Neural Machine Translation by Jointly Learning to Align and Translate

github.com/ShlomoLibo/Attentional-Translation

Introduction and Discussion

Machine Translation

We define the task of machine translation as mapping sentences in one language, which we call the "source" language, into an another language, which we call the "target" language.

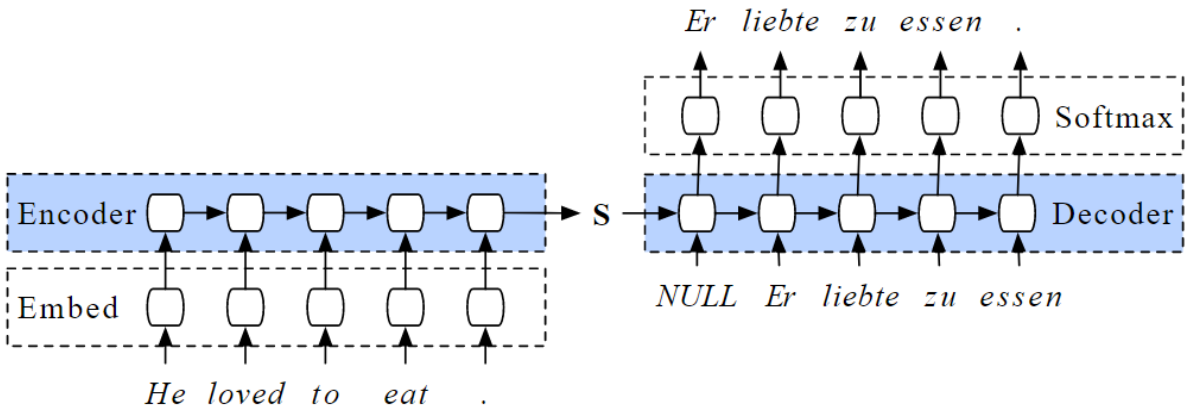
Traditionally, two main approaches exist for the task of machine translation.
One approach, is to make use of the human knowledge and understanding of grammatical rules, and their mapping between the source and the target language. This set of rules is used to create a coherent, grammatically correct sentence-to-sentence translation.
Second approach, is to leverage neural-network architectures (sometimes called “Neural Machine Translation”). This allows to solve the problem without the guidance of humans. Instead large corpora of data are used, which consist of large amounts of sentence pairs from both languages, which have a shared meaning.
Typically, using neural-networks for this purpose involves a two-stage architecture. This architecture consists of an encoder, which outputs a vector that represents the source sentence, and a decoder, which takes the encoded vector as in input and outputs a sentence in the target language [1] [2].

Previous Work on Neural Machine Translation

Here we describe in more detail the models by *Cho et al. 2014* [1] and *Sutskever et al. 2014* [2],
Both state-of-the-art models (at the time of this paper), can be described in the following manner:

Let $x = (x_1, \dots, x_{T_x})$ an input sentence, or an embedding thereof.
Using an RNN with a hidden state $h_t = f(x_t, h_{t-1})$, we produce a vector $c = q(\{h_1, \dots, h_{T_x}\})$.
Then the probability of the words y_t are modeled using the decoder hidden state s_t and $p(y_t \mid \{y_1, \dots, y_{t-1}\}, c) = g(y_{t-1}, s_t, c)$ (a vector of probabilities for each possible word).
During Testing we usually take $y_t = argmax_{y_t}\, p(y_t \mid \{y_1, \dots, y_{t-1}\}, c)$.

An Illustration of *Sutskever et al. 2014* ($q(\{h_1, \dots, h_T\}) = h_T$):



Method

One problem that arises from using the encoder-decoder paradigm, is that the sentence at one point is compressed to a single vector, and every part of the sentence is derived from it.
We view the transition between the encoder stage and the decoder stage as an unnecessary bottleneck. When a human translator translates a sentence, they don't look at the whole sentence once, but instead for every word they might look a different part of the sentence.

We arrive at the idea of **attention** presented by *Dzmitry et al.* [3] – instead of compressing the sentence to a single encoding, we essentially allow the decoder to "look" at the original sentence for each new word it outputs. Concretely, the following architecture is introduced:

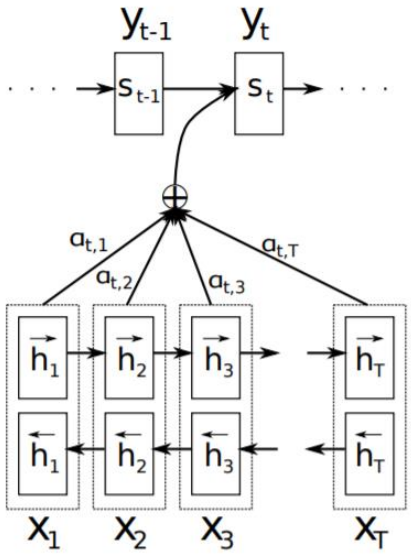
Encoder + Attention

Let $x = (x_1, \dots, x_{T_x})$ an input sentence, or an embedding thereof.
We use a 2-way RNN with a hidden state $h_t = (\overrightarrow{h_t}, \overleftarrow{h_t})$ where $\overrightarrow{h_t} = \vec{f}(x_t, h_{t-1})$ and $\overleftarrow{h_t} = \overleftarrow{f}(x_t, h_{t+1})$, to create at time i the context vector $c_i = \Sigma_{j=1}^{T_x} a_{ij} h_j$.
c_i is a weighted sum of the hidden states h_t, and so the coefficients represent how much "attention" each word gets. We note that we are not guaranteed by the architecture that the attention placed will correspond to human intuition, mainly because the weighted sum is of the hidden states, and not the words / embedding themselves.
a_{ij} computed by $a_{ij} = \frac{\exp(e_{ij})}{\Sigma_{k=1}^{T_x} \exp(e_{ik})}$ where $e_{ij} = a(s_{i-1}, h_j)$ and s_i is the hidden state of the decoder.

Decoder

Similarly to previous neural machine translation methods, the decoder described by a network g, where the probability of the words y_t are modeled using the decoder hidden state s_t and $p(y_t \mid \{y_1, \dots, y_{t-1}\}, c_t) = g(y_{t-1}, s_t, c_t)$. We draw attention to the main difference between the presented approach and previous approaches – instead of having a single context vector c for every time t, we may have a different context vector c_t for each new word.

An Illustration of the described approach:



The loss chosen for this task is cross-entropy loss between the $g(y_{t-1}, s_t, c_t)$ and the one-hot vector representing the ground truth. This allows for training using gradient-based optimization methods.

Evaluation and Results

For evaluating the described method, the architecture was trained on the WMT English-German dataset. Before doing any further experiments, we assess the accuracy of our network by testing it on a testing dataset and sampling a few examples.

Overall, if we use cross-entropy between the predictions of the model and one-hot vectors representing the correct word, as a measure of accuracy of prediction, we get a score of 0.381 compared to 1.272 of a randomly initialized model.

Cross-entropy is chosen as the quantitative metric for model performance, since cross-entropy allows for uncertainty in the model prediction. This is helpful in the case of machine translation, as a sentence might have multiple possible translations, each with a different word choice and sentence structure.

When sampling sentences from the test-dataset we get the following results:

- **German:** eine frau spielt volleyball.
  English: a woman is playing volleyball.
  Translation: a woman is playing volleyball.

- **German:** drei teenager in einer u-bahn albern herum.
  English: three teenagers in a subway playing around
  Translation: three teenagers are in a subway are in around..

- **German:** ein blondes kind schaukely auf einer schaukel.
  English: a blond child swinging on a swing.
  Translation: a blond child is swinging swing swing swing

- **German:** drei hunde spielen auf einem feld.
  English: three dogs play with each other out in the field.
  Translation: three dogs are playing a field field

- **German:** zwei manner mit munzen
  English: two men wearing hats.
  Translation: two men wearing hats hats hats

- **German:** zwei braune hunde rennen durch den schnee.
  English: two brown dogs are running through the snow.
  Translation: two brown dogs running through the snow

- **German:** feuerwehrmanner kommen aus einer u-bahnstation.
  English: firemen are coming out a subway station.
  Translation: firemen are out out a a.

- **German:** ein radfahrer springt ueber eine hindernis.
  English: a biker jumps an obstacle.
  Translation: a cyclist jumps over a hurdle.

We conclude the model works well, although not perfectly, as neural machine translation is one of the most challenging tasks in the field of machine learning.

Choosing German as the source language allows to examine the behavior of the method on the following phenomena, unique to the German language:

Compound words

In German, sometimes words that are separate in other languages are compounded into a single word. This is why German is famous for having long words (*Mamutwoerter* – mammoth words).

For example:

- *Was ist deine Liblingsfarbe?*

Is translated to:

- *What is your favorite color?*

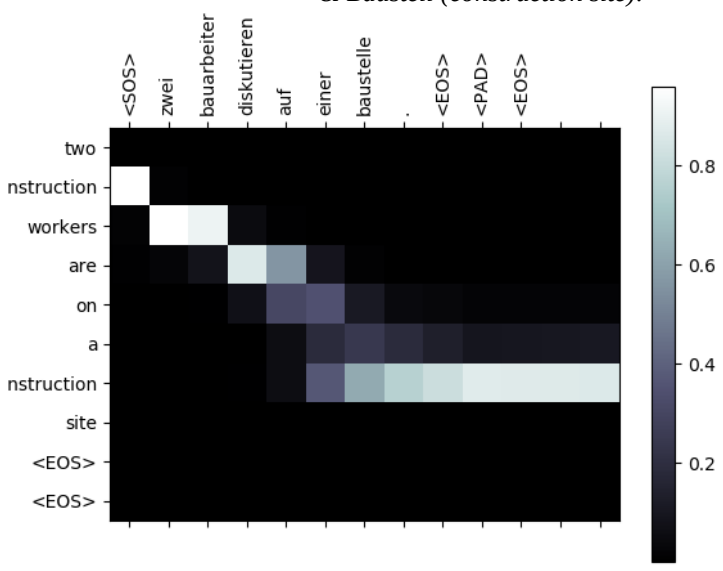
Where "*Liblings*" corresponds to "*favorite*" and "*farbe*" corresponds to "*color*".

It is interesting to see whether the model will place its attention both when translating "favorite" and when translating "*farbe*", or the attention will be placed only when translating one of the words and the rest will be carried through the hidden sates.

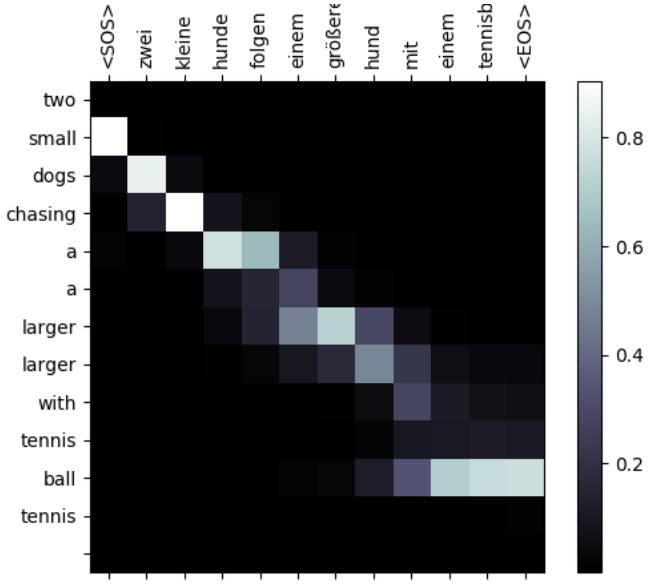
Results

Here are the attention allocation results for a few instances of compound words:

Bauarbeiter (construction workers) & Baustell (construction site):



Tennisball (tennis ball)



We can see that in all three instances, attention is placed only when translating one of the compound-words. This is actually logical, since when parsing a compound-word it might be more efficient to parse it completely and transfer the result through the hidden state, then parsing it again later.

Separable Verbs

In German, one verb can be separated across the sentence. For example, the sentence:

- *Ich mache meine Auge zu.*

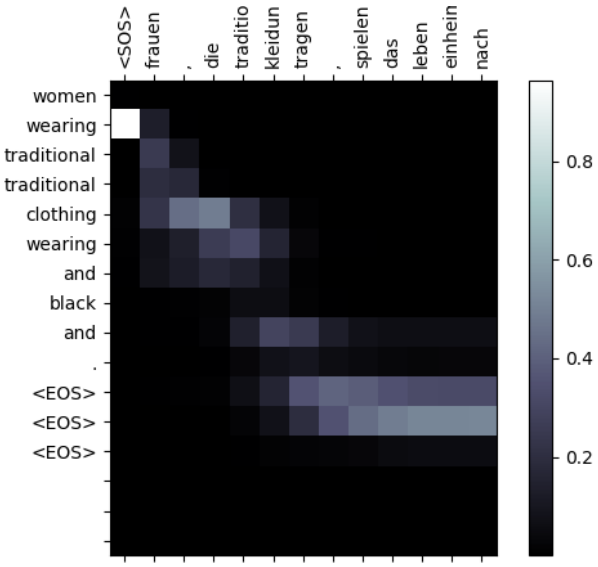
Translates to:

- *I close my eyes.*

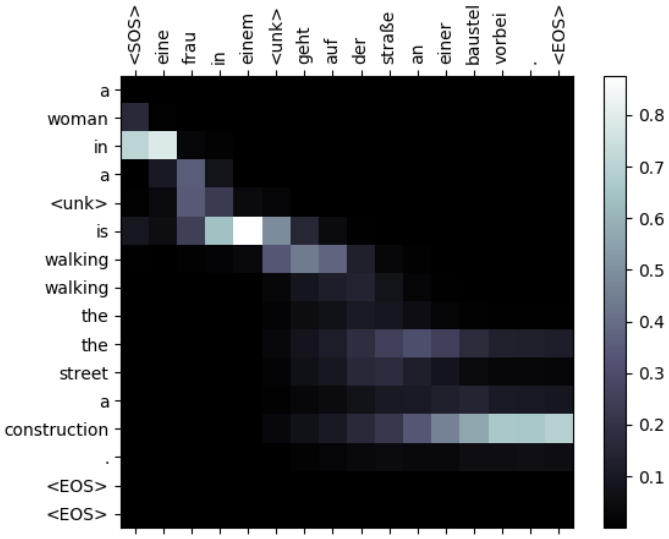
Both "*mache*" and "*zu*" together represent a single verb, "*zumachen*" (to close). Without the information that "zu" is in the end of the sentence, the original sentence can also be "*Ich mache meine Auge auf"* (I open my eyes). Therefore, we expect that after translating "I", attention will be placed both at "*mache*" and "*zu*".

Results

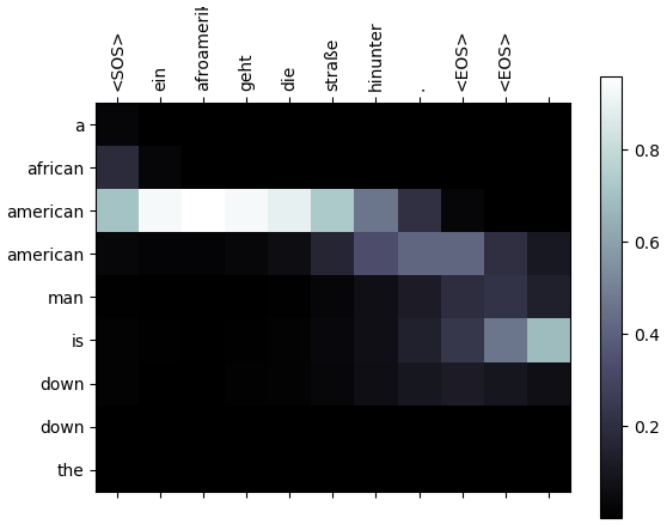
nachspielen (to replay):



vorbeigehen (walking past):



Hinhuntergehen (to go down):



Overall, we see that the model struggles with seperable verbs. This can be explained by the fact that usually attention is placed locally, and it's hard for the model to learn the edge cases where it should place the attention both at the beginning and the end of a sentence. Still, exploring how to make the model place its attention past the current word's local region can be an interesting topic of reseach.

It should be noted that by doing these experiments we also confirmed a result in the original paper. Namely, that the model generally places attention where we expect it to be.

References

[1] Cho et al., "Learning Phrase Representations using RNN Encoder–Decoder," 2014.

[2] Sutskever et al., "Sequence to Sequence Learning," *nips,* 2014.

[3] Dzmitry et al., "Neural Machine Translation by Jointly Learning to Align and Translate," *ICLR,* 2015.