017. The neural noisy channel. In *Proc. of ICLR*.

Lei Yu, Laurent Sartran, Wojciech Stokowiec, Wang Ling, Lingpeng Kong, Phil Blunsom, and Chris Dyer. 2020. Better document-level machine translation with bayes' rule. *TACL*.