# RNN Machine Translation Faster Decoder

**Weiqiu You**
wyou@cs.umass.edu

## 1 Problem statement

Machine translation systems have brought convenience to people's lives as they provide us a tool to access more texts without knowing the languages. As it is expensive to have a human translator all the time, machine translation becomes an useful tool when the quality is enough for the situation. At first, rule-based machine translation was explored. Statistical machine translation followed and became a success. Nowadays, neural machine translation (NMT) has become the state-of-the-art approach. An RNN-NMT usually contains an encoder and a decoder part. The encoder encodes each word in the source language sequentially using recurrent neural networks (RNN), and the decoder takes last hidden layer of the encoder and produces words in the target language one at a time also using RNNs. Common different types of RNNs include LSTM and GRU. Recently, the Transformer model came out and allows encoding in parallel.

As translation qualities increase, we hope to investigate more in speeding up the translation process to give better user experiences. In this project, we focus on speeding up the decoder part for RNN models. Instead of decoding one word at a time in an usual autoregressive style, we are going to decode multiple words in a semi-autoregressive pattern. However, this will usually lower the quality of translation since the words outputted at the same time have no knowledge of each other. Therefore, we originally hope to incorporate syntax in training the decoder, hoping to speed up the translation while not compromising output qualities too much (Figure 1). As the project progresses, we have encountered a lot of difficulties in implementations, mainly in speeding up training and debugging errors. Due to limited time, we switch the focus of our project to just implementing the $k$-span decoder which outputs $k$ words at a time and do an analysis on the time and BLEU scores between models of different span sizes (Figure 2). Incorporating syntax and potentially improving translation quality will become a future work.

## 2 What we proposed vs. what we accomplished

- ~~Tokenize the dataset with standard tokenizer~~
- ~~Convert dataset into byte-pair-encoding~~
- ~~Build a sampler that samples by number of tokens~~
- ~~Build and train RNN translation model with span of 1 on IWSLT dataset and examine its performance~~
- ~~Build and train k-span decoder model and examine its performance~~
- ~~Implement beam search~~
- *Build syntax decoder model and examine its performance*: We failed to do this because of lack of time
- ~~Perform in-depth error analysis to figure out what kinds of examples our approach struggles with~~
- *Get the best BLEU score we can get with our model*: We failed to do this because of lack of time to train the model until it does not improve

## 3 Related work

For data preprocessing, nowadays researcher usually use byte-pair-encoding (BPE) to represent sentences (Sennrich et al., 2016), which divides

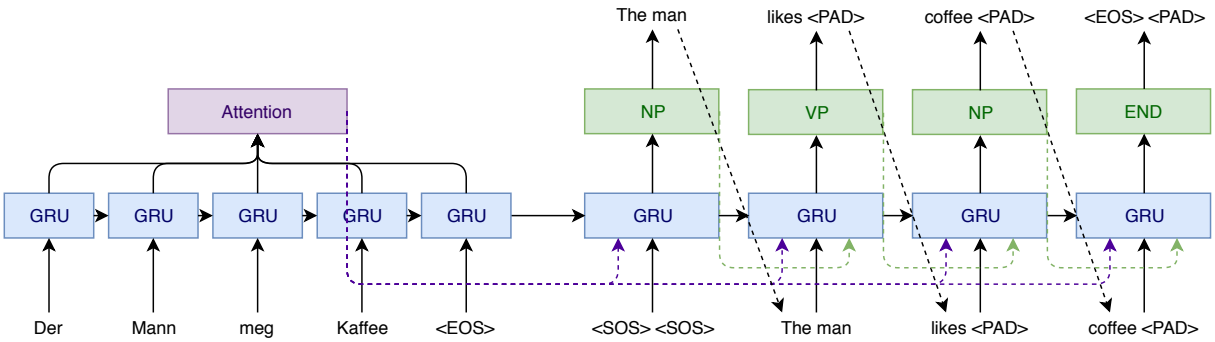


Figure 1: Initial Thought - Syntax Decoder Model

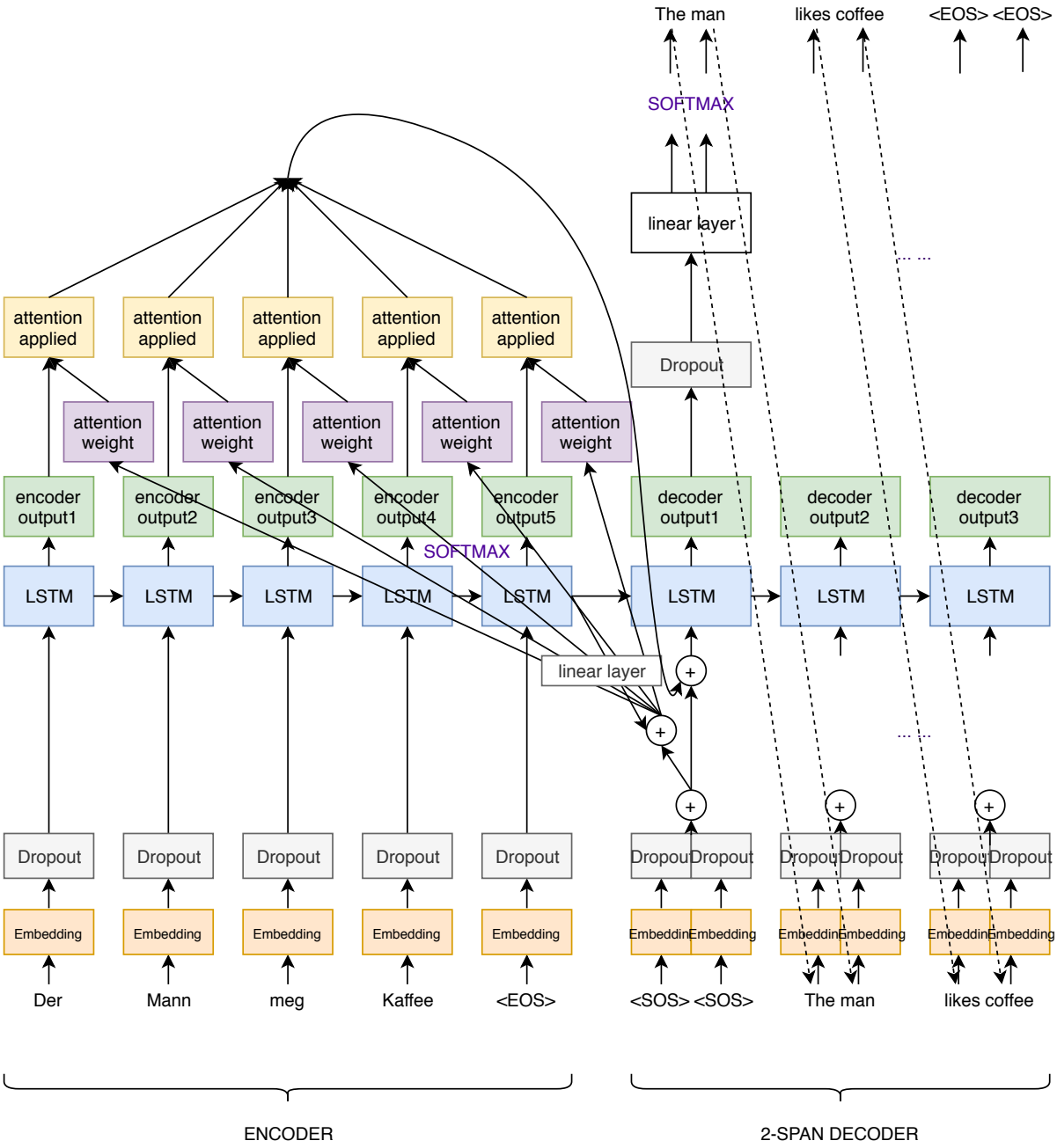


Figure 2: Finished Implementation of K-span Decoder Model

each word into subwords and train the model to learn representations of subwords instead of words. For German-English translation, it is a common practice to use a shared vocabulary. BPE significantly alleviates the problem of unknown words.

Most RNN machine translation models all deploy attention mechanisms that allow the decoders to focus on certain part of the source sentence when decoding. Dzmitry Bahdanau (2015); Minh-Thang Luong (2015) are two commonly used attention mechanisms. The main difference between Bahdanau attention and Luong attention is that Dzmitry Bahdanau (2015) puts attention before going through GRU or LSTM, while Minh-Thang Luong (2015) puts attention after going through GRU or LSTM.

Recently, researchers have been investigating ways to speed up machine translation. A natural way of thinking is to decode multiple words at a time. Xinyi Wang (2018) uses a semi-autoregressive model to decode a fixed span of words each time using Transformer. At training time, the mask is changed to reveal only words for previous spans, but still predict the target one word, so its training time is the same as autoregressive models. At test time, the model can predict all words in the next span each time, resulting in a faster decoding with some degradation in performance. Another similar work Lukasz Kaiser (2018) first encodes the target sentences into shorter sequences of latent variables, and then decodes them back after prediction. Jiatao Gu (2018) uses a non-autoregressive model to decode all words at the same time, also using a modified Transformer architecture. Instead of feeding in the last predicted word, they feed source words into the decoder. They also introduce a concept of *fertility*, which copies each source word a varied number of times when feeding into the decoder, to tackle the multimodal problem.

For our orignal goal of combining syntax into our model, there has been a lot of previous work in incorporating syntax in machine translation, not necessarily speeding up the process. Aharoni and Goldberg (2017) uses a flattened parse tree as the target sentence to improve translation quality. Xinyi Wang (2018) utilizes two RNNs, an RuleRNN that generates context-free-grammar rules by previous rules, and a WordRNN that generates words from each linguistic component generated by RuleRNN. Aside from the machine translation task, Kai Sheng Tai (2015); Xingxing Zhang (2016); Iyyer and John Wieting (2018) also utilize parses or tree structures in other natural language processing tasks such as sentence and sentiment classification, sentence completion, paraphrasing, etc.

Works that do not use parse tree labels but still decode by chunks include Shonosuke Ishiwatari (2017); Hao Zhou (2017). These two works both first train the translator to output chunks, and then output words base on chunks. The difference between the two is that Shonosuke Ishiwatari (2017) uses end-of-chunk labels to determine the end of a chunk, while Hao Zhou (2017) adds a boundary-gate to the LSTM to determine the end of a chunk.

Right now, Professor Mohit Iyyer and his PhD student Nader Akoury are working on speeding the decoder for the Transformer model. Our work, on the other hand, is to examine how speed up can work out in an RNN model.

## 4 Our dataset

We are using IWSLT16 dataset (Table 1). There are 196884 sentence pairs in the training set, and 189015 sentences are below 50 words. There are 7883 sentence pairs in validation set, and 7592 sentences are below 50 words. We are using a shared source-target vocabulary file that contains 167433 tokens. This dataset still needs to be tokenized and preprocessed into bpe.

| Split | # Pairs | # Pairs $\leq$ 50 tokens |
|---|---|---|
| Train | 196884 | 189015 |
| Validation | 7883 | 7592 |

Table 1: Data Statistics

When looking into the IWSLT dataset, we notice that there are some very long sentences that contain more than 700 words. These "sentences" are in fact multiple sentences in a whole song lyrics. It is hard to make the model learn such long dependencies, so we cut off the part longer than 60 tokens for each sentence.

We originally also want to test our model on WMT14 dataset, but are not able to finish due to limited time. WMT14 dataset is a much larget and noisier dataset, which would have been harder to train on.

### 4.1 Data preprocessing

We use SacreMoses (SacreMoses, 2018) to first tokenize German and English texts in training set and development set. Then we use (FastBPE, 2018), an implementation of (Sennrich et al., 2016), to produce byte-pair-encoding representations of inputs and outputs. We use a shared vocabulary of about 37000 tokens, similar to (Ashish Vaswani, 2017).

## 5 Baselines

Our baseline and modified model are using the same architecture (Figure 2), but with different span sizes. When the span size is set to 1, the model becomes a basic RNN with attention model that translates in an autoregressive manner. When the span size is set to greater than 1, it becomes a semi-autoregressive model. In fact, our baseline is expected to behave better than the proposed model, because the proposed model does not condition on all previously predicted words for each new span to be generated. The purpose of this project is to compare the trade-off between speed up and performance.

For encoder, we use two-layer bidirectional LSTM, and concatenate the two directions when feeding the last hidden layer into the decoder as the first hidden layer. We use a fixed length Bahdanau attention that needs the inputs to be cropped or padded to the same length of 60 tokens. We concatenate the embedding layers of each span during decoding, which is just 1 word here. Then we concatenate the concatenated embedding with the last previous hidden layer to project onto a dimension of 60 to get the attention weights for encoder outputs after going through softmax. Then the attention weights are multiplied to encoder outputs to create an attention vector of hidden size length for each sentence. This attention is then concatenated with the embedding, applied ReLU non-linearity, and fed into a 2-layer LSTM. The output of the LSTM then goes through a linear layer to get back to span-size number (still just 1 here for the baseline) of words. The output goes through another dropout and then log softmax is applied to each word, which is again just the single word here. Negative log likelihood loss is used as the training criterion.

We use 2 layers LSTMs with a hidden size of 256, and the encoder is bidirectional. Our embedding is also 256 dimensional. Parameters of LSTMs are uniformly initialized in [-0.1, 0.1]. We use a dropout rate of 0.2 as suggested in (Minh-Thang Luong, 2015). There are dropouts on the embedding, LSTMs, and the outputs. For the current best model, we train for 71 epochs using Adam optimizer with a learning rate of 0.001 for the first 20 epochs and 0.0003 for rest of the epochs. Our mini-batch size is 1000 tokens. The gradients are clipped to 1. We use both greedy decoding and beam search decoding, with beam search decoding having slightly better performance.

We experiment with both Bahdanau attention and Luong attention, and Bahdanau attention gives better results. We do a grid search, as shown in Table 2, to settle on these hyperparameters. We have tried hidden sizes of 256, 512 and 1024, learning rate of 0.01, 1e-3, 3e-4 and 1e-4. We first experiment with different learning rates and 3e-4 works the best. Larger embedding and hidden sizes do not improve the performance. Experiments show that 1024 dimensional LSTMs do not work well as it overfits quickly as Table 2 shows, so we resort to lower dimensional LSTMs, and settle on 256 dimensions. We first start with 1 layer LSTM and only dropout on the embeddings. When the models stop improving, we add a second dropout on outputs of LSTMs, and this allows our model to finally achieve BLEU score more than 25. The model is still learning, but due to limited training time, we are not able to give the best possible result.

## 6 Our approach

Our approach is mostly the same as described in the baseline, except that now it is actually utilizing the concatenation of a span of embeddings. And at the output, it transforms each 512 dimensional output into $512k$ dimensional output in a $k$-span model.

We manage to complete a working implementation mostly from scratch. We use PyTorch to implement the whole model, with the baseline model referencing the attention used in the PyTorch translation tutorial (Robertson, 2018). We implement the part to concatenate embeddings of $k$ words together and retransforming the output back to $k$ words. We also add different dropout locations and implement batch training. We train the model on one 1080ti GPU on gypsum. There has been memory issues before but is now mostly

| Hidden Size | LSTM Layers | Dropout Layers | Dropout Rate | Max Length | Learning Rate | Epochs | BLEU | Test Time(s) |
|---|---|---|---|---|---|---|---|---|
| 1024 | 2 | 1 | 0.1 | 60 | 3e-4 | 16 | 22.6 | 30.37 |
| 1024 | 2 | 1 | 0.3 | 60 | 3e-4 | 16 | 21.6 | 30.22 |
| 1024 | 2 | 1 | 0.1 | 60 | 3e-4 | 19 | 22.5 | 30.09 |
| 1024 | 2 | 1 | 0.2 | 60 | 3e-4 | 19 | 22.5 | 29.79 |
| 1024 | 2 | 1 | 0.3 | 60 | 3e-4 | 19 | 21.6 | 29.57 |
| 1024 | 2 | 1 | 0.1 | 60 | 3e-4 | 26 | 22.1 | 28.83 |
| 1024 | 2 | 1 | 0.3 | 60 | 3e-4 | 26 | 21.8 | 29.07 |
| 512 | 1 | 1 | 0.2 | 50 | 3e-4 | 31 | 23.1 | 13.86 |
| 512 | 1 | 1 | 0.2 | 50 | 3e-4 | 35 | 22.9 | 12.86 |
| 512 | 1 | 1 | 0.2 | 50 | 3e-4 | 40 | 23.0 | 13.44 |
| 512 | 1 | 1 | 0.2 | 50 | 3e-4 | 49 | 22.7 | 13.34 |
| 512 | 1 | 1 | 0.2 | 60 | 3e-4 | 49 | 23.3 | 22.50 |
| 256 | 1 | 1 | 0.2 | 60 | 3e-4 | 31 | 22.2 | 14.93 |
| 256 | 1 | 1 | 0.2 | 60 | 3e-4 | 25 | 21.0 | 13.48 |
| 256 | 1 | 1 | 0.2 | 60 | 3e-4 | 49 | 23.6 | 13.64 |
| 256 | 1 | 1 | 0.2 | 60 | 3e-4 | 39 | 23.2 | 17.52 |
| 256 | 1 | 1 | 0.2 | 60 | 3e-4 | 65 | 24.2 | 19.04 |
| 256 | 2 | 1 | 0.2 | 60 | 3e-4 | 15 | 18.2 | 19.78 |
| 256 | 2 | 2 | 0.2 | 60 | $\leq$ 20 epochs: 1e-3, $>$: 3e-4 | 30 | 23.8 | 19.70 |
| 256 | 2 | 2 | 0.2 | 60 | $\leq$ 20 epochs: 1e-3, $>$: 3e-4 | 71 | 25.8 | 25.57 |

Table 2: Grid Search for 1-Span Baseline Model Using Greedy Search

| | |
|---|---|
| Span | 1 |
| Hidden Size | 256 |
| LSTM Layers | 2 |
| Dropout Layers | 2 |
| Dropout Rate | 0.2 |
| Max Length | 60 |
| Learning Rate | epoch 20: 1e-3, $>$: 3e-4 |
| Epochs | 71 |
| Greedy Search Minibatch Size | 128 |
| Greedy Search BLEU | 25.8 |
| Greedy Search Test Time (s) | 25.57 |
| Beam Search Width | 4 |
| Beam Search BLEU | 26.4 |
| Beam Search Test Time (s) | 1449.86 |

Table 3: Best Model

| Span | BLEU | BLEU ratio | Train time | Train speed up | Test time | Test speed up |
|---|---|---|---|---|---|---|
| 1 | 23.6 | | 25.5h | | 13.6s | |
| 2 | 16.9 | 0.72$\times$ | 21.8h | 1.17$\times$ | 12.29s | 1.11$\times$ |
| 4 | 8.5 | 0.36$\times$ | 14.1h | 1.81$\times$ | 11.28s | 1.16$\times$ |

Table 4: Different Span Size Comparison

resolved by using one instead of multiple GPUs and using a smaller batch size.

We train and test on span sizes of 2, 4, besides the baseline size 1. There has been unfixed memory issue for span size of 6 so we do not train on span size greater than or equal to 6. The comparision is shown in Table 4.

Here, the results are obtained using greedy search, with minibatch of 128 sentences. The BLEU scores degrade significantly for 2-span and 4-span models. 2-span model obtains only $0.72\times$ BLEU socre of 1-span model, while 4-span model obtains only $0.36\times$ BLEU score of 1-span model, which is half of 2-span model. The training time speed up by $1.17\times$ for 2-span model, and $1.81\times$ for 4-span model. This might be due to a lot of overheads such as softmax that cannot be computed jointly for the whole span of words but have to be computed separately for each word in a span. Test time does not have a large speed up since it is already fast due to parallelism. There is a speed up of $1.11\times$ for 2-span model and $1.16\times$ for 4-span model.

The performance degradation is too much, while the speed up is not as much. The speed up does not worth the performance trade-off. Therefore, simply producing a constant of $k$ words at a time is not practical.

Also, although we are using the same model for both baseline and $k$-span approach, what we are supposed to do for baseline is to implement a model that does not have the embedding concatenation layer and the output projection layer. Without those two layers, the model might be able to train much faster and converge faster. This is the true baseline we should be comparing to instead. Our experiments do show speedup for $k$-span models, but it is not compared to the optimal 1-span model. Therefore, it is a bit unfair to claim such speedup.

## 7 Error analysis

Due to lack of time to train the best model on different span sizes, we will analyze the error on outputs of an earlier inferior model that has 1 layer LSTM and only dropout on the embedding but not the output, and a constant learning rate of 3e-4, with everything else the same as described in section 5

Table 5 shows an analysis of outputs of models trained to output words with span sizes of 1, 2, and 4 trained with the same amount of time. Since the losses are still going down at the end of training, these are not the best models, but are here to show the different types of errors that are likely to occur for different span sizes. The table also includes the outputs of the best model, whose hyperparameters are presented in Table 3.

### 7.1 Non-literal Translation in Reference

We can see that models with span of 1 often translate reasonably well. The errors for span size of 1 often occur in translating things that are in the source sentences but not in the reference sentences because references not literal translations. If we look at sentence 1, 2, 5, 7, 9, and 10, we can see that this is a very common issue. For example, in sentence 1, 1-span model translates “hatte” to “had”, which is the correct literal translation but does not match “saw” in the reference. Both should be acceptable. In sentence 2, “und” in the source sentence kind of separates the sentence into two parts. 1-span model does give a translation that keeps this structure, with “, and” separating the two parts, but this does not match the reference which does not keep the structure. In sentence 10, the reference uses a phrase or idiom “kicked the can down the road”, which means “to postpone making a decision or a difficult issue”. This idiom is not used in German so the German source sentence uses a literal way of expression, and thus translated to “were looking at the problems before” in 1-span model and similar texts in other models. Of course, these models’ translations are not perfect either, but their correctness should not be evaluated on whether or not they are able to translate a phrase to a specific English idiom.

### 7.2 Wrong Numbers

The models often get numbers wrong, but not always. In sentence 1, it gets “20s” correctly, but in sentence 3, it gets “26-year-old” wrongly as “2-year-old”. When we inspect the tokenized byte-pair-encoding representations of the source sentence, we find that “20s” is a single token “20ern”, while “26-year-old” is separated into three tokens “2@@”, “6-@@”, and “jhrige”. Therefore, it is easy to explain why the error occurs. The model translates “2@@” and “jährige” but misses the “6@@” which is in the middle. For the best model, it is starting to learn to join another number in between, but fails by producing “22” instead of “26”.

| | | Sentences | Comments |
|---|---|---|---|
| 1 | Source | Als ich in meinen 20ern war, hatte ich meine erste Psychotherapie-Patientin. | |
| | Reference | When I was in my 20s, I saw my very first psychotherapy client. | |
| | Best | When I was in my 20s, I had my first psychotherapy team. | |
| | 1-Span | When I was in my 20s, I had my first psychotherapy. | "Saw" in reference is not literal translation. "had" is translated from "hatte" literally and is acceptable. |
| | 2-Span | When I was in my, I I had my first first patient response transplant. | Has duplicates due to translation by span. |
| | 4-Span | When I was in my, I I was my first transplant patient. | Has duplicates due to translation by span. |
| 2 | Source | Ich war Doktorandin und studierte Klinische Psychologie in Berkeley. | |
| | Reference | I was a Ph.D. student in clinical psychology at Berkeley. | |
| | Best | I was a Ph.D. student, and I studied anthropology in Wolfram. | Even the best model still hasn't learned all proper names. |
| | 1-Span | I was a graduate student in Kenya, and I was a psychologist in India. | Translates more literal than reference, connecting two subsentences with "and" like in the source sentence with "und"; proper name translated wrong. |
| | 2-Span | I was a student in graduate student researcher in Michigan. | Captures some information about studying, but misses the details. |
| | 4-Span | I was a graduate student, college Church in in in. | Second part of the sentence is really off. |
| 3 | Source | Sie war eine 26-jährige Frau namens Alex. | |
| | Reference | She was a 26-year-old woman named Alex. | |
| | Best | She was a 22-year-old woman named Alex. | The best model is starting to learn how to join numbers together. |
| | 1-Span | She was a 2-year-old woman named Alex. | The model doesn't learn the numbers correctly. Might be too sparse. |
| | 2-Span | She was a 2-year-old -year-old son named Albert. | Number, gender and name are wrong. Duplicates. |
| | 4-Span | She was a a young year-old young daughter. | Loses more structures. Duplicates. |
| 4 | Source | Als Alex in die erste Sitzung kam, trug sie Jeans und ein ausgebeultes Top. Sie fiel auf das Sofa in meinem Bro, schleuderte ihre Sandalen von sich und erzhlte mir, sie wre da, um ber Mnnerprobleme zu reden. | |
| | Reference | Now Alex walked into her first session wearing jeans and a big slouchy top, and she dropped onto the couch in my office and kicked off her flats and told me she was there to talk about guy problems. | |
| | Best | When the surgeon came to the first meeting, she had jeans and a well-intentioned list, and she went to the couch in my office, and she filled her sandwiches and told me she was going to talk about issues about issues. | Somehow wrongly misses the name "Alex" and replaces with "surgeon". |
| | 1-Span | As Alex said, the first time she sang, she got jeans and a permanent one, and she fell on the couch in my office, her lower jaw, and told me, she was there to talk about issues about issues. | Gets most of the semantic information. Some duplicates at the end. Not fully grammatical. |
| | 2-Span | When when the came first to the first night, she was a bowl and a stand-up of the, and it was on the home in my office, cleaning on their sandvis, and told me, she'd be to talk about the. | Adjacent + non-adjacent duplicates. Have the basic semantic structure. |
| | 4-Span | When the was in the first,, she had, and she smodrunk a dirt jar, and she in in at the baseball,, sitting my kitchen desk, they her her grandfather, telling me, they she to be about about | More adjacent duplicates. Not grammatical. |
| 5 | Source | Und als ich das hörte, war ich erleichtert. | |
| | Reference | Now when I heard this, I was so relieved. | |
| | Best | And when I heard that, I was relieved. | Misses "so", but "so" is not in the source. |
| | 1-Span | And when I heard that, I was relieved. | Misses "so", but "so" is not in the source. |
| | 2-Span | And when I heard this, I was a. | Gets the important adjective wrong. |
| | 4-Span | And when I heard it, was was appropriate. | Duplicate. Wrong adjective. |
| 6 | Source | Meine Kommilitonin bekam nämlich einen Brandstifter als ersten Patienten. | |
| | Reference | My classmate got an arsonist for her first client. | |
| | Best | My credit engine has been a bribe in the first patient. | ""Arsonist" and "bribe" both have negative connotations. |
| | 1-Span | My comfort was in a shower, first cloned as a patient. | A lot of semantic information is lost and distorted. But "Patienten" can actually be translated as "patient". |
| | 2-Span | My my favorite competitor was actually to have a Phof the patient's first.. | Duplicates. |
| | 4-Span | My my was in Kiegang was has successfully first strategy. | Duplicates. More than one verb. |
| 7 | Source | Und ich bekam eine Frau in den 20ern, die über Jungs reden wollte. | |
| | Reference | And I got a twentysomething who wanted to talk about boys. | |
| | Best | And I got a woman in the islands who talked about boys. | Somehow misses the number completely and generates "islands" which is not related to the input. |
| | 1-Span | And I got a woman in the '60s, who was talking about boys. | Number wrong. Translates "Frau" which means "woman", which is not in the reference. |
| | 2-Span | And I got a woman woman who the thought who talked wanted to talk about | |
| | 4-Span | And I was a woman to the the person who to to tell about to. | |
| 8 | Source | Das kriege ich hin, dachte ich mir. | |
| | Reference | This I thought I could handle. | |
| | Best | So I'm going to go, I thought. | More similar to source's structure than reference. |
| | 1-Span | I'll leave it, I thought. | More similar to source's structure than reference. And the first half can be similar to this. |
| | 2-Span | I was going the day, I thought. | Loses some semantic information. |
| | 4-Span | I was me to me, I I thought. | Loses a lot of semantic information. |
| 9 | Source | Aber ich habe es nicht hingekriegt. | |
| | Reference | But I didn't handle it. | |
| | Best | But I didn't get it. | Matches source according to Google translation. |
| | 1-Span | But I didn't get it. | |
| | 2-Span | But I didn't get it. | |
| | 4-Span | But I didn't put it. | Surprisingly good for span of 4 too. |
| 10 | Source | Mit den lustigen Geschichten, die Alex mit in die Sitzung brachte, war es leicht fr mich, einfach mit dem Kopf zu nicken, whrend wir die Probleme vor uns herschoben. | |
| | Reference | With the funny stories that Alex would bring to session, it was easy for me just to nod my head while we kicked the can down the road. | "kick the can down the road" is a non-literal phrase, so it's normal that it does not get translated literally. |
| | Best | With the funny stories that Alex came to the lecture, it was easy for me to just bend the head while we put the problems out there. | Very close to the reference for a sentence this long. |
| | 1-Span | With the funny stories, Alex the Alex was coming in, it was easy for me just to head, as we were looking at the problems before. | Duplicate. Small semantic errors. |
| | 2-Span | With the fun stories that Alex was with the the project, it was easy easy to me with just feel your head, when we were the problems in front. | Duplicates. Some semantic errors. |
| | 4-Span | With the artifacts stories that originally with with the,, the way was was just easy me me me to the head goal, we we we were collecting the our problems. | Duplicating the same words three times. More semantic errors. |

Table 5: Comparisons between Best Outputs and Different Outputs of Different Span Sizes Trained with 1 layer LSTM for 50 Epochs

However, in sentence 7, "20ern" occurs again without being separated into subwords, but it is not translated out by any of the models. It is not that simple to explain for every single example, but it is possible that numbers are too sparse that it is hard for the models to learn how to translate every single number. It would be good to add copy mechanisms that add more probabilities to words that have occurred in the source sentence.

### 7.3 Duplication

Duplication happens a lot for spans more than 1. This is expected, since the words within each span are not conditioned on each other. Some examples include "I I" and "first first" for sentence 1 in 2-span and 4-span models. We can see that they are always from the same span to be produced. The first "I" is the first word in the fourth span and the second "I" is the second word in the fourth span for 2-span model.

Sometimes, even 1-span model has duplication problems. As we can see from sentence 10, the 1-span model (not the best model) outputs "Alex" twice. This might be due to insufficient training, as the problem disappears in the best model.

### 7.4 Vulnerability

We can see from the comparison table that even the best model is very vulnerable. It usually gets the correct meaning and produces grammatical outputs, but it can sometimes get the numbers wrong, and produces unrelated words in the middle.

For sentence 4, the best model does not translate out the name "Alex" but instead replaces with an unrelated word "surgeon". For sentence 7, the best model outputs "island" which does not relate to the source sentence. If we have time to inspect the training data in the future, we might be able to see if this can be an overfitting issue. Possibly there are similar combinations in the training data which leads to the model outputs those unrelated words.

## 8 Contributions of member

Weiqiu You:

- data processing,
- built and trained models
- built sequence length sampler
- built beam search decoder
- error analysis
- writings

## 9 Conclusion

We have spent a lot of time debugging on the small bugs that are not related to modeling, such as not feeding the start-of-sentence tokens into the decoder when switching to minibatch training, accidentally sorting the dataset when evaluating, memory issues, etc. Due to spending a enormous time fixing these problems, we only have 2 weeks left to tune hypterparameters of the baseline model, and do not enough have time to train $k$-span decoders with syntax to improve its performance. These bugs turn out to be simple, but can stop the project from moving forward for a long time. In future projects, we would have a better debugging methodology of checking one step at a time following the inputs to the outputs.

The result is as expected: as span size goes up, the performance degrades. However, we did not expect that the performance would degrade this much. This proves the need to incorporate syntax when generating multiple words at a time.

## 10 Future Work

The next step of the project is to first train the models of different spans to their best performance and know the limit of the model. To improve the performance, we can add copy mechanisms and try other attention mechanisms, etc. Then, we want to experiment a little bit more with different models to get state-of-the-art BLEU score. We also want to remove the two extra linear layers for the baseline 1-span model, to compare the true speedup.

After reaching state-of-the-art for 1-span baseline model, we will proceed to incorporate syntax into training. The initial thought is to first predict syntax labels for each phrase, and then use the label along with the previously predicted phrase to predict the next phrase. We will also experiment with other methods to improve performance of $k$-span models, such as adding positional embeddings within each span, find the best combination of words for whole spasn during beam search instead of score independently for each position of the span, etc.

## References


Aharoni, R. and Goldberg, Y. (2017). Towards string-to-tree neural machine translation. *Proceedings of the 55th Annual Meeting on Association for Computational Linguistics*.

Ashish Vaswani, Noam Shazeer, N. P. J. U. L. J. A. N. G. L. K. I. P. (2017). Attention is all you need. *31st Conference on Neural Information Processing Systems*.

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

FastBPE (2018). Fastbpe. https://github.com/glample/fastBPE.

Hao Zhou, Zhaopeng Tu, S. H. X. L. H. L. J. C. (2017). Chunk-based bi-scale decoder for neural machine translation. *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*.

Iyyer, M. and John Wieting, Kevin Gimpel, L. Z. (2018). Adversarial example generationwith syntactically controlled paraphrase networks. *Conference of the North American Chapter of the Association for Computational Linguistics*.

Jiatao Gu, James Bradbury, C. X. V. O. L. R. S. (2018). Non-autoregressive neural machine translation. *Sixth International Conference on Learning Representations*.

Kai Sheng Tai, Richard Socher, C. D. M. (2015). Improved semantic representations from tree-structured long short-term memory networks. *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*.

Lukasz Kaiser, Aurko Roy, A. V. N. P. S. B. J. U. N. S. (2018). Fast decoding in sequence models using discrete latent variables. *ICML*.

Minh-Thang Luong, Hieu Pham, C. D. M. (2015). Effective approaches to attention-based neural machine translation. *Conference on Empirical Methods in Natural Language Processing*.

Robertson, S. (2018). Translation with a sequence to sequence network and attention. https://pytorch.org/tutorials/intermediate/seq2seq_translation_tutorial.html.

SacreMoses (2018). Sacremoses. https://github.com/alvations/sacremoses.

Sennrich, R., Haddow, B., and Birch, A. (2016). Neural machine translation of rare words with subword units. *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*.

Shonosuke Ishiwatari, Jingtao Yao, S. L. M. L. M. Z. N. Y. M. K. W. J. (2017). Chunk-based decoder for neural machine translation. *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*.

Xingxing Zhang, Liang Lu, M. L. (2016). Top-down tree long short-term memory networks. *Conference of the North American Chapter of the Association for Computational Linguistics*.

Xinyi Wang, Hieu Pham, P. Y. G. N. (2018). A tree-based decoder for neural machine translation. *Conference on Empirical Methods in Natural Language Processing*.