# CS287 Homework 3: Translation

Shayne O'Brien shayneob@mit.edu
David McClure dclure@mit.edu

March 4, 2018

## 1 Introduction

In this problem set we train neural machine translation (NMT) systems using end-to-end networks on the IWSLT-2016 dataset. This corpus consists of Ted Talks translated between German and English. The utterances of each language are aligned, which allows us to use it to train translation systems. We implement (1) a sequence to sequence model as described by Sutskever et al. [5], (2) an attention model as introduced by Bahdanau et. al [1] with dot-product attention computation as per Luong et. al [3], beam search to improve translation quality, and (4) visualizations of how the attention mechanism makes a soft-alignment between words in the source and target sentences. We also experiment with the effects of batch size, masking padding, and teacher forcing on the validation set perplexity.

## 2 Problem Description

In machine translation, the objective is to translate sentences from a source language into a target language. In NMT, we approach this task by jointly training end-to-end deep neural networks to predict the likelihood of a sequence of output target tokens given a sequence of input tokens of a different language. More formally, we define machine translation as translating a source sentence $G_i = g_1,\ldots, g_J = g_i^{|G|}$ into a target sentence $E_i = e_1,\ldots, e_K = e_i^{|E|}$. Using this notation, we can define a translation system as the function

$$\hat{E} = \arg\max_{E} P(E|G;\theta), \tag{1}$$

where $\theta$ are the parameters of the model specifying the probability distribution. We learn $\theta$ from data consisting of aligned sentences in the source and target languages. In the case of this problem set, German is our source language and English is our target language.

## 3 Models and Algorithms

In the encoder-decoder framework, an encoder reads an input sequence into a context vector $c$ and a decoder is trained to predict an output token $\hat{e}_t$ given the context vector and all previously predicted tokens $\{\hat{e}_1,\ldots,\hat{e}_{t-1}\}$. In this way, the decoder defines a probability distribution over a translation $E$. Formally,

$$p(E) = \prod_{t=1}^{K} p(e_t | \{e_1, \ldots, e_{t-1}\}, c) \tag{2}$$

During generation for unlabeled input sequences, tokens are usually output until an end-of-sentence token is output or a specified maximum length is reached.

In the context of this homework, this conditional probability is modeled using recurrent neural networks (RNN). With this being the case, we let $p(e_t | \{e_1, \ldots, e_{t-1}\}) = z(e_{t-1}, s_t, c)$ for simplicity where $s_t$ is the hidden state of the RNN and $z$ is a nonlinear, possibly multi-layered function that outputs the probability of $e_t$ (Neubig, 2017).

### 3.1 Encoder-Decoder

Inspired by Cho et al. [2] and Sutskever et al. [5], the name encoder-decoder comes from the idea that we can use a model of two RNNs to translate between languages: one that processes an input sequence $G$ and "encodes" its information as a vector of real numbers (hidden state), and another that is used to "decode" this vector to predict the corresponding target sequence $E$. In particular if we let the encoder be expressed as $\text{RNN}^{(g)}(\cdot)$, then the decoder is expressed as $\text{RNN}^{(e)}(\cdot)$ and we can represent the overall model for time step $t$ as follows:

$$\begin{aligned}
\mathbf{m}_t^{(g)} &= M_{\cdot, g_t}^{(g)} \\
\mathbf{h}_t^{(g)} &= \begin{cases} \text{RNN}^{(g)}(\mathbf{m}_t^{(g)}, \mathbf{h}_{t-1}^{(g)}) & t \geq 1 \\ 0 & \textit{otherwise} \end{cases} \\
\mathbf{m}_t^{(e)} &= M_{\cdot, e_{t-1}}^{(e)} \\
\mathbf{h}_t^{(e)} &= \begin{cases} \text{RNN}^{(e)}(\mathbf{m}_t^{(e)}, \mathbf{h}_{t-1}^{(e)}) & t \geq 1 \\ \mathbf{h}_{|G|}^{(g)} & \textit{otherwise} \end{cases} \\
\mathbf{p}_t^{(e)} &= \text{softmax}(W_{hs}\mathbf{h}_t^{(e)} + b_s)
\end{aligned} \tag{3}$$

where $\mathbf{m}_t^{(g)}$ is the source language embedding lookup, $\mathbf{h}_t^{(g)}$ is the encoder hidden state, $\mathbf{m}_t^{(e)}$ is the source language embedding lookup, $\mathbf{h}_t^{(e)}$ is the decoder hidden state, and $\mathbf{p}_t^{(e)}$ is the softmax to turn $\text{RNN}^{(e)}$'s hidden state into a probability.

Note that $\mathbf{h}_0^{(e)}$ is initialized to $\vec{\mathbf{0}}$ and $\mathbf{h}_{|G|}^{(g)} = c$ the encoder has seen all words in the source sentence. In this setup, we are feeding the true $e_{t-1}$ target as the input to the decoder for time step $t$. This is known as teacher forcing. When we are not employing teacher forcing, as is the case in some of our experiments, $\mathbf{m}_t^{(e)}$ $M_{\cdot, \hat{e_{t-1}}}$ where $e_{t-1} = \arg\max[p_{t-1}^{(e)}]$, the most likely output token from the previous time step ().

### 3.2 Attention Decoder

To incorporate attention into our decoder, we redefine the conditional probability of $e_t$ to be

$$p(e_t | \{e_1, \ldots, e_{t-1}\}, G_i) = z(e_{t-1}, s_t, c_t) \tag{4}$$

Here and unlike the formulation of $p(e_t)$ in previous sections, the probability is conditioned on a distinct context vector $c_t$ for each target token $e_t$. The context vector $c_t$ depends on a sequence of annotations $h_1, \ldots, h_t$ to which an encoder maps the input sequence. The context vector $c_t$ is then computed as a weighted sum of these annotations:

$$c_t = \sum_{j=1}^{J} \alpha_{tj} h_j \tag{5}$$

where

$$\alpha = \frac{exp(e_{tj})}{\sum_{k=1}^{G_i} exp(e_{tk})} \tag{6}$$

and

$$e_{tj} = A(s_{t-1}, h_j) \tag{7}$$

is an *alignment model* which scores how well the inputs around position $j$ and the output at position $t$ match. The alignment model $A$ is parametrized as a simple feedforward neural network that is jointly trained with all other components of the translation system. Note that the alignment is not a latent variable, which allows backpropagation through this gradient due to its differentiability. Intuitively, $\alpha_{tj}$ reflects the importance of $h_j$ with respect to $s_{t-1}$ in deciding $s_t$ and generating $e_t$. It is believed that giving the decoder an attention mechanism relieves the encoder from having to encode all information in the source sequence needed for translation to a fixed-length vector (Bahdanau et al. [1]).

### 3.3 Beam Search

Since the model is only trained to predict the probability of the next word at a given time step in the target sentence, a decoding procedure is needed to produce an actual translation, where the goal is to maximize the probability of all words in the target sentence:

$$p(E) = \prod_{t=1}^{K} p(e_t | \{e_1, \ldots, e_{t-1}\}, c) \tag{8}$$

The simplest approach is to proceed in a greedy fashion, taking the most likely word under the model at each time step and feeding it back in as the next input until the end-of-sentence token is produced:

$$\hat{e}_t = \underset{i}{\text{argmax}}\, p_{t,i}^{(e)} \tag{9}$$

This is not guaranteed to produce the highest-scoring sentence, though, since in some cases the selection of a lower-probability word at a given time-step will lead to downstream predictions that produce a higher-scoring sentence overall. To account for this, we follow Sutskever et al.[5] and use a beam search – instead of greedily accumulating a single series of tokens across time-steps in the target sentence, we instead keep track of a fixed-size set of candidate decodings (the "beam"). At each step, we extend each candidate with all words in the vocabulary, update the cumulative score for each new candidate given the probabilities produced by the model, and then

skim off the $b$ highest-scoring candidates to keep the size of the beam constant. (Otherwise it would be a breadth-first search of all possible permutations.) After the new beam is pruned, any candidates ending with the end-of-sentence token are removed from the beam and added to a set of completed candidates, and the beam size $b$ is reduced by 1. This continued until the beam is empty, and $b$ complete candidate sentences have been produced.

If the final set of candidate sentences is sorted on the summed log-probabilities of each word, the procedure will strongly favor short sentences, since the addition of each new word will drive down the joint probability of the sentence. A range of strategies for length normalization have been proposed, but we take the simplest approach, which is to divide the total log-probability of the sentence by the number of words to get an average per-word log-probability.

$$\hat{E} = \underset{E}{\operatorname{argmax}} \, logP(E|F)/|E| \tag{10}$$

Following Sutskever,[5] we use beam size of 10. And, in the interest of speed – when extending candidates with new words at each time-step, we only consider the 1000 most frequent words under the model, which gives a significant speedup at little or no cost in accuracy.

# 4 Experiments

We experimented with a number of modifications to the baseline attention model:

1. In addition to the standard approach for calculating attention where the hidden layer $h_t^e$ is dotted with the hidden states at each time-step from the encoder:

$$\text{attn\_score}(H^{(f)}, h_t^{(e)}) := H_j^{(f)\intercal} h_t^{(e)} \tag{11}$$

   We also implemented the attention scoring as a multi-layer perceptron as described by Bahdanau et al [1]:

$$\text{attn\_score}(h_t^{(e)}, h_j^{(f)}) := w_{a2}^{\intercal} \tanh(W_{a1}[h_t^{(e)}; h_j^{(f)}]) \tag{12}$$

2. We tried a range of learning rate schedulers – train for a fixed number of epochs (4, 8, 10) and then decay by 0.5 each epoch after that (as described in the reference model); decay by 0.5 every $N$ epochs, where $N \in \{2, 3, 4\}$; and various values for the decay ratio gamma.

3. We implemented the decoder as a bidirectional LSTM as described by Bahdanau,[1] which in theory should allow the encoder to capture information about both the left- and right-hand contexts at each time step.

4. Last, we experimented with a wide range of hyperparameter settings:

   - Word embedding dimensions: 200, 300, 500.
   - LSTM hidden layers: 1, 2, 4.
   - LSTM hidden layer dimensions: 200, 500, 1000.
   - SGD and Adam optimizers.

- Batch lengths of 32 and 64.
- Clipping gradients to 5.

Though we didn't have the resources to conduct a full grid search across all combinations of these architectures and parameters, our best performing models used 2-layer LSTMs with dot-product attention. All final models were optimized with SGD starting with a learning rate of 1. Early stopping was used to identify the best-performing model on the validation set.

| Model | Perplexity |
|---|---|
| ENCODER-DECODER (NO ATTENTION) | 15.612 |
| DOT ATTENTION – 300D EMBEDDINGS, 500D LSTMS, BIDIRECTIONAL ENCODER | 9.680 |
| DOT ATTENTION – 500D EMBEDDINGS, 1000D LSTMS | 9.672 |
| DOT ATTENTION – 200D EMBEDDINGS, 200D LSTMS | 9.636 |
| DOT ATTENTION – 300D EMBEDDINGS, 500D LSTMS | **9.501** |

Generally, we found that it was difficult to find a single learning rate scheduler that worked well for different hyperparameter settings – a decay rate that seemed to work well for the 500-unit LSTM seemed too slow for the 200-unit model, etc. Given indefinite resources, ideally we would run a broad grid search to pick the best model.

See the appendix for visualizations of the attention weights and comparisons between the translations produced by the beam coder and the Google Translate predictions.

## 5 Conclusion

We trained two classes of models – a basic encoder-decoder architecture as described by Sutskever,[5] and a series of models that use dot-product attention to give the decoder a more flexible representation of the input. Though we experimented with more complex architectures – bidirectional LSTMs and multilayer perceptrons for attention weighting – our best-performing models used the basic dot-product attention.

## References


[1] Bahdanau, D., Cho, K. & Bengio, Y. Neural Machine Translation by Jointly Learning to Align and Translate. (2014). doi:10.1146/annurev.neuro.26.041002.131047

[2] Cho, K. et al. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. (2014). doi:10.3115/v1/D14-1179

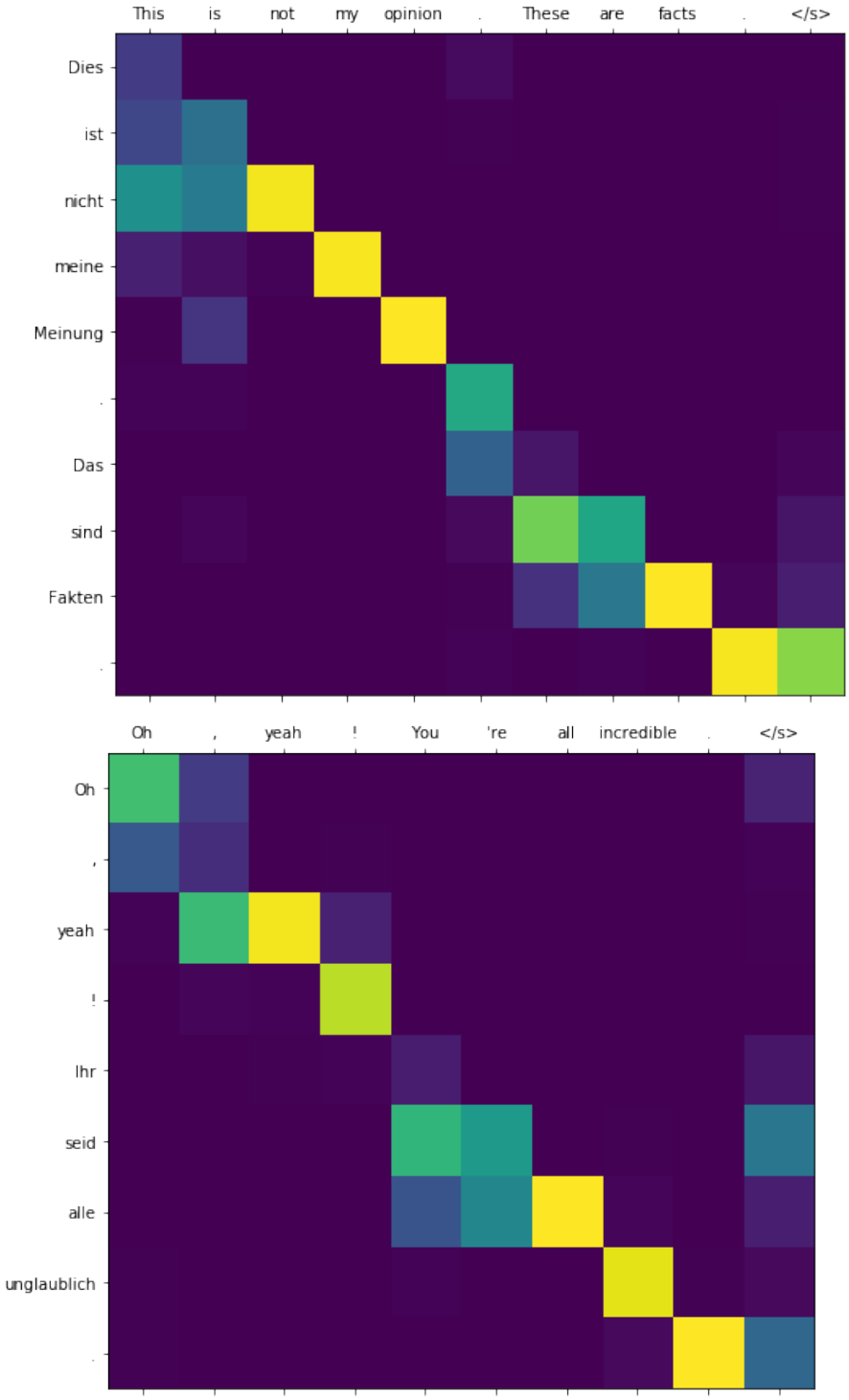
[3] Luong, M.-T., Pham, H. & Manning, C. D. Effective Approaches to Attention-based Neural Machine Translation. (2015). doi:10.18653/v1/D15-1166

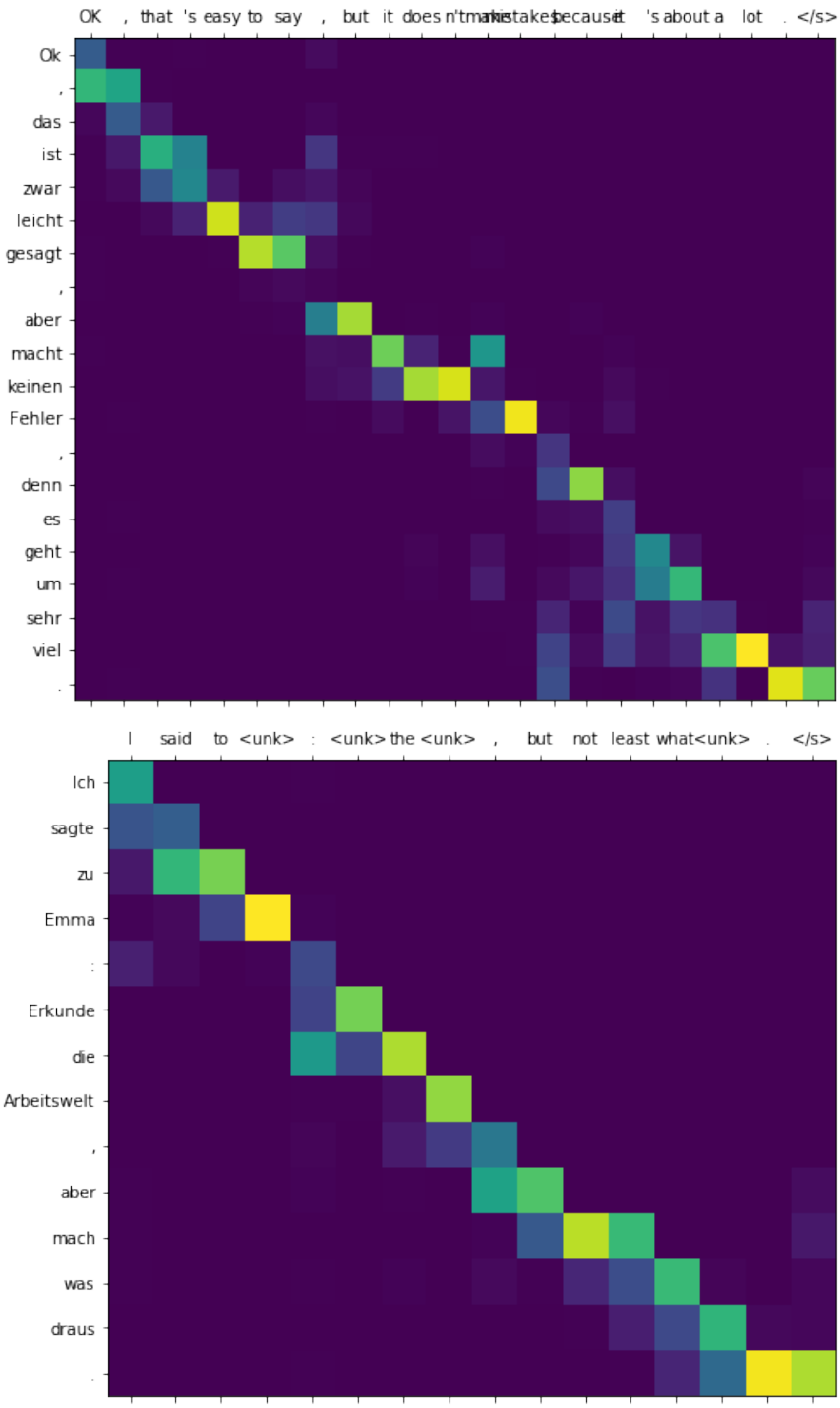
[4] Neubig, G. Neural Machine Translation and Sequence-to-sequence Models: A Tutorial. (2017).

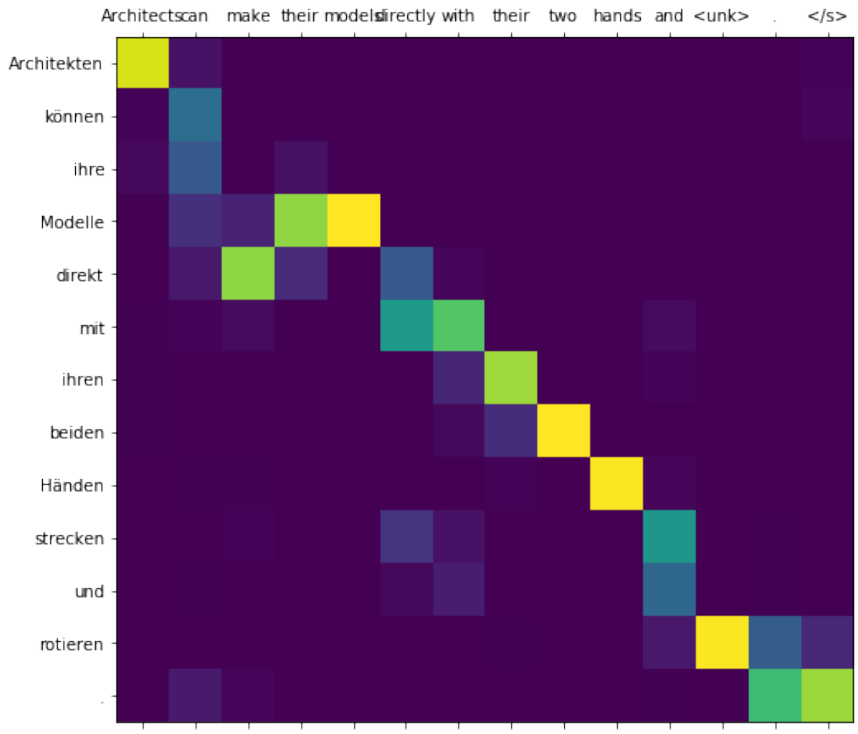
[5] Sutskever, I., Vinyals, O. & Le, Q. V. Sequence to sequence learning with neural networks. Advances in Neural Information Processing Systems (NIPS) 31043112 (2014). doi:10.1007/s10107-014-0839-0

# Appendix A Attention visualizations

We can visualize the dot-product attention by plotting the weights assigned to the encoder hidden states for each step during decoding (using greedy decoding, here).




OK , that 's easy to say , but it does n't make a mistake because it 's about a lot . </s>
Ok
,
das
ist
zwar
leicht
gesagt
,
aber
macht
keinen
Fehler
,
denn
es
geht
um
sehr
viel
.
I said to <unk> : <unk> the <unk> , but not least what <unk> . </s>
Ich
sagte
zu
Emma
:
Erkunde
die
Arbeitswelt
,
aber
mach
was
draus
.




## Appendix B Translation examples

We can compare the translations produced by the beam search decoder to the predictions from Google translate:

1. 
   - **Source**: `Arbeit kam spter, Heiraten kam spter, Kinder kamen spter, selbst der Tod kam spter.`
   - **Our model**: `work came later , <unk> came later later , children came later later , even the death came later .`
   - **Google**: `Work came later, marriages came later, children came later, even death came later`
2. 
   - **Source**: `Das ist es , was Psychologen einen Aha-Moment nennen .`
   - **Our model**: `That 's what psychologists call a <unk> call .`
   - **Google**: `That's what psychologists call an aha moment.`
3. 
   - **Source**: `Dies ist nicht meine Meinung . Das sind Fakten .`
   - **Our model**: `This is not my opinion . These are facts .`
   - **Google**: `This is not my opinion. These are facts.`
4. 
   - **Source**: `In den 20ern sollte man sich also weiterbilden ber den Krper und die eigenen Mglichkeiten`

- **Our model**: So in the <unk> , you should be thinking about the body and the own possibilities .
- **Google**: In the 20’s you should continue to educate yourself about the body and your own possibilities

5.
- **Source**: Wie wrde eine solche Zukunft aussehen ?
- **Our model**: What would such a future look like ?
- **Google**: What would such a future look like?