THE NEURAL NOISY CHANNEL

Lei Yu[1]*, **Phil Blunsom**[1,2], **Chris Dyer**[2], **Edward Grefenstette**[2], **and Tomáš Kočiský**[1,2]
[1]University of Oxford and [2]DeepMind
lei.yu@cs.ox.ac.uk, {pblunsom,cdyer,etg,tkocisky}@google.com

ABSTRACT

We formulate sequence to sequence transduction as a noisy channel decoding problem and use recurrent neural networks to parameterise the source and channel models. Unlike direct models which can suffer from explaining-away effects during training, noisy channel models must produce outputs that explain their inputs, and their component models can be trained with not only paired training samples but also unpaired samples from the marginal output distribution. Using a latent variable to control how much of the conditioning sequence the channel model needs to read in order to generate a subsequent symbol, we obtain a tractable and effective beam search decoder. Experimental results on abstractive sentence summarisation, morphological inflection, and machine translation show that noisy channel models outperform direct models, and that they significantly benefit from increased amounts of unpaired output data that direct models cannot easily use.

1 INTRODUCTION

Recurrent neural network sequence to sequence models (Kalchbrenner & Blunsom, 2013; Sutskever et al., 2014; Bahdanau et al., 2015) are excellent models of $p(\text{output sequence } \boldsymbol{y} \mid \text{input sequence } \boldsymbol{x})$, provided sufficient input–output $(\boldsymbol{x}, \boldsymbol{y})$ pairs are available for estimating their parameters. However, in many domains, vastly more unpaired output examples are available than input–output pairs (e.g., transcribed speech is relatively rare although non-spoken texts are abundant; Swahili–English translations are rare although English texts are abundant; etc.). A classic strategy for exploiting both kinds of data is to use Bayes' rule to rewrite $p(\boldsymbol{y} \mid \boldsymbol{x})$ as $p(\boldsymbol{x} \mid \boldsymbol{y})p(\boldsymbol{y})/p(\boldsymbol{x})$, a factorisation which is called a **noisy channel model** (Shannon, 1948). A noisy channel model thus consists of two component models: the conditional **channel model**, $p(\boldsymbol{x} \mid \boldsymbol{y})$, which characterizes the *reverse* transduction problem and whose parameters are estimated from the paired $(\boldsymbol{x}, \boldsymbol{y})$ samples, and the unconditional **source model**, $p(\boldsymbol{y})$, whose parameters are estimated from both the paired and (usually much more numerous) unpaired samples.[1]

Beyond their data omnivorousness, noisy channel models have other benefits. First, the two component models mean that two different aspects of the transduction problem can be addressed independently. For example, in many applications, source models are language models and innovations in these can be leveraged to obtain improvements in any system that uses them as a component. Second, the component models can have complementary strengths, since inference is carried out in the product space; this simplifies design because a single model does not have to get everything perfectly right. Third, the noisy channel operates by selecting outputs that both are *a priori* likely *and* that explain the input well. This addresses a failure mode that can occur in conditional models in which inputs are "explained away" by highly predictive output prefixes, resulting in poor training (Klein & Manning, 2001). Since the noisy channel formulation requires its outputs to explain the observed input, this problem is avoided.

In principle, the noisy channel decomposition is straightforward; however, in practice, decoding (i.e., computing $\arg\max_{\boldsymbol{y}} p(\boldsymbol{x} \mid \boldsymbol{y})p(\boldsymbol{y})$) is a significant computational challenge, and tractability concerns impose restrictions on the form the component models can take. To illustrate, an appealing parameterization would be to use an attentional seq2seq network (Bahdanau et al., 2015) to model

*Work completed at DeepMind.

[1]We do not model $p(\boldsymbol{x})$ since, in general, we will be interested in finding $\arg\max_{\mathbf{y}} p(\mathbf{y} \mid \mathbf{x})$, and $\arg\max_{\mathbf{y}} p(\mathbf{y} \mid \mathbf{x}) = \arg\max_{\mathbf{y}} \frac{p(\mathbf{x}|\mathbf{y})p(\mathbf{y})}{p(\mathbf{x})} = \arg\max_{\mathbf{y}} p(\mathbf{x} \mid \mathbf{y})p(\mathbf{y})$.

the channel probability $p(\boldsymbol{x} \mid \boldsymbol{y})$. However, seq2seq models are designed under the assumption that the complete conditioning sequence is available before any prefix probabilities of the output sequence can be computed. This assumption is problematic for channel models since it means that a complete output sequence must be constructed before the channel model can be evaluated (since the channel model conditions on the output). Therefore, to be practical, the channel probability must decompose in terms of prefixes of the conditioning variable, $\boldsymbol{y}$. While the chain rule justifies decomposing output variable probabilities in terms of successive extensions of a partial prefix, no such convenience exists for conditioning variables, and approximations must be introduced.

In this work, we use a variant of the newly proposed online seq2seq model of Yu et al. (2016) which uses a latent alignment variable to enable its probabilities to factorize in terms of prefixes of both the input and output, making it an appropriate channel model (§2). Using this channel model, the decoding problem then becomes similar to the problem faced when decoding with direct models (§3). Experiments on abstractive summarization, machine translation, and morphological inflection show that the noisy channel can significantly improve performance and exploit unpaired output training samples and that models that combine the direct model and a noisy channel model offer further improvements still (§4).

2 Background: Segment to Segment Neural Transduction

Our model is based on the Segment to Segment Neural Transduction model (SSNT) of Yu et al., 2016. At a high level, the model alternates between encoding more of the input sequence and decoding output tokens from the encoded representation. This presentation deviates from the Yu et al.'s presentation so as to emphasize the incremental construction of the conditioning context that is enabled by the latent variable.

2.1 Model description

Similar to other neural sequence to sequence models, SSNT models the conditional probability $p(\boldsymbol{y} \mid \boldsymbol{x})$ of a output sequence $\boldsymbol{y}$ given a input sequence $\boldsymbol{x}$.

To avoid having to observe the complete input sequence $\boldsymbol{x}$ before making a prediction of the beginning of the output sequence, we introduce a latent alignment variable $\boldsymbol{z}$ which indicates when each token of the output sequence is to be generated as the input sequence is being read. Since we assume that the input is read just once from left to right, we restrict $\boldsymbol{z}$ to be a monotonically increasing alignment (i.e., $z_{j+1} \geq z_j$ is true with probability 1), where $z_j = i$ denotes that the output token at position j (y_j) is generated when the input sequence up through position i is has been read. The SSNT model is:

$$\begin{aligned} p(\boldsymbol{y} \mid \boldsymbol{x}) &= \sum_{\boldsymbol{z}} p(\boldsymbol{y}, \boldsymbol{z} \mid \boldsymbol{x}) \\ p(\boldsymbol{y}, \boldsymbol{z} \mid \boldsymbol{x}) &\approx \prod_{j=1}^{|\boldsymbol{y}|} \underbrace{p(z_j \mid z_{j-1}, \boldsymbol{x}_1^{z_j}, \boldsymbol{y}_1^{j-1})}_{\text{alignment probability}} \underbrace{p(y_j \mid \boldsymbol{x}_1^{z_j}, \boldsymbol{y}_1^{j-1})}_{\text{word probability}}. \end{aligned} \tag{1}$$

We explain the model in terms of its two components, starting with the word generation term. In the SSNT, the input and output sequences $\boldsymbol{x}$, $\boldsymbol{y}$ are encoded with two separate LSTMs (Hochreiter & Schmidhuber, 1997), resulting in sequences of hidden states representing prefixes of these sequences. In Yu et al.'s formulation, the input sequence encoder (i.e., the conditioning context encoder) can either be a unidirectional or bidirectional LSTM, but here we assume that it is a unidirectional LSTM, which ensures that it will function well as a channel model that can compute probabilities with incomplete conditioning contexts (this is necessary since, at decoding time, we will be constructing the conditioning context incrementally). Let $\mathbf{h}_i$ represent the input sequence encoding for the prefix $\boldsymbol{x}_1^i$. Since the final action at timestep j will be to predict y_j, it is convenient to let $\mathbf{s}_j$ denote the hidden state that excludes y_j, i.e., the encoding of the prefix $\boldsymbol{y}_1^{j-1}$.

The probability of the next token y_j is calculated by concatenating the aligned hidden state vectors $\mathbf{s}_j$ and $\mathbf{h}_{z_j}$ followed by a softmax layer,

$$p(y_j \mid \boldsymbol{x}_1^{z_j}, \boldsymbol{y}_1^{j-1}) \propto \exp(\mathbf{W}_w[\mathbf{h}_{z_j}; \mathbf{s}_j] + \mathbf{b}_w).$$

The model thus depends on the current alignment position z_j, which determines how far into $\boldsymbol{x}$ it has read.

We now discuss how the sequence of z_j's are generated. First, we remark that modelling this distribution requires some care so as to avoid conditioning on the entire input sequence. To illustrate why one might induce a dependency on the entire input sequence in this model, it is useful to compare to a standard attention model. Attention models operate by computing a score using a representation of alignment candidate (in our case, the candidates would be every unread token remaining in the input). If we followed this strategy, it would be necessary to observe the full input sequence when making the first alignment decision.

We instead model the alignment transition from timestep j to $j+1$ by decomposing it into a sequence of conditionally independent SHIFT and EMIT operations that progressively decide whether to read another token or stop reading. That is, at input position i, the model decides to EMIT, i.e., to set $z_j = i$ and predict the next output token y_j from the word model, or it decides to SHIFT, i.e., to read one more input token and increment the input position $i \leftarrow i + 1$. The probability $p(a_{i,j} = \text{EMIT} \mid \boldsymbol{x}_1^i, \boldsymbol{y}_1^{j-1})$ is calculated using the encoder and decoder states defined above as:

$$p(a_{i,j} = \text{EMIT} \mid \boldsymbol{x}_1^i, \boldsymbol{y}_1^{j-1}) = \sigma(\text{MLP}(\mathbf{W}_t[\mathbf{h}_i; \mathbf{s}_j] + b_t)).$$

The probability of SHIFT is simply $1 - p(a_{i,j} = \text{EMIT})$. In this formulation, the probabilities of aligning z_j to each alignment candidate i can be computed by reading just $\boldsymbol{x}_1^i$ (rather than the entire sequence). The probabilities are also independent of the contents of the suffix $\boldsymbol{x}_{i+1}^{|\boldsymbol{x}|}$.

Using the probabilities of the auxiliary $a_{i,j}$ variables, the alignment probabilities needed in Eq. 1 are computed as:

$$p(z_j = i \mid z_{j-1}, \boldsymbol{y}_1^{j-1}, \boldsymbol{x}_1^i) = \begin{cases} 0 & \text{if } i < z_{j-1} \\ p(a_{i,j} = \text{EMIT}) & \text{if } i = z_{j-1} \\ \left(\prod_{i'=z_{j-1}}^{i-1} p(a_{i',j} = \text{SHIFT})\right) p(a_{i,j} = \text{EMIT}) & \text{if } i > z_{j-1} \end{cases}$$

2.2 Inference algorithms

In SSNT, the probability of generating each y_j depends only on the current output position's alignment (z_j), the current output prefix ($\boldsymbol{y}_1^{j-1}$), the input prefix up to the current alignment ($\boldsymbol{x}_1^{z_j}$). It does *not* depend on the history of the alignment decisions. Likewise, the alignment decisions at each position are also conditionally independent of the history of alignment decisions. Because of these independence assumptions, $\boldsymbol{z}$ can be marginalised using a $O(|\boldsymbol{x}|^2 \cdot |\boldsymbol{y}|)$ time dynamic programming algorithm where each fill in a chart with computing the following marginal probabilities:

$$\alpha(i, j) = p(z_j = i, \boldsymbol{y}_1^j \mid \boldsymbol{x}_1^{z_j}) = \sum_{i'=1}^{i} \alpha(i', j-1) \underbrace{p(z_j \mid z_{j-1}, \boldsymbol{x}_1^{z_j}, \boldsymbol{y}_1^{j-1})}_{\text{alignment probability}} \underbrace{p(y_j \mid \boldsymbol{x}_1^{z_j}, \boldsymbol{y}_1^{j-1})}_{\text{word probability}}.$$

The model is trained to minimize the negative log likelihood of the parallel corpus S:

$$\begin{aligned} \mathcal{L}(\boldsymbol{\theta}) &= -\sum_{(\boldsymbol{x},\boldsymbol{y}) \in S} \log p(\boldsymbol{y} \mid \boldsymbol{x}; \boldsymbol{\theta}) \\ &= -\sum_{(\boldsymbol{x},\boldsymbol{y}) \in S} \log \alpha(|\boldsymbol{x}|, |\boldsymbol{y}|). \end{aligned} \tag{2}$$

The gradients of this objective with respect to the component probability models can be computed using automatic differentiation or using a secondary dynamic program that computes 'backward' probabilities. We refer the reader to Section 3.1 of Yu et al. (2016) for details.

In this paper, we use a slightly different objective from the one described in Yu et al. (2016). Rather than marginalizing over the paths that end in any possible input positions $\sum_{i=1}^{I} \alpha(i, |\boldsymbol{y}|)$, we require that the full input be consumed when the final output symbol is generated. This constraint biases away from predicting outputs without explaining them using the input sequence.

3 Decoding

We now turn to the problem of decoding, that is, of computing

$$\hat{\boldsymbol{y}} = \arg\max_{\boldsymbol{y}} p(\boldsymbol{x} \mid \boldsymbol{y}) p(\boldsymbol{y}),$$

where we are using the SSNT model described in the previous section as the channel model and a language model that delivers prior probabilities of the output sequence in left-to-right order, i.e., $p(y_i \mid \boldsymbol{y}^{i-1})$.

Marginalizing the latent variable during search is computationally hard (Sima'an, 1996), and so we approximate the search problem as

$$\hat{\boldsymbol{y}} = \arg\max_{\boldsymbol{y}} \max_{\boldsymbol{z}} p(\boldsymbol{x}, \boldsymbol{z} \mid \boldsymbol{y}) p(\boldsymbol{y}).$$

However, even with this simplification, the search problem remains nontrivial. On one hand, we must search over the space of all possible outputs with a model that makes no Markovian assumptions. This is similar to the decoding problem faced in standard seq2seq transducers. On the other hand, our model computes the probability of the given input conditional on the predicted output hypothesis. Therefore, instead of just relying on a single softmax to provide a probability for every output word type (as we conveniently can in the direct model), we must loop over each output word type, and run a softmax over the input vocabulary—a computational expense that is quadratic in the size of the vocabulary!

To reduce this computational effort, we make use of an auxiliary direct model $q(\boldsymbol{y}, \boldsymbol{z} \mid \boldsymbol{x})$ to explore probable extensions of partial hypotheses, rather than trying to perform an exhaustive search over the vocabulary each time we extend an item on the beam.

Algorithm 1, in Appendix A, describes the decoding algorithm based on a formulation by Tillmann et al. (1997). The idea is to create a matrix Q of partial hypotheses. Each hypothesis in cell (i, j) covers the first i words of the input ($\boldsymbol{x}_1^i$) and corresponds to an output hypothesis prefix of length j ($\boldsymbol{y}_1^j$). The hypothesis is associated with a model score. For each cell (i, j), the direct proposal model first calculates the scores of possible extensions of previous cells that could then reach (i, j) by considering every token in the output vocabulary, from all previous candidate cells $(i-1, \leq j)$. The top K_1 partial output sequences. These partial output sequences are subsequently rescored by the noisy channel model, and the K_2 best candidates are kept in the beam and used for further extension. The beam size K_1 and K_2 are hyperparameters to be tuned in the experiments.

3.1 Model combination

The decoder we have just described makes use of an auxiliary decoding model. This means that, as a generalisation, it is capable of decoding under an objective that is a linear combination of the direct model, channel model, language model and a bias for the output length,

$$O_{\boldsymbol{x}_1^i, \boldsymbol{y}_1^j} = \lambda_1 \log p(\boldsymbol{y}_1^j \mid \boldsymbol{x}_1^i) + \lambda_2 \log p(\boldsymbol{x}_1^i \mid \boldsymbol{y}_1^j) + \lambda_3 \log p(\boldsymbol{y}_1^j) + \lambda_4 |\boldsymbol{y}_1^j|. \tag{3}$$

The bias is used to penalize the noisy channel model for generating too-short (or long) sequences. The λ's are hyperparameters to be tuned using on a small amount of held-out development data.

4 Experiments

We evaluate our model on three natural language processing tasks, abstractive sentence summarisation, machine translation and morphological inflection generation. For each task, we compare the performance of the direct model, noisy channel model, and the interpolation of the two models.

4.1 Abstractive Sentence Summarisation

Sentence summarisation is the problem of constructing a shortened version of a sentence while preserving the majority of its meaning. In contrast to extractive summarisation, which can only copy words from the original sentence, abstractive summarisation permits arbitrary rewording of the

Model	# Parallel data	# Data for LM	RG-1	RG-2	RG-L
direct (uni)*	1.0m	-	30.94	14.20	28.72
direct (bi)	1.0m	-	30.78	14.67	28.57
direct (bi)	3.8m	-	33.82	16.66	31.50
channel + LM + bias (uni)*	1.0m	1.0m	31.92	14.75	29.58
channel + LM + bias (bi)	1.0m	1.0m	31.96	14.89	29.51
direct + channel + LM + bias (uni)	1.0m	1.0m	33.07	15.21	30.29
direct + channel + LM + bias (bi)	1.0m	1.0m	33.18	15.65	30.45
channel + LM + bias (uni)*	1.0m	3.8m	32.59	15.05	30.06
channel + LM + bias (bi)	1.0m	3.8m	32.51	15.00	29.90
direct + channel + LM + bias (uni)	1.0m	3.8m	33.16	15.63	30.53
direct + channel + LM + bias (bi)	1.0m	3.8m	33.35	15.77	30.68
chanel + LM + bias (bi)	3.8m	3.8m	34.12	16.41	31.38
direct + channel + LM + bias (bi)	3.8m	3.8m	**34.41**	**16.86**	**31.83**

Table 1: ROUGE F1 scores on the sentence summarisation test set. The ‘uni’ and ‘bi’ in the parentheses denotes the encoder for the model proposing candidates is a unidirectional LSTM or bidirectional LSTM. Those rows marked with an $*$ denote models that process their input online.

sentence. The dataset (Rush et al., 2015) that we use is constructed by pairing the first sentence and the headline of each article from the annotated Gigaword corpus (Graff et al., 2003; Napoles et al., 2012). There are 3.8m, 190k and 381k sentence pairs in the training, validation and test sets, respectively. Yu et al. (2016) filtered the dataset by restricting the lengths of the input and output sentences to be no greater than 50 and 25 tokens, respectively. From the filtered data, they further sampled 1 million sentence pairs for training. We experimented on training the direct model and channel model with both the sampled 1 million and the full 3.8 million parallel data. The language model is trained on the target side of the parallel data, i.e. the headlines. We evaluated the generated summaries of 2000 randomly sampled sentence pairs using full length ROUGE F1. This setup is in line with the previous work on this task (Rush et al., 2015; Chopra et al., 2016; Gülçehre et al., 2016; Yu et al., 2016).

The same configuration is used to train the direct model and the channel model. The loss (Equation 2) is optimized by Adam (Kingma & Ba, 2015), with initial learning rate of 0.001. We use LSTMs with 1 layer for both the encoder and decoders, with hidden units of 256. The mini-batch size is 32, and dropout of 0.5 is applied to the input and output of LSTMs. For the language model, we use a 2-layer LSTM with 1024 hidden units and 0.5 dropout. The learning rate is 0.0001. All the hyperparameters are optimised via grid search on the perplexity of the validation set. During decoding, beam search is employed with the number of proposals generated by the direct model $K_1 = 20$, and the number of best candidates selected by the noisy channel model $K_2 = 10$.

Table 1 presents the ROUGE-F1 scores of the test set from the direct model, noisy channel model (channel + LM + bias) and the interpolation of the direct model and the noisy channel model (direct + channel + LM + bias) trained on different sizes of data. The noisy channel model with the language model trained on the target side of the 1 million parallel data outperforms the direct model by approximately 1 point. Such improvement indicates that the language model helps improve the quality of the output sequence when no extra unlabelled data is available. Training the language model with all the headlines in the dataset, i.e. 3.8 million sentences, gives a further boost to the ROUGE score. This is in line with our expectation that the model benefits from adding large amounts of unlabelled data. The interpolation of the direct model, channel model, language model and bias of the output length achieves the best results — the ROUGE score is close to the direct model trained on all the parallel data. Although there is still improvement, when the direct model is trained with more data, the gap between the direct model and the noisy channel model is smaller.

Table 2 surveys published results on this task, and places our best models in the context of the current state-of-the-art results. ABS+ (Rush et al., 2015), RAS-LSTM and RAS-Elman (Chopra et al., 2016) are different variations of the attentive models. *Pointing the unkown words* uses pointer networks (Vinyals et al., 2015) to select the output token from the input sequence in order to avoid generating unknown tokens. ASC + FSC (Miao & Blunsom, 2016) is a semi-supervised model

Model	# Parallel data	# Unpaired data	RG-1	RG-2	RG-L
ABS+	3.8m	-	29.55	11.32	26.42
RAS-LSTM	3.8m	-	32.55	14.70	30.03
RAS-Elman	3.8m	-	33.78	15.97	31.15
Pointing unkown words	3.8m	-	**35.19**	16.66	**32.51**
ASC + FSC	1.0m	3.8m	31.09	12.79	28.97
ASC + FSC	3.8m	3.8m	34.17	15.94	31.92
direct + channel + LM + bias (bi)	1.0m	3.8m	33.35	15.77	30.68
direct + channel + LM + bias (bi)	3.8m	3.8m	34.41	**16.86**	31.83

Table 2: Overview of results on the abstractive sentence summarisation task. ABS+ (Rush et al., 2015) is the attentive model with bag-of-words as the encoder. RAS-LSTM and RAS-Elman (Chopra et al., 2016) are the sequence to sequence models with attention with the RNN cell implemented as LSTMs and an Elman architecture (Elman, 1990), respectively. Pointing the unknown words (Gülçehre et al., 2016) uses pointer networks (Vinyals et al., 2015) to select the output token from the input sequence in order to avoid generating unknown tokens. ASC + FSC (Miao & Blunsom, 2016) is the semi-supervised model based on a variational autoencoder.

based on a variational autoencoder. Trained on 1m paired samples and 3.8m unpaired samples, the noisy channel achieves comparable or better results than (direct) models trained with 3.8m paired samples. Compared to Miao & Blunsom (2016), whose ASC + FSC models is an alternative strategy for using unpaired data, the noisy channel is significantly more effective — 33.35 versus 31.09 in ROUGE-1.

Finally, motivated by the qualitative observation that noisy channel model outputs were quite fluent and often used reformulations of the input rather than a strict compression (which would be poorly scored by ROUGE), we carried out a human preference evaluation whose results are summarised in Table 3. This confirms that noisy channel summaries are strongly preferred over those of the direct model.

Model	count
both bad	188
both good	106
direct $>$ noisy channel	135
noisy channel $>$ direct	**212**

Table 3: Preference ratings for 641 segments from the test set (each segment had ratings from at least 2 raters with $\geq$ 50% agreement on the label and where one label had a plurality of the votes).

4.2 Machine Translation

We next evaluate our models on a Chinese–English machine translation task. We used parallel data with 184k sentence pairs (from the FBIS corpus, LDC2003E14) and monolingual data with 4.3 million of English sentences (selected from the English Gigaword). The training data is preprocessed by lowercasing the English sentences, replacing digits with '#' token, and replacing tokens appearing less than 5 times with an UNK token. This results in vocabulary sizes of 30k and 20k for Chinese sentences and English sentences, respectively.

The models are trained using Adam (Kingma & Ba, 2015) with initial learning rate of 0.001 for the direct model and the channel model, and 0.0001 for the language model. The LSTMs for the direct and channel models have 512 hidden units and 1 layer, and 2 layers with 1024 hidden units per layer for the language model. Dropout of 0.5 on the input and output of LSTMs is set for all the model training. The noisy channel decoding uses K_1 = 20 and K_2 = 10 as the beam sizes.

Table 4 lists the translation performance of different models in BLEU scores. To set a benchmark, we train the attentional sequence to sequence model (Bahdanau et al., 2015) using the same parallel data. For direct models, we leverage bidirectional LSTMs as the encoder for this task. We can see

Model	BLEU
seq2seq w/ attention	19.27
direct (bi)	19.92
channel + LM + bias (bi)	22.21
direct + channel + LM + bias (bi)	**23.06**

Table 4: BLEU scores from different models for the Chinese to English machine translation task.

that the noisy channel model is approximately 2 points higher in BLEU, and the combination of nosiy channel and direct model gives extra boost. This pattern is the same as the results we obtained from the sentence summarisation task.

4.3 Morphological Inflection Generation

Morphological inflection is the task of generating a target (inflected form) word from a source word (base form), given a morphological attribute, e.g. number, tense, and person etc.. It is useful for reducing data sparsity issues in translating morphologically rich languages. The transformation from the base form to the inflected form is usually to add prefix or suffix, or to do character replacement. The dataset (Durrett & DeNero, 2013) that we use in the experiments is created from Wiktionary, including inflections for German nouns, German verbs, Spanish Verbs, Finnish noun and adjective, and Finnish verbs. We only experimented on German nouns and German verbs, as German nouns is the most difficult task[2], and the direct model does not perform as well as other state-of-the-art systems on German verbs. The train/dev/test split for German nouns is 2364/200/200, and for German verbs is 1617/200/200. There are 8 and 27 inflection types in German nouns and German verbs, respectively. Following previous work, we learn a separate model for each type of inflection independent of the other inflections. We report results on the average accuracy across different inflections. Our language models were trained on word types extracted by running a morphological analysis tool on the WMT 2016 monolingual data and extracting examples of appropriately inflected word forms.[3] After annotation the number of instances for training the language model ranged from 300k to 3.8m for different inflection types in German nouns, and from 200 to 54k in German verbs.

The experimental setup that we use on this task is K_1 = 60, K_2 = 30,

- direct and channel model: 1 layer LSTM with 128 hidden, $\eta = 0.001$, dropout = 0.5.
- language model: 2 layer LSTM with 512 hidden, $\eta = 0.0001$, dropout = 0.5.

Table 1 summarises the results from our models. On both datasets, the noisy channel model (channel + LM + bias) does not perform as well as the direct model, but the interpolation of the direct model and noisy channel model (direct + channel + LM + bias) significantly outperforms the direct model. For further comparison, we also included the state-of-the-art results as benchmarks. NCK15 (Nicolai et al., 2015) tackles the task based on the three-stage approach: (1) align the source and target word, (2) extract inflection rules, (3) apply the rule to new examples. FTND16 (Faruqui et al., 2016) is based on neural sequence to sequence models. Both models (NCK15+ and FTND16+) rerank the candidate outputs by the scores predicted from n-gram language models, together with other features.

5 Analysis

By observing the output generated by the direct model and noisy channel model, we find (in line with theoretical critiques of conditional models) that the direct model may leave out key information. By contrast, the noisy channel model does seem to avoid this issue. To illustrate, in Example 1 (see Appendix B) in Table 5, the direct model ignores the key phrase ‘coping with’, resulting in incomplete meaning, but the noisy channel model covers it. Similarly, in Example 4, the direct

[2]While state-of-the-art systems can achieve 99% accuracies on Spanish verbs and Finnish verbs, they can only get 89% accuracy on German nouns.

[3]`http://www.statmt.org/wmt16/translation-task.html`

Model	Acc.
NCK15	88.60
FTND16	88.12
NCK15+	89.90
FTND16+	89.31
direct (uni)	82.25
direct (bi)	87.68
channel + LM + bias (uni)	78.38
channel + LM + bias (bi)	78.13
direct + channel + LM + bias (uni)	88.44
direct + channel + LM + bias (bi)	**90.94**

(a)

Model	Acc.
NCK15	97.50
FTND16	**97.92**
NCK15+	97.90
FTND16+	97.11
direct (uni)	87.85
direct (bi)	94.83
channel + LM + bias (uni)	84.42
channel + LM + bias (bi)	92.13
direct + channel + LM + bias (uni)	92.20
direct + channel + LM + bias (bi)	97.15

(b)

Figure 1: Accuracy on morphological inflection of German nouns (a), and German verbs (b). NCK15 (Nicolai et al., 2015) and FTND16 (Faruqui et al., 2016) are previous state-of-the-art on this task, with NCK15 based on feature engineering, and FTND16 based on neural networks. NCK15+ and FTND16+ are the semi-supervised setups of these models.

model does not translate the Chinese word corresponding to 'investigation'. We also observe that while the direct model mostly copies words from the source sentence, the noisy channel model prefers generating paraphrases. For instance, in Example 2, while the direct model copies the word 'accelerate' in the generated output, the noisy channel model generate 'speed up' instead. While one might argue that copying is a preferable compression technique than paraphrasing (as long as it produces grammatical outputs), it does show the power of these models.

6 Related work

Noisy channel decompositions have been successfully used in a variety of problems, including speech recognition (Jelinek, 1998), machine translation (Brown et al., 1993), spelling correction (Brill & Moore, 2000), and question answering (Echihabi & Marcu, 2003). The idea of adding language models and monolingual data in machine translation has been explored in earlier work. Gülçehre et al. (2015) propose two strategies of combining a language model with a neural sequence to sequence model. In shallow fusion, during decoding the sequence to sequence model (direct model) proposes candidate outputs and these candidates are reranked based on the scores calculated by a weighted sum of the probability of the translation model and that of the language model. In deep fusion, the language model is integrated into the decoder of the sequence to sequence model by concatenating their hidden state at each time step. Sennrich et al. (2016) generates additional parallel data by doing backward-translation from monolingual data. Significant performance gains are obtained by training with much larger (lower quality) parallel data. Cheng et al. (2016) leverages the abundant monolingual data by doing multitask learning with an autoencoding objective. Another trend of work that is related to our model is the investigation of making online prediction for machine translation (Gu et al., 2016; Grissom II et al., 2014; Sankaran et al., 2010) and speech recognition (Hwang & Sung, 2016; Jaitly et al., 2016).

7 Conclusion

We have presented and empirically validated a noisy channel transduction model that uses component models based on recurrent neural networks. This formulation lets us use unpaired outputs to estimate the parameters of the source model and input-output pairs to train the channel model. Despite the channel model's ability to condition on long sequences, we are able to maintain tractable decoding by using a latent segmentation variable that breaks the conditioning context up into a series of monotonically growing segments. Our experiments show that this model makes excellent use of unpaired training data.

REFERENCES

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

Eric Brill and Robert C. Moore. An improved error model for noisy channel spelling correction. In *Proc. ACL*, 2000.

Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, and Robert. L. Mercer. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19: 263–311, 1993.

Yong Cheng, Wei Xu, Zhongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu. Semi-supervised learning for neural machine translation. In *Proc. ACL*, 2016.

Sumit Chopra, Michael Auli, and Alexander M. Rush. Abstractive sentence summarization with attentive recurrent neural networks. In *Proc. NAACL*, 2016.

Greg Durrett and John DeNero. Supervised learning of complete morphological paradigms. In *HLT-NAACL*, 2013.

Abdessamad Echihabi and Daniel Marcu. A noisy-channel approach to question answering. In *Proc. ACL*, 2003.

Jeffrey L. Elman. Finding structure in time. *Cognitive science*, 14(2):179–211, 1990.

Manaal Faruqui, Yulia Tsvetkov, Graham Neubig, and Chris Dyer. Morphological inflection generation using character sequence to sequence learning. In *HLT-NAACL*, 2016.

David Graff, Junbo Kong, Ke Chen, and Kazuaki Maeda. English gigaword. *Linguistic Data Consortium, Philadelphia*, 2003.

Alvin Grissom II, Jordan Boyd-Graber, He He, John Morgan, and Hal Daumé III. Don't until the final verb wait: Reinforcement learning for simultaneous machine translation. In *Empirical Methods in Natural Language Processing*, 2014.

Jiatao Gu, Graham Neubig, Kyunghyun Cho, and Victor O.K. Li. Learning to translate in real-time with neural machine translation. *CoRR*, abs/1610.00388, 2016.

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

Çaglar Gülçehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio. Pointing the unknown words. *CoRR*, abs/1603.08148, 2016.

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

Kyuyeon Hwang and Wonyong Sung. Character-level incremental speech recognition with recurrent neural networks. In *Proc. ICASSP*, 2016.

Navdeep Jaitly, David Sussillo, Quoc V Le, Oriol Vinyals, Ilya Sutskever, and Samy Bengio. A neural transducer. *NIPS*, 2016.

Frederick Jelinek. *Statistical Methods for Speech Recognition*. MIT, 1998.

Nal Kalchbrenner and Phil Blunsom. Recurrent continuous translation models. In *Proc. EMNLP*, 2013.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proc. ICIR*, 2015.

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

Yishu Miao and Phil Blunsom. Language as a latent variable: Discrete generative models for sentence compression. In *Proc. EMNLP*, 2016.

Courtney Napoles, Matthew Gormley, and Benjamin Van Durme. Annotated gigaword. In *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction*, 2012.

Garrett Nicolai, Colin Cherry, and Grzegorz Kondrak. Inflection generation as discriminative string transduction. In *HLT-NAACL*, 2015.

Alexander M. Rush, Sumit Chopra, and Jason Weston. A neural attention model for abstractive sentence summarization. In *Proc. EMNLP*, 2015.

Baskaran Sankaran, Ajeet Grewal, and Anoop Sarkar. Incremental decoding for phrase-based statistical machine translation. In *Proc. WMT*, 2010.

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

Claude Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27(3): 379–423, 1948.

Khalil Sima'an. Computational complexity of probabilistic disambiguation by means of tree-grammars. In *Proc. COLING*, 1996.

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

Christoph Tillmann, Stephan Vogel, Hermann Ney, and Alex Zubiaga. A DP-based search using monotone alignments in statistical translation. In *Proc. EACL*, 1997.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks. In *Proc. NIPS*, 2015.

Lei Yu, Jan Buys, and Phil Blunsom. Online segment to segment neural transduction. In *Proc. EMNLP*, 2016.

A ALGORITHM

Algorithm 1 Noisy Channel Decoding

```
Notation: $Q$ is the Viterbi matrix, bp is the backpointer, $W$ stores the predicted tokens, $\mathcal{V}$ refers to the vocabulary, $I = |\boldsymbol{x}|$, and $J_{\text{max}}$ denotes the maximum number of output tokens that can be predicted.
Input: source sequence $\boldsymbol{x}$
Output: best output sequence $\boldsymbol{y}^*$
Initialisation: $Q \in \mathbb{R}^{I \times J_{\text{max}} \times K_1}$, bp $\in \mathbb{N}^{I \times J_{\text{max}} \times K_1}$, $W \in \mathbb{N}^{I \times J_{\text{max}} \times K_1}$,
                $Q_{temp} \in \mathbb{R}^{K_1}$, $bp_{temp} \in \mathbb{N}^{K_1}$, $W_{temp} \in \mathbb{N}^{K_1}$
for $i \in [1, I]$ do
    $Q_{temp} \leftarrow \text{topk}(K_1)_{y \in \mathcal{V}} q(z_1 = i)\ q(y \mid \text{START}, z_1, \boldsymbol{x}_1^{z_1})$  ▷ Candidates generated by $q(\boldsymbol{y} \mid \boldsymbol{x})$.
    $bp_{temp} \leftarrow 0$
    $W_{temp} \leftarrow \arg\text{topk}(K_1)_{y \in \mathcal{V}} q(z_1 = i)\ q(y \mid \text{START}, z_1, \boldsymbol{x}_1^{z_1})$
    $Q[i, 1] \leftarrow \text{topk}(K_2)_{y \in W_{temp}} O_{\boldsymbol{x}_1^i, y}$  ▷ Rerank the candidates by objective ($O$).
    $W[i, 1] \leftarrow \arg\text{topk}(K_2)_{y \in W_{temp}} O_{\boldsymbol{x}_1^i, y}$
end for
for $j \in [2, J_{\text{max}}]$ do
    for $i \in [1, I]$ do
        $Q_{temp} \leftarrow \text{topk}(K_1)_{y \in \mathcal{V}, k \in [1,i]} Q[k, j-1] \cdot q(z_j = i \mid z_{j-1} = k) q(y \mid \boldsymbol{y}_1^{j-1}, z_j, \boldsymbol{x}_1^{z_j})$
        $bp_{temp}, W_{temp} \leftarrow \arg\text{topk}(K_1)_{y \in \mathcal{V}, k \in [1,i]}\ Q[k, j-1] q(z_j = i \mid z_{j-1} = k) \cdot$
                $q(y \mid \boldsymbol{y}_1^{j-1}, z_j, \boldsymbol{x})$
        $Y \leftarrow \text{getCandidateOutputs}(bp_{temp}, W_{temp})$  ▷ Get partial candidate $\boldsymbol{y}_1^j$.
        $Q[i, j] \leftarrow \text{topk}(K_2)_{\boldsymbol{y}_j \in Y} O_{\boldsymbol{x}_1^i, \boldsymbol{y}_1^j}$
        $bp[i, j], W[i, j] \leftarrow \arg\text{topk}(K_2)_{\boldsymbol{y}_1^j \in Y}\ O_{\boldsymbol{x}_1^i, \boldsymbol{y}_1^j}$
    end for
end for
return a sequence of words stored in $W$ by following backpointers starting from $(I, \arg\max_j Q[I, j])$.
```

B EXAMPLE OUTPUTS

	Summarisation
Example 1:	
source:	the european commission on health and consumers protection –lrb– _unk_ –rrb– has offered cooperation to indonesia in coping with the spread of avian influenza in the country , official news agency antara said wednesday .
reference:	eu offers indonesia cooperation in avian flu eradication
direct:	eu offers cooperation to indonesia in avian flu
nc:	eu offers cooperation to indonesia in coping with bird flu
Example 2:	
source:	vietnam will accelerate the export of industrial goods mainly by developing auxiliary industries , and helping enterprises sharpen competitive edges , according to the ministry of industry on thursday .
reference:	vietnam to boost industrial goods export
direct:	vietnam to accelerate export of industrial goods
nc:	vietnam to speed up export of industrial goods
	Translation
Example 3:	
source:	欧盟 和 美国 都 表示 可以 接受 这 一 妥协 方案 。
reference:	both the eu and the us indicated that they can accept this plan for a compromise .
direct:	the eu and the united states indicated that it can accept this compromise .
nc:	the european union and the united states have said that they can accept such a compromise plan .
Example 4:	
source:	那么 这些 这个 方面 呢 是 现在 警方 调查 重点 。
reference:	well , this is the current focus of police investigation .
direct:	these are present at the current police .
nc:	then these are the key to the current police investigation .

Table 5: Example outputs on the test set from the direct model and noisy channel model for the summarisation task and machine translation.