Plan, Attend, Generate: Character-Level Neural Machine Translation with Planning

Caglar Gulcehre* University of Montreal
Francis Dutil* University of Montreal
Adam Trischler Microsoft Research
Yoshua Bengio University of Montreal

Abstract

We investigate the integration of a planning mechanism into an encoder-decoder architecture with attention. We develop a model that can plan ahead when it computes alignments between the source and target sequences not only for a single time-step, but for the next k timesteps as well by constructing a matrix of proposed future alignments and a commitment vector that governs whether to follow or recompute the plan. This mechanism is inspired by strategic attentive reader and writer (STRAW) model, a recent neural architecture for planning with hierarchical reinforcement learning that can also learn higher level temporal abstractions. Our proposed model is end-to-end trainable with differentiable operations. We show that our model outperforms strong baselines on character-level translation task from WMT'15 with less parameters and computes alignments that are qualitatively intuitive.

1 Introduction

Character-level neural machine translation (NMT) is an attractive research problem (Lee et al., 2016; Chung et al., 2016; Luong and Manning, 2016) because it addresses important issues encountered in word-level NMT. Word-level NMT systems can suffer from problems with rare words(Gulcehre et al., 2016) or data sparsity, and the existence of compound words without explicit segmentation in certain language pairs can make learning alignments and translations more difficult. Character-level neural machine translation mitigates these issues.

In this work we propose integrating a planning algorithm with the standard encoder-decoder architecture for character-level NMT, using planning specifically to improve the alignment between source and target sequences. We cast alignment (also called *attention*) as a planning problem, whereas it has traditionally been treated as a search problem.

The model we propose creates an explicit plan of source-target alignments to use at future time-steps, based on its current observation and a summary of its past actions; it may modify this plan as needed. The planning mechanism itself is inspired by the *strategic attentive reader and writer* (STRAW) of Vezhnevets et al. (2016).

Our work is motivated by the intuition that, although natural language (speech and writing) is *generated* sequentially because of human physiological constraints, it is almost certainly not *conceived* word-by-word.

Planning, i.e., choosing some goal along with candidate macro-actions to arrive at it, is one way to induce *coherence* in natural language. Learning to generate long coherent sequences or how to form alignments over long source contexts is difficult for existing models. In the case of machine translation, performance of encoder-decoder models with attention deteriorates as sequence length increases (Cho et al., 2014; Sutskever et al., 2014). This effect can be more pronounced in character-level NMT, because the length of sequences in character-level translation can be much longer than word-level translation. A planning mechanism could make the decoder's search for alignments more tractable and scalable.

Our model is based on the well-known encoder-decoder framework for NMT. Its encoder is a recurrent neural network (RNN) that reads the source (a sequence of byte pairs representing text in some language) and encodes it as a sequence of vector representations; the decoder is a second RNN that generates the target translation character-by-character in the target language. The decoder uses an attention mechanism to align its internal state to vectors in the source encoding that are relevant to the current generation step

*Equal Contribution

Proceedings of the 2nd Workshop on Representation Learning for NLP, pages 228–234,
Vancouver, Canada, August 3, 2017.

(see Bahdanau et al. (2015) for the original description). To plan ahead explicitly rather than focusing primarily on what is relevant at the present time, our model's internal state is augmented with (i) an *action plan* matrix and (ii) a *commitment plan* vector. The action plan matrix is a template of alignments that the model intends to follow at future time-steps, specifically a sequence of probability distributions over source tokens. The commitment plan vector governs whether to recompute the action plan or to continue following it, and as such models discrete decisions.

Because of computational constraints we here apply planning only on the input sequence, via searching for alignments. We find this alignment-based planning to be helpful in the translation task. For other NLP tasks, however, planning could be applied explicitly for generation as well. Recent work by Bahdanau et al. (2016) on actor-critic methods for sequence prediction, for example, can be seen as this kind of generative planning.

We evaluate our model and report results on character-level translation tasks from WMT'15 for English to German, English to Finnish, and English to Czech language pairs. On almost all pairs we observe improvements over a baseline that represents the state-of-the-art in neural character-level translation. In our NMT experiments, our model outperforms the baseline despite using significantly fewer parameters and converges faster in training.

2 Planning for Character-level Neural Machine Translation

We now describe how to integrate a planning mechanism into a sequence-to-sequence architecture with attention (Bahdanau et al., 2015). Our model first creates a *plan*, then computes a soft *alignment* based on the plan, and *generates* at each time-step in the decoder. We refer to our model as PAG (Plan-Attend-Generate).

2.1 Notation and Encoder

As input our model receives a sequence of tokens, $X = (x_0, \cdots, x_{|X|})$, where $|X|$ denotes the length of X. It processes these with the encoder, a bidirectional RNN. At each input position i we obtain annotation vector $\mathbf{h}_i$ by concatenating the forward and backward encoder states, $\mathbf{h}_i = [\mathbf{h}_i^{\rightarrow}; \mathbf{h}_i^{\leftarrow}]$, where $\mathbf{h}_i^{\rightarrow}$ denotes the hidden state of the encoder's forward RNN and $\mathbf{h}_i^{\leftarrow}$ denotes the hidden state of the encoder's backward RNN.

Through the decoder the model predicts a sequence of output tokens, $Y = (y_1, \cdots, y_{|Y|})$. We denote by $\mathbf{s}_t$ the hidden state of the decoder RNN generating the target output token at time-step t.

2.2 Alignment and Decoder

Our goal is a mechanism that plans which parts of the input sequence to focus on for the next k time-steps of decoding. For this purpose, our model computes an alignment plan matrix $\mathbf{A}_t \in \mathbb{R}^{k \times |X|}$ and commitment plan vector $\mathbf{c}_t \in \mathbb{R}^k$ at each time-step. Matrix $\mathbf{A}_t$ stores the alignments for the current and the next $k-1$ timesteps; it is conditioned on the current input, i.e. the token predicted at the previous time-step $\mathbf{y}_t$, and the current context ψ_t, which is computed from the input annotations $\mathbf{h}_i$. The recurrent decoder function, $f_{\text{dec-rnn}}(\cdot)$, receives $\mathbf{s}_{t-1}$, $\mathbf{y}_t$, ψ_t as inputs and computes the hidden state vector

$$\mathbf{s}_t = f_{\text{dec-rnn}}(\mathbf{s}_{t-1}, \mathbf{y}_t, \psi_t). \tag{1}$$

Context ψ_t is obtained by a weighted sum of the encoder annotations,

$$\psi_t = \sum_{i}^{|X|} \alpha_{ti} \mathbf{h}_i, \tag{2}$$

where the soft-alignment vector $\alpha_t = \texttt{softmax}(\mathbf{A}_t[0]) \in \mathbb{R}^{|X|}$ is a function of the first row of the alignment matrix. At each time-step, we compute a candidate alignment-plan matrix $\bar{\mathbf{A}}_t$ whose entry at the i^{th} row is

$$\bar{\mathbf{A}}_t[i] = f_{\text{align}}(\mathbf{s}_{t-1}, \mathbf{h}_j, \beta_t^i, \mathbf{y}_t), \tag{3}$$

where $f_{\text{align}}(\cdot)$ is an MLP and β_t^i denotes a summary of the alignment matrix's i^{th} row at time $t-1$. The summary is computed using an MLP, $f_r(\cdot)$, operating row-wise on $\mathbf{A}_{t-1}$: $\beta_t^i = f_r(\mathbf{A}_{t-1}[i])$.

The commitment plan vector $\mathbf{c}_t$ governs whether to follow the existing alignment plan, by shifting it forward from $t-1$, or to recompute it. Thus, $\mathbf{c}_t$ represents a discrete decision. For the model to operate discretely, we use the recently proposed Gumbel-Softmax trick (Jang et al., 2016; Maddison et al., 2016) in conjunction with the straight-through estimator (Bengio et al., 2013) to backpropagate through $\mathbf{c}_t$.[1] The model further learns the temperature for the Gumbel-Softmax as proposed in (Gulcehre et al., 2017). Both the commitment vector and the action plan matrix are initialized with ones; this initialization is not modified through training.

[1] We also experimented with training $\mathbf{c}_t$ using REINFORCE (Williams, 1992) but found that Gumbel-Softmax led to better performance.

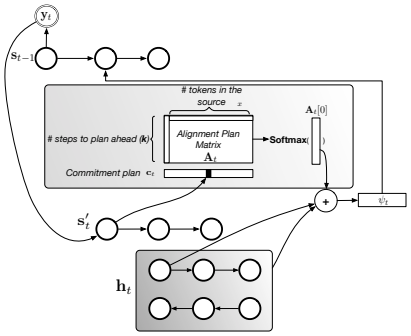

Figure 1: Our planning mechanism in a sequence-to-sequence model that learns to plan and execute alignments. Distinct from a standard sequence-to-sequence model with attention, rather than using a simple MLP to predict alignments our model makes a plan of future alignments using its alignment-plan matrix and decides when to follow the plan by learning a separate commitment vector. We illustrate the model for a decoder with two layers $\mathbf{s}'_t$ for the first layer and the $\mathbf{s}_t$ for the second layer of the decoder. The planning mechanism is conditioned on the first layer of the decoder ($\mathbf{s}'_t$).

Alignment-plan update Our decoder updates its alignment plan as governed by the commitment plan. Denoted by g_t the first element of the discretized commitment plan $\bar{\mathbf{c}}_t$. In more detail, $g_t = \bar{\mathbf{c}}_t[0]$, where the discretized commitment plan is obtained by setting $\mathbf{c}_t$'s largest element to 1 and all other elements to 0. Thus, g_t is a binary indicator variable; we refer to it as the commitment switch. When $g_t = 0$, the decoder simply advances the time index by shifting the action plan matrix $\mathbf{A}_{t-1}$ forward via the shift function $\rho(\cdot)$. When $g_t = 1$, the controller reads the action-plan matrix to produce the summary of the plan, β^i_t. We then compute the updated alignment plan by interpolating the previous alignment plan matrix $\mathbf{A}_{t-1}$ with the candidate alignment plan matrix $\bar{\mathbf{A}}_t$. The mixing ratio is determined by a learned update gate $\mathbf{u}_t \in \mathbb{R}^{k \times |X|}$, whose elements $\mathbf{u}_{ti}$ correspond to tokens in the input sequence and are computed by an MLP with sigmoid activation, $f_{\text{up}}(\cdot)$:

$$\mathbf{u}_{ti} = f_{\text{up}}(\mathbf{h}_i, \mathbf{s}_{t-1}),$$
$$\mathbf{A}_t[:,i] = (1-\mathbf{u}_{ti}) \odot \mathbf{A}_{t-1}[:,i] + \mathbf{u}_{ti} \odot \bar{\mathbf{A}}_t[:,i].$$

To reiterate, the model only updates its alignment plan when the current commitment switch g_t is active. Otherwise it uses the alignments planned and committed at previous time-steps.

Algorithm 1: Pseudocode for updating the alignment plan and commitment vector.

for $j \in \{1, \cdots |X|\}$ **do**
  for $t \in \{1, \cdots |Y|\}$ **do**
    if $g_t = 1$ **then**
      $\mathbf{c}_t = \text{softmax}(f_c(\mathbf{s}_{t-1}))$
      $\beta^j_t = f_r(\mathbf{A}_{t-1}[j])$ {Read alignment plan}
      $\bar{\mathbf{A}}_t[i] = f_{\text{align}}(\mathbf{s}_{t-1}, \mathbf{h}_j, \beta^i_t, \mathbf{y}_t)$
      {Compute candidate alignment plan}
      $\mathbf{u}_{ti} = f_{\text{up}}(\mathbf{h}_i, \mathbf{s}_{t-1}, \psi_{t-1})$ {Compute update gate}
      $\mathbf{A}_t = (1 - \mathbf{u}_{ti}) \odot \mathbf{A}_{t-1} + \mathbf{u}_{ti} \odot \bar{\mathbf{A}}_t$
      {Update alignment plan}
    else
      $\mathbf{A}_t = \rho(\mathbf{A}_{t-1})$ {Shift alignment plan}
      $\mathbf{c}_t = \rho(\mathbf{c}_{t-1})$ {Shift commitment plan}
    end if
    Compute the alignment as $\alpha_t = \texttt{softmax}(\mathbf{A}_t[0])$
  end for
end for

Commitment-plan update The commitment plan also updates when g_t becomes 1. If g_t is 0, the shift function $\rho(\cdot)$ shifts the commitment vector forward and appends a 0-element. If g_t is 1, the model recomputes $\mathbf{c}_t$ using a single layer MLP ($f_c(\cdot)$) followed by a Gumbel-Softmax, and $\bar{\mathbf{c}}_t$ is recomputed by discretizing $\mathbf{c}_t$ as a one-hot vector:

$$\mathbf{c}_t = \texttt{gumbel\_softmax}(f_c(\mathbf{s}_{t-1})), \quad (4)$$
$$\bar{\mathbf{c}}_t = \texttt{one\_hot}(\mathbf{c}_t). \quad (5)$$

We provide pseudocode for the algorithm to compute the commitment plan vector and the action plan matrix in Algorithm 2. An overview of the model is depicted in Figure 1.

2.2.1 Alignment Repeat

In order to reduce the model's computational cost, we also propose an alternative approach to computing the candidate alignment-plan matrix at every step. Specifically, we propose a model variant that reuses the alignment from the previous time-step until the commitment switch activates, at which time the model computes a new alignment. We call this variant *repeat, plan, attend, and generate* (rPAG). rPAG can be viewed as learning an explicit segmentation with an implicit planning mechanism in an unsupervised fashion. Repetition can reduce the computational complexity of the alignment mechanism drastically; it also eliminates the need for an explicit alignment-plan matrix, which reduces the model's memory consumption as well. We provide pseudocode for rPAG in Algorithm 2.

Algorithm 2: Pseudocode for updating the repeat alignment and commitment vector.

```
for j ∈ {1,···|X|} do
  for t ∈ {1,···|Y|} do
    if g_t = 1 then
      c_t = softmax(f_c(s_{t−1}, ψ_{t−1}))
      α_t = softmax(f_align(s_{t−1}, h_j, y_t))
    else
      c_t = ρ(c_{t−1}) {Shift the commitment vector c_{t−1}}
      α_t = α_{t−1} {Reuse the old the alignment}
    end if
  end for
end for
```

2.3 Training

We use a deep output layer (Pascanu et al., 2013) to compute the conditional distribution over output tokens,

$$p(\mathbf{y}_t|\mathbf{y}_{<t},\mathbf{x}) \propto \mathbf{y}_t^\top \exp(\mathbf{W}_o f_o(\mathbf{s}_t,\mathbf{y}_{t-1},\psi_t)), \quad (6)$$

where $\mathbf{W}_o$ is a matrix of learned parameters and we have omitted the bias for brevity. Function f_o is an MLP with $\tanh$ activation.

The full model, including both the encoder and decoder, is jointly trained to minimize the (conditional) negative log-likelihood

$$\mathcal{L} = -\frac{1}{N}\sum_{n=1}^{N} \log p_\theta(\mathbf{y}^{(n)}|\mathbf{x}^{(n)}),$$

where the training corpus is a set of $(\mathbf{x}^{(n)},\mathbf{y}^{(n)})$ pairs and $\boldsymbol{\theta}$ denotes the set of all tunable parameters. As noted in (Vezhnevets et al., 2016), the proposed model can learn to recompute very often which decreases the utility of planning. In order to avoid this behavior, we introduce a loss that penalizes the model for committing too often,

$$\mathcal{L}_{\text{com}} = \lambda_{\text{com}} \sum_{t=1}^{|X|} \sum_{i=0}^{k} ||\frac{1}{k} - \mathbf{c}_{ti}||_2^2, \quad (7)$$

where λ_{com} is the commitment hyperparameter and k is the timescale over which plans operate.

3 Experiments

Character-level neural machine translation (NMT) is an attractive research problem (Lee et al., 2016; Chung et al., 2016; Luong and Manning, 2016) because it addresses important issues encountered in word-level NMT. Word-level NMT systems can suffer from problems with rare words (Gulcehre et al., 2016) or data sparsity, and the existence of compound words without explicit segmentation in some language pairs can make learning alignments between different languages and translations to be more difficult. Character-level neural machine translation mitigates these issues.

In our NMT experiments we use byte pair encoding (BPE) (Sennrich et al., 2015) for the source sequence and characters at the target, the same setup described in Chung et al. (2016). We also use the same preprocessing as in that work.[2] We present our experimental results in Table 2. Models were tested on the WMT'15 tasks for English to German (En→De), English to Czech (En→Cs), and English to Finnish (En→Fi) language pairs. The table shows that our planning mechanism improves translation performance over our baseline (which reproduces the results reported in (Chung et al., 2016) to within a small margin). It does this with fewer updates and fewer parameters. We trained (r)PAG for 350K updates on the training set, while the baseline was trained for 680K updates. We used 600 units in (r)PAG's encoder and decoder, while the baseline used 512 in the encoder and 1024 units in the decoder. In total our model has about 4M fewer parameters than the baseline. We tested all models with a beam size of 15.

As can be seen from Table 2, layer normalization (Ba et al., 2016) improves the performance of PAG model significantly. However, according to our results on En→De, layer norm affects the performance of our rPAG only marginally. Thus, we decided not to train rPAG with layer norm on other language pairs.

In Table 1, we present the results for PAG using the biscale decoder.

Table 1: WMT'15 En→De Results

	Beam Size	Development	Test Set
Baseline	8	20.39	20.11
Baseline	24	20.52	20.39
PAG	8	21.19	20.84
PAG	24	**21.26**	**20.98**

In Figure 2, we show qualitatively that our model constructs smoother alignments. At each word that the baseline decoder generates, it aligns the first few characters to a word in the source sequence, but for the remaining characters places the largest alignment weight on the last, empty token of the source sequence. This is because the baseline becomes confident of which word to generate after the first few

[2]Our implementation is based on the code available at `https://github.com/nyu-dl/dl4mt-cdec`

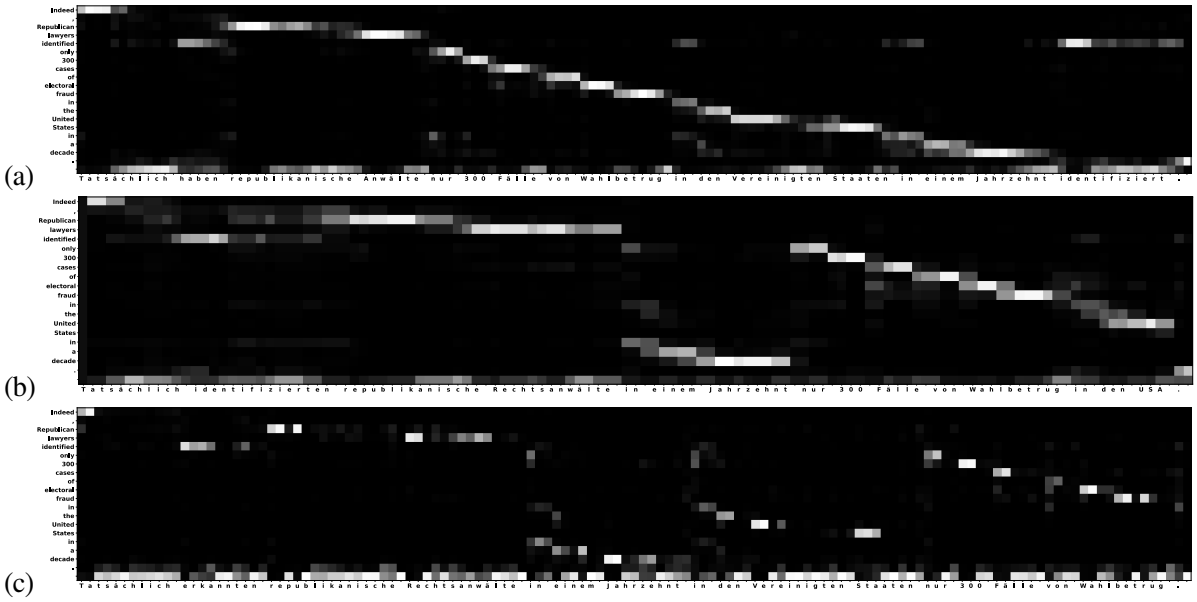

Figure 2: We visualize the alignments learned by PAG in (a) and the biscale baseline model in (b). As depicted, the alignments learned by PAG look more accurate intuitively and appear smoother than those of the baseline. The baseline tends to focus too much attention on the last word of the sequence, which is sensible to do on average because of German's structure, whereas our model places higher weight on the last word mainly when it generates a space token.

	Model	Layer Norm	Dev	Test 2014	Test 2015
En→De	Baseline	✗	21.57	21.33	**23.45**
	Baseline†	✗	21.4	21.16	22.1
	PAG	✗	21.52	21.35	22.21
		✓	**22.12**	**21.93**	22.83
	rPAG	✗	21.81	21.71	22.45
		✓	21.67	21.81	22.73
En→Cs	Baseline	✗	17.68	19.27	16.98
	PAG	✗	17.44	18.72	16.99
		✓	**18.78**	**20.9**	**18.59**
	rPAG	✗	17.83	19.54	17.79
En→Fi	Baseline	✗	11.19	-	10.93
	PAG	✗	11.51	-	11.13
		✓	**12.67**	-	**11.84**
	rPAG	✗	11.50	-	10.59

Table 2: The results of different models on WMT'15 task on English to German, English to Czech and English to Finnish language pairs. We report BLEU scores of each model computed via the *multi-blue.perl* script. The best-score of each model for each language pair appears in bold-face. We use *newstest2013* as our development set, *newstest2014* as our "Test 2014" and *newstest2015* as our "Test 2015" set. $(^{\dagger})$ denotes the results of the baseline that we trained using the hyperparameters reported in (Chung et al., 2016) and the code provided with that paper. For our baseline, we only report the median result, and do not have multiple runs of our models.

characters, and it generates the remainder of the word mainly by relying on language-model predictions. We observe that (r)PAG converges faster with the help of the improved alignments, as illustrated by the learning curves in Figure 3.

4 Conclusions and Future Work

In this work we addressed a fundamental issue in neural generation of long sequences by integrating *planning* into the alignment mechanism of sequence-to-sequence architectures. We proposed two different planning mechanisms: PAG, which constructs explicit plans in the form of stored matrices, and rPAG, which plans implicitly and is computationally cheaper. The

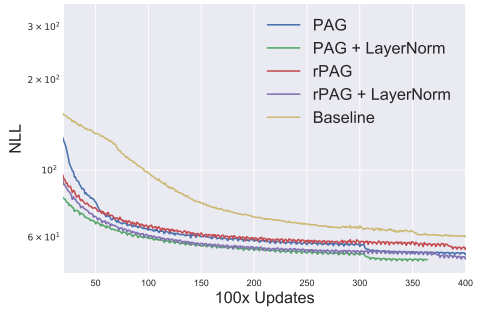

Figure 3: Learning curves for different models on WMT'15 for En→De. Models with the planning mechanism converge faster than our baseline (which has larger capacity).

(r)PAG approach empirically improves alignments over long input sequences. We demonstrated our models' capabilities through results on character-level machine translation, an algorithmic task, and question generation. In machine translation, models with planning outperform a state-of-the-art baseline on almost all language pairs using fewer parameters. As a future work, we plan to test our planning mechanism at the outputs of the model and other sequence to sequence tasks as well.

References

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450* .

Dzmitry Bahdanau, Philemon Brakel, Kelvin Xu, Anirudh Goyal, Ryan Lowe, Joelle Pineau, Aaron Courville, and Yoshua Bengio. 2016. An actor-critic algorithm for sequence prediction. *arXiv preprint arXiv:1607.07086* .

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. *International Conference on Learning Representations (ICLR)* .

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. 2013. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432* .

Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259* .

Junyoung Chung, Kyunghyun Cho, and Yoshua Bengio. 2016. A character-level decoder without explicit segmentation for neural machine translation. *arXiv preprint arXiv:1603.06147* .

Caglar Gulcehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio. 2016. Pointing the unknown words. *arXiv preprint arXiv:1603.08148* .

Caglar Gulcehre, Sarath Chandar, and Yoshua Bengio. 2017. Memory augmented neural networks with wormhole connections. *arXiv preprint arXiv:1701.08718* .

Eric Jang, Shixiang Gu, and Ben Poole. 2016. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144* .

Jason Lee, Kyunghyun Cho, and Thomas Hofmann. 2016. Fully character-level neural machine translation without explicit segmentation. *arXiv preprint arXiv:1610.03017* .

Minh-Thang Luong and Christopher D Manning. 2016. Achieving open vocabulary neural machine translation with hybrid word-character models. *arXiv preprint arXiv:1604.00788* .

Chris J Maddison, Andriy Mnih, and Yee Whye Teh. 2016. The concrete distribution: A continuous relaxation of discrete random variables. *arXiv preprint arXiv:1611.00712* .

Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. 2013. How to construct deep recurrent neural networks. *arXiv preprint arXiv:1312.6026* .

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909* .

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*. pages 3104–3112.

Alexander Vezhnevets, Volodymyr Mnih, John Agapiou, Simon Osindero, Alex Graves, Oriol Vinyals, and Koray Kavukcuoglu. 2016. Strategic attentive writer for learning macro-actions. In *Advances in Neural Information Processing Systems*. pages 3486–3494.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning* 8(3-4):229–256.

A Qualitative Translations from both Models

In Table 3, we present example translations from our model and the baseline along with the ground-truth. [3]

Table 3: Randomly chosen example translations from the development-set.

	Groundtruth	Our Model (PAG + Biscale)	Baseline (Biscale)
1	Eine republikanische Strategie , um der Wiederwahl von Obama entgegenzutreten	Eine republikanische Strategie gegen die Wiederwahl von Obama	Eine republikanische Strategie zur Bekämpfung der Wahlen von Obama
2	Die Führungskräfte der Republikaner rechtfertigen ihre Politik mit der Notwendigkeit , den Wahlbetrug zu bekämpfen .	Republikanische Führungspersönlichkeiten haben ihre Politik durch die Notwendigkeit gerechtfertigt , Wahlbetrug zu bekämpfen .	Die politischen Führer der Republikaner haben ihre Politik durch die Notwendigkeit der Bekämpfung des Wahlbetrugs gerechtfertigt .
3	Der Generalanwalt der USA hat eingegriffen , um die umstrittensten Gesetze auszusetzen .	Die Generalstaatsanwälte der Vereinigten Staaten intervenieren , um die umstrittensten Gesetze auszusetzen .	Der Generalstaatsanwalt der Vereinigten Staaten hat dazu gebracht , die umstrittensten Gesetze auszusetzen .
4	Sie konnten die Schäden teilweise begrenzen	Sie konnten die Schaden teilweise begrenzen	Sie konnten den Schaden teilweise begrenzen
5	Darüber hinaus haben Sie das Recht von Einzelpersonen und Gruppen beschränkt , jenen Wählern Hilfestellung zu leisten , die sich registrieren möchten .	Darüber hinaus begrenzten sie das Recht des Einzelnen und der Gruppen , den Wählern Unterstützung zu leisten , die sich registrieren möchten .	Darüber hinaus unterstreicht Herr Beaulieu die Bedeutung der Diskussion Ihrer Bedenken und Ihrer Familiengeschichte mit Ihrem Arzt .

[3] These examples are randomly chosen from the first 100 examples of the development set. None of the authors of this paper can speak or understand German.