Lecture 8: Machine Translation and Sequence-to-Sequence Models

André Martins & Vlad Niculae

Deep Structured Learning Course, Fall 2019

Announcements

- The deadline for the project midterm report is November 25 and the final report is due January 6. The class presentations will be in January 10, 13, and 17.
- Homework 2 grades and solutions are out.
- The deadline for turning in Homework 3 is November 22.
- No class next Monday, November 18!
- Due to dissertation season, we need to change the room for the November 22 class to V1.10. Ben Peters will teach this class.

Today's Roadmap

Last lecture we talked about sequence tagging and sequence generation. Today we'll talk about sequence-to-sequence models.

- Machine translation
- Sequence vector representation
- Encoder-decoder architecture
- Sequence matrix representation
- Attention mechanism
- Encoder-decoder with attention
- Convolutional sequence-to-sequence models
- Self-attention and transformer networks
- Pre-trained models and transfer learning

Sequence-to-Sequence

Sequence-to-sequence models map a source sequence (of arbitrary length) into a target sequence (also of arbitrary length)

Note: This is different from **sequence tagging**, where we assume the two sequences are of the same size

Example: Machine Translation

Goal: translate a **source sentence** $\boldsymbol{x}$ in one language into a **target sentence** $\boldsymbol{y}$ in another language.

Example (Portuguese to English):

$\boldsymbol{x}$: *"A ilha de Utopia tem 200 milhas de diâmetro na parte central."*

↓

$\boldsymbol{y}$: *"The island of Utopia is two hundred miles across in the middle part."*

Outline

1. **Statistical Machine Translation**

2. **Neural Machine Translation**

   Encoder-Decoder Architecture

   Encoder-Decoder with Attention

   Convolutional Encoder-Decoder

   Self-Attention and Transformer Networks

3. **Transfer Learning with Transformers**

4. **Conclusions**

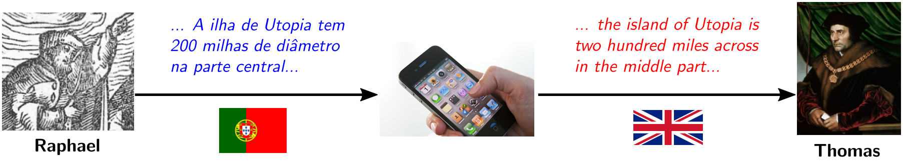
1950s: Early Machine Translation

(Source: `https://youtu.be/K-HfpsHPmvw`)

- MT research began in early 1950s
- Mostly Russian-English (motivated by the Cold War!)
- Systems were mostly rule-based, using a bilingual dictionary

Noisy Channel Model (Shannon and Weaver, 1949)

"When I look at an article in Russian, I say: 'This is really written in English, but it has been coded in some strange symbols. I will now proceed to decode.' "

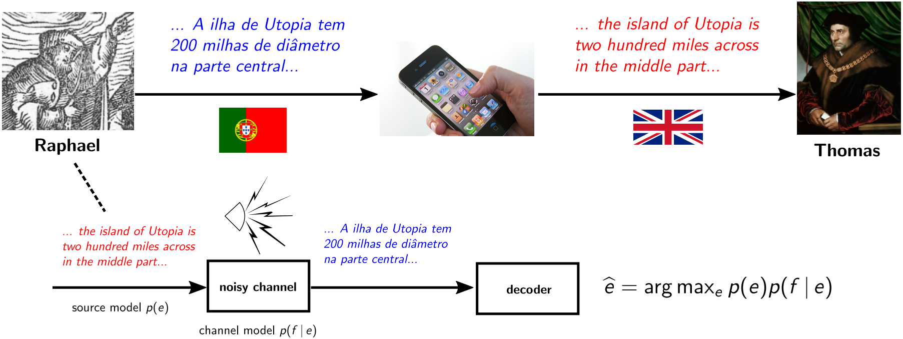


Raphael

... A ilha de Utopia tem 200 milhas de diâmetro na parte central...

... the island of Utopia is two hundred miles across in the middle part...

Thomas



Raphael
... A ilha de Utopia tem 200 milhas de diâmetro na parte central...
... the island of Utopia is two hundred miles across in the middle part...
Thomas
... the island of Utopia is two hundred miles across in the middle part...
source model $p(e)$
noisy channel
channel model $p(f \mid e)$
... A ilha de Utopia tem 200 milhas de diâmetro na parte central...
decoder

$$\widehat{e} = \arg\max_e p(e)p(f \mid e)$$

A very simple model: builds a generative story that works "backwards" (flips source and target)

Yet: the dominant paradigm in MT for several decades (until 2014)

2014 was the year of neural machine translation (later)

1990s-2010s: Statistical Machine Translation

Goal: find the best English sentence $\boldsymbol{y}$, given Russian sentence $\boldsymbol{x}$

$$\widehat{\boldsymbol{y}} = \arg\max_{\boldsymbol{y}} \mathbb{P}(\boldsymbol{y} \mid \boldsymbol{x})$$

Key idea: use Bayes' rule to break this down into two components:

$$\widehat{\boldsymbol{y}} = \arg\max_{\boldsymbol{y}} \mathbb{P}(\boldsymbol{x} \mid \boldsymbol{y})\mathbb{P}(\boldsymbol{y})$$

- **Translation model:** models how words/phrases are translated (learnt from parallel data)
- **Language model:** models how to generate fluent English (learn from monolingual data)

How to Learn the Language Model?

Need large amounts of monolingual data (easy to get for most languages).

How to learn a language model from these data?

How to Learn the Language Model?

Need large amounts of monolingual data (easy to get for most languages).

How to learn a language model from these data?

We covered language models in previous lectures:

- Markov models with smoothing (e.g. Kneser-Ney)
- Neural language models
- ...

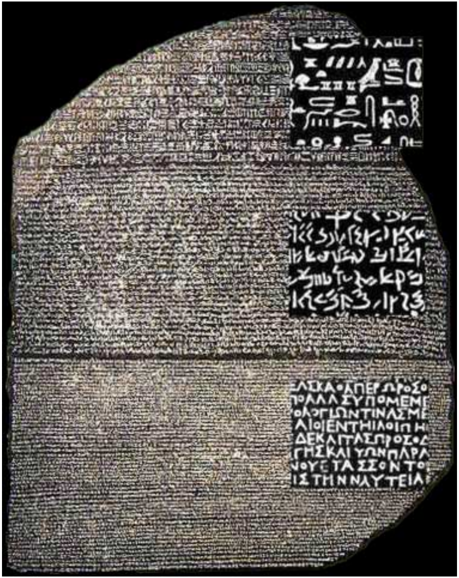
Pick your favorite!

How to Learn the Translation Model?

Need large amounts of parallel data!

(i.e. pairs of human translated Russian/English sentences.)

Rosetta Stone



- (Re-)discovered in 1799 near Alexandria
- Parallel corpora: ancient Egyptian, demotic Egyptian, ancient Greek

Europarl

- Proceedings from the European parliament sessions, translated into all EU official languages
- Around $\sim$ 1M parallel sentences for some language pairs
- Other corpora: Hansard, MultiUN, News Commentary, Wikipedia, OpenSubtitles, Paracrawl, ...

1990s: IBM Models for Statistical MT

How to learn the translation model $\mathbb{P}(\boldsymbol{x} \mid \boldsymbol{y})$?

Assume we have enough parallel training data.

Break it down further: consider instead

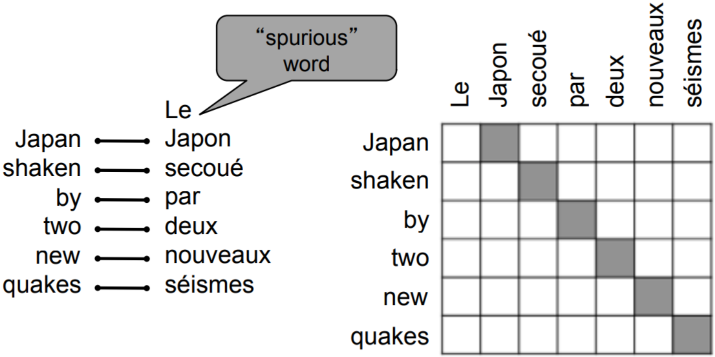
$$\mathbb{P}(\boldsymbol{x}, \boldsymbol{a} \mid \boldsymbol{y}),$$

where $\boldsymbol{a}$ are word alignments, i.e., word-level correspondences between Russian sentence $\boldsymbol{x}$ and English sentence $\boldsymbol{y}$

Word alignments are generally a latent variable at training time, and need to be marginalized over at test time.

Word Alignments

Example for English-French:

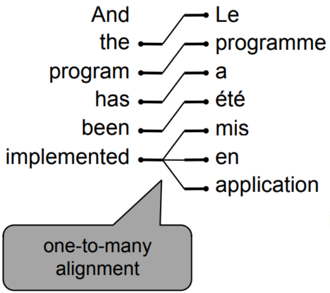


	Le	Japon	secoué	par	deux	nouveaux	séismes
Japan		▨					
shaken			▨				
by				▨			
two					▨		
new						▨	
quakes							▨

Some words may be unaligned (no counterpart in the other language)!

Word Alignments

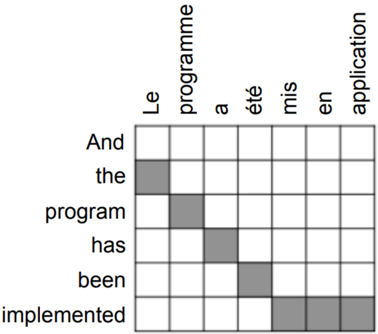
Alignment can be one-to-many (word fertility):



And the program has been implemented

Le programme a été mis en application

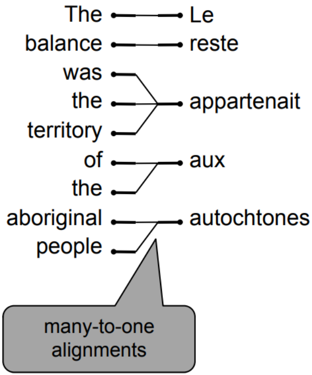
one-to-many alignment

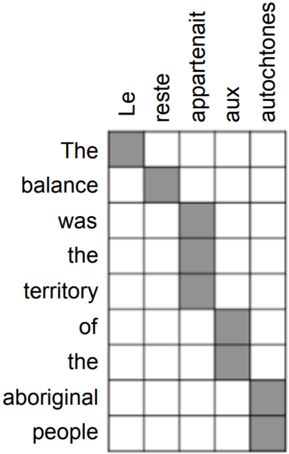


	Le	programme	a	été	mis	en	application
And							
the	■						
program		■					
has			■				
been				■			
implemented					■	■	■

Word Alignments

Alignment can be many-to-one:

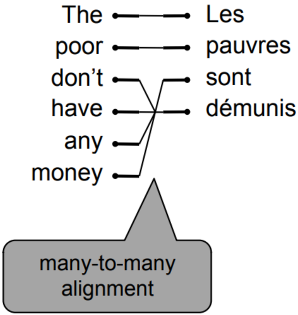


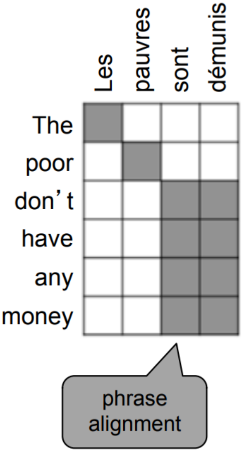


	Le	reste	appartenait	aux	autochtones
The	■				
balance		■			
was			■		
the			■		
territory			■		
of				■	
the				■	
aboriginal					■
people					■

Word Alignments

Alignment can be many-to-many (phrase-level): phrase-based MT:





1990s: IBM Models for Statistical MT

How to learn the translation model $\mathbb{P}(\boldsymbol{x} \mid \boldsymbol{y})$?

Assume we have enough parallel training data.

Break it down further: consider instead

$$\mathbb{P}(\boldsymbol{x}, \boldsymbol{a} \mid \boldsymbol{y}).$$

We learn $\mathbb{P}(\boldsymbol{x}, \boldsymbol{a} \mid \boldsymbol{y})$ as a combination of several factors:

- Probability of particular words aligning (co-occurrence, relative position, etc.)
- Probability of words having a particular fertility
- ...

This leads to IBM models 1, 2, 3, 4, ...

1990s: IBM Models for Statistical MT

To search the best translation, we need to solve

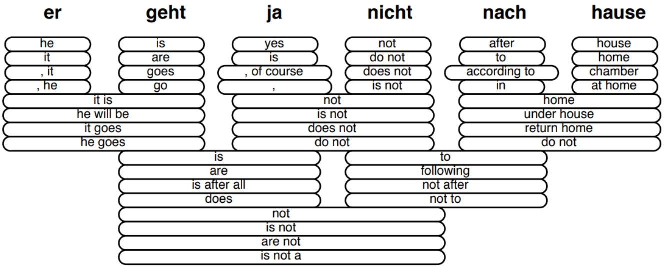
$$\widehat{\boldsymbol{y}} = \arg\max_{\boldsymbol{y}} \sum_{\boldsymbol{a}} \mathbb{P}(\boldsymbol{x}, \boldsymbol{a} \mid \boldsymbol{y})\mathbb{P}(\boldsymbol{y}),$$

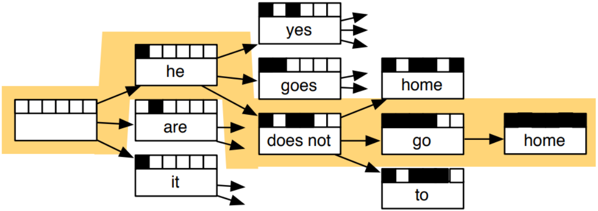
combining the translation and language models.

Enumerating all possible hypothesis and alignments is intractable.

Typical approach: heuristic search to gradually build the translation, discarding hypotheses that are too low probability.

Searching for the Best Translation





(Slide credit: `https://web.stanford.edu/class/cs224n/lectures/lecture10.pdf`)

To Sum Up: Statistical Machine Translation

We only saw the tip of the iceberg: SMT is (was?) a huge research field.

- The best systems are extremely complex
- It's a big pipeline with many separately-designed subcomponents (translation and language model are only two examples)
- Lots of feature engineering
- System design is very language dependent
- Require compiling and maintaining extra resources (e.g., phrase tables)
- Models are disk/memory hungry
- Lots of human effort to maintain.

2014: Neural Machine Translation

Outline

1. Statistical Machine Translation

2. **Neural Machine Translation**

   Encoder-Decoder Architecture

   Encoder-Decoder with Attention

   Convolutional Encoder-Decoder

   Self-Attention and Transformer Networks

3. Transfer Learning with Transformers

4. Conclusions

What is Neural Machine Translation?

- A way to do MT with a single neural network
- The system is trained end-to-end with parallel data (no more complex pipelines!)
- The underlying architecture is an encoder-decoder (also called a **sequence-to-sequence model**)
- To be rigorous, neural MT is also *statistical*; however, historically, “statistical MT” refers to non-neural models, and “neural MT” to neural network based models.

Outline

1. Statistical Machine Translation

2. **Neural Machine Translation**

   Encoder-Decoder Architecture

   Encoder-Decoder with Attention

   Convolutional Encoder-Decoder

   Self-Attention and Transformer Networks

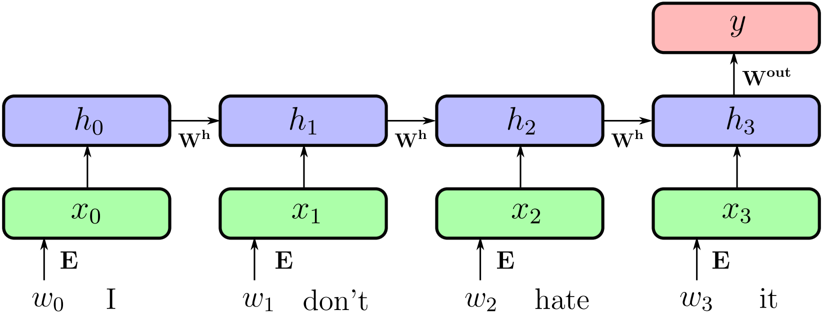
3. Transfer Learning with Transformers

4. Conclusions

Recap: Recurrent Neural Networks

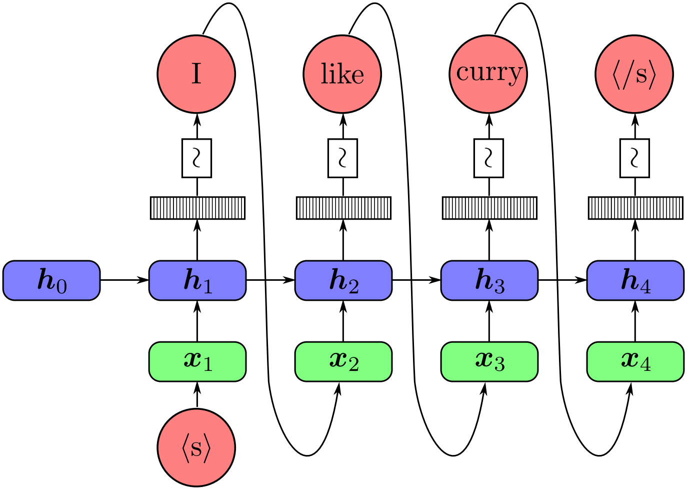
In the last lecture, we covered RNNs and we saw they can have several usages...

Recap: RNNs for Pooled Classification



(Slide credit: Ollion & Grisel)

Recap: Auto-Regressive RNNs for Sequence Generation



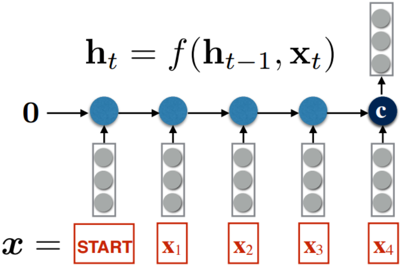
Sequence-to-Sequence Learning (Cho et al., 2014; Sutskever et al., 2014)

Can we put the two things together?

Idea:

1. An encoder RNN to encode the source sentence and generate a vector state
2. A decoder RNN to generate the target sentence **conditioned on that vector state**.

Encode a Sequence as a Vector



(Slide credit: Chris Dyer)

What is a vector representation of a sequence $\boldsymbol{x}$?

$$\boldsymbol{c} = \textbf{RNN}(\boldsymbol{x})$$

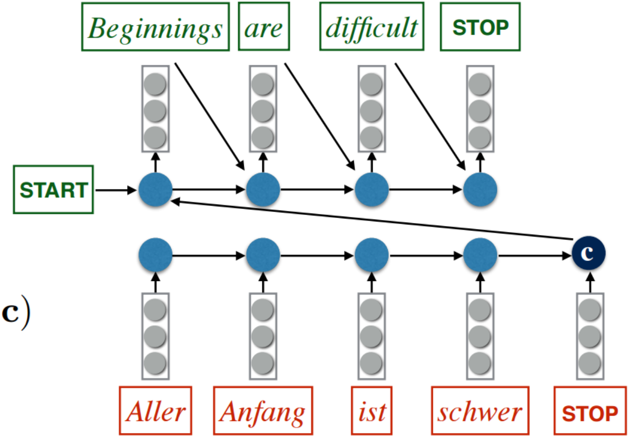
What is the probability of a sequence $\boldsymbol{y} \mid \boldsymbol{x}$?

$$\boldsymbol{y} \mid \boldsymbol{x} \sim \textbf{RNNLM}(\boldsymbol{c})$$

Encoder-Decoder Architecture

$$\mathbf{c} = \mathrm{RNN}(\boldsymbol{x})$$

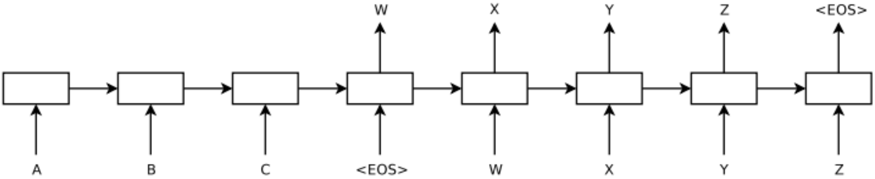
$$\boldsymbol{y} \mid \mathbf{c} \sim \mathrm{RNNLM}(\mathbf{c})$$



(Slide credit: Chris Dyer)

Encoder-Decoder Architecture

Another way of depicting it (from Sutskever et al. (2014)):



Some Problems

If the source sentence is long, the encoder may forget the initial words and the translation will be degraded

- Poor man's solution: reverse the source sentence.

The decoder does greedy search—this leads to error propagation

- Solution: beam search.

Beam Search

Ideally we want to find the target sentence $\boldsymbol{y}$ that maximizes

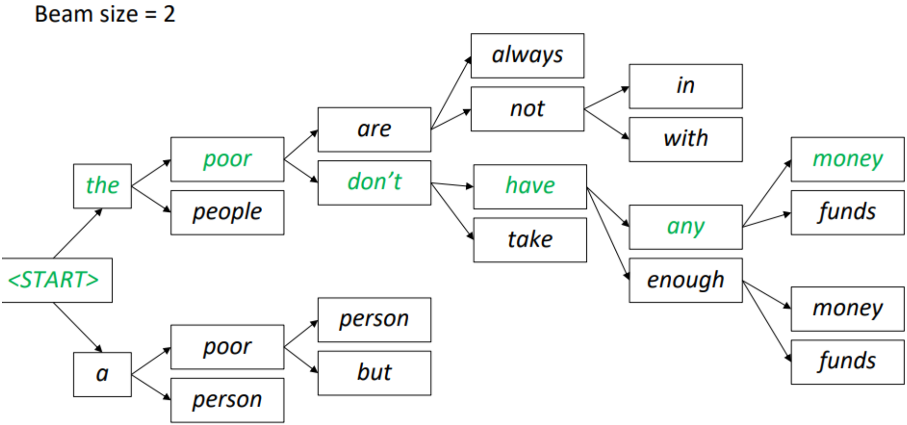
$$\mathbb{P}(\boldsymbol{y} \mid \boldsymbol{x}) = \prod_{i=1}^{L} \mathbb{P}(y_i \mid y_{1:i-1}, \boldsymbol{x})$$

Enumerating all $\boldsymbol{y}$ is intractable!

Beam Search:

- an approximate search strategy
- on each step of the decoder, keep track of the k most probable partial translations; discard the rest
- k is the beam size
- if $k = 1$, we recover greedy search.

Beam Search



(Source: `https://web.stanford.edu/class/cs224n/lectures/lecture10.pdf`)

Beam Search

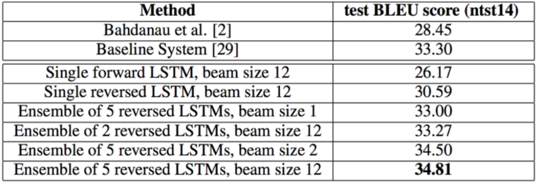
A little better than greedy search, but still greedy

Runtime linear as a function of beam size: trade-off speed/accuracy

In practice: beam sizes $\sim$ 4–12

Some Additional Tricks

From Sutskever et al. (2014):



Method	test BLEU score (ntst14)
Bahdanau et al. [2]	28.45
Baseline System [29]	33.30
Single forward LSTM, beam size 12	26.17
Single reversed LSTM, beam size 12	30.59
Ensemble of 5 reversed LSTMs, beam size 1	33.00
Ensemble of 2 reversed LSTMs, beam size 12	33.27
Ensemble of 5 reversed LSTMs, beam size 2	34.50
Ensemble of 5 reversed LSTMs, beam size 12	**34.81**

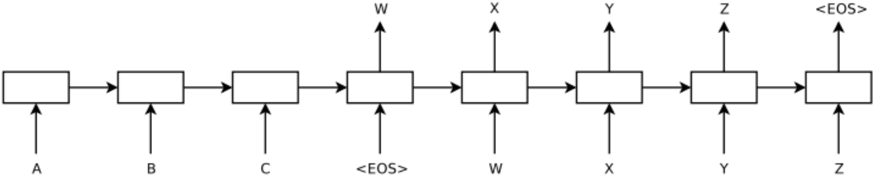
- Deep LSTMs
- Reversing the source sentence

At run time:

- Beam search
- Ensembling: combine N independently trained models and obtaining a “consensus” (always helps!)

End-to-End Neural Machine Translation

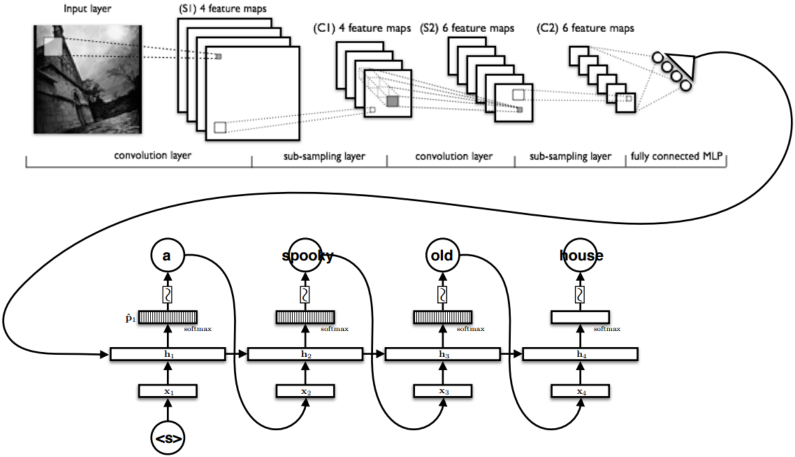
- Previous statistical machine translation models were complicated pipelines (word alignments → phrase table extraction → language model → decoding a phrase lattice)
- As an alternative, can do end-to-end NMT using a simple encoder-decoder
- Doesn't quite work yet, but gets close to top performance



Encode Everything as a Vector

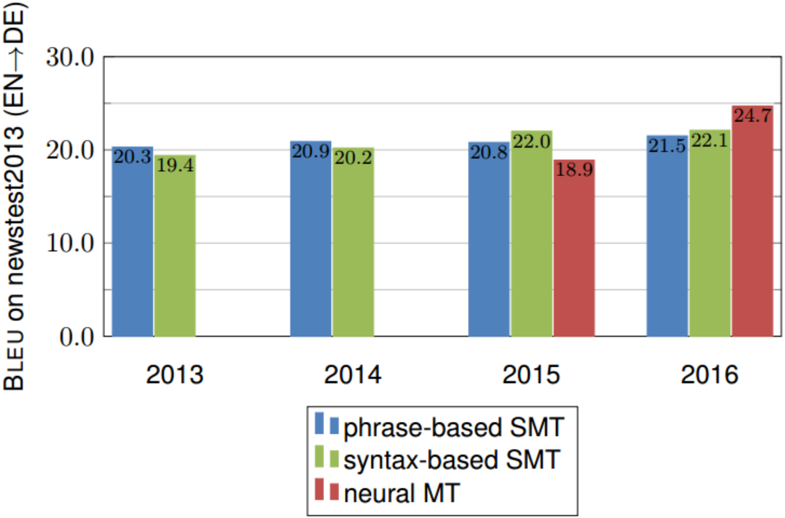
Works for image inputs too!

Caption Generation



(Slide credit: Chris Dyer)

Progress in Machine Translation



Slide credit: Rico Sennrich

NMT: A Success Story

Neural MT went from a fringe research activity in 2014 to the leading standard method in 2016

- 2014: First seq2seq paper published
- 2016: Google Translate switches from SMT to NMT

This is amazing!

SMT systems, built by hundreds of engineers over many years, outperformed by NMT systems trained by a handful of engineers in a few months.

So Is Machine Translation Solved?

No. Many difficulties remain:

- Out-of-vocabulary words
- Domain mismatch between train and test data
- Low-resource language pairs
- Maintaining context over longer text (coming next!)

Limitations

A possible conceptual problem:

- Sentences have unbounded lengths
- Vectors have finite capacity

"You can't cram the meaning of a whole %&$# sentence into a single $&# vector!" (Ray Mooney)

A possible practical problem:

- Distance between "translations" and their sources are distant—can LSTMs learn this?

Outline

1. Statistical Machine Translation

2. Neural Machine Translation

   Encoder-Decoder Architecture

   Encoder-Decoder with Attention

   Convolutional Encoder-Decoder

   Self-Attention and Transformer Networks

3. Transfer Learning with Transformers

4. Conclusions

Encode Sentences as Matrices, Not Vectors

Problem with the fixed-size vector model:

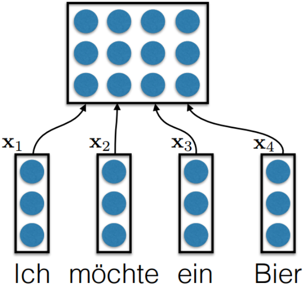
- Sentences are of different sizes but vectors are of the same size
- Bottleneck problem: a single vector needs to represent the full source sentence!

Solution: use matrices instead!

- Fixed number of rows, but number of columns depends on the number of words
- Then, before generating each word in the decoder, use an attention mechanism to condition on the relevant source words only

How to Encode a Sentence as a Matrix?

First shot: define the sentence words' vectors as the columns



(Image credit: Chris Dyer)

- Not very effective, since the word vectors carry no contextual information

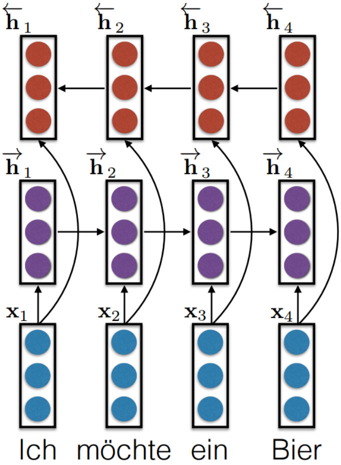
How to Encode a Sentence as a Matrix?

Other strategies:

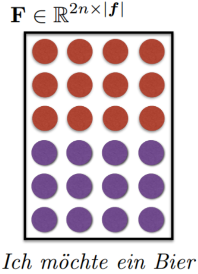
- Convolutional neural networks (Kalchbrenner et al., 2014): can capture context
- **Typical choice:** Bidirectional LSTMs (Bahdanau et al., 2015)
- Later: Transformer networks (Vaswani et al., 2017).

Bidirectional LSTM Encoder

$$\mathbf{f}_i = [\overleftarrow{\mathbf{h}}_i; \overrightarrow{\mathbf{h}}_i]$$



$\overleftarrow{\mathbf{h}}_1$
$\overleftarrow{\mathbf{h}}_2$
$\overleftarrow{\mathbf{h}}_3$
$\overleftarrow{\mathbf{h}}_4$
$\overrightarrow{\mathbf{h}}_1$
$\overrightarrow{\mathbf{h}}_2$
$\overrightarrow{\mathbf{h}}_3$
$\overrightarrow{\mathbf{h}}_4$
$\mathbf{x}_1$
$\mathbf{x}_2$
$\mathbf{x}_3$
$\mathbf{x}_4$
Ich
möchte
ein
Bier



(Slide credit: Chris Dyer)

Generation from Matrices

We now have a matrix $\mathbf{F}$ representing the input. How to generate from it?

Answer: use attention! (Bahdanau et al., 2015)

Attention is the neural counterpart of word alignments.

Generation from Matrices with Attention

Generate the output sentence word by word using an RNN

At each output position t, the RNN receives two inputs:

- a fixed-size vector embedding of the previous output symbol y_{t-1}
- a fixed-size vector encoding a “view” of the input matrix $\mathbf{F}$, via a weighted sum of its columns (i.e., words): $\mathbf{F}\boldsymbol{a}_t$

The weighting of the input columns at each time-step ($\boldsymbol{a}_t$) is called the attention distribution.

Attention Mechanism (Bahdanau et al., 2015)

Let $\boldsymbol{s}_1, \boldsymbol{s}_2, \ldots$ be the states produced by the decoder RNN

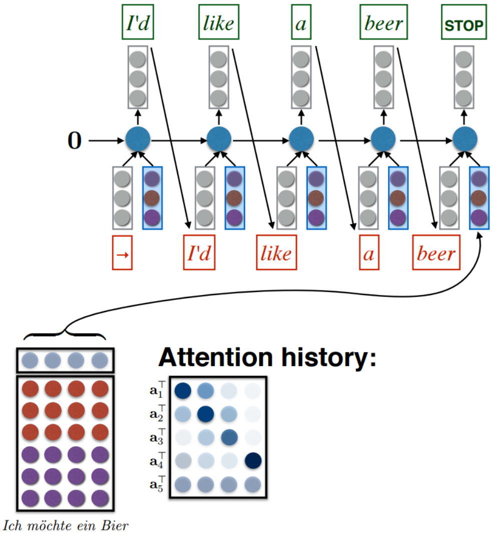
When predicting the tth target word:

1. Compute "similarity" with each of the source words:

   $$z_{t,i} = \boldsymbol{v} \cdot \boldsymbol{g}(\mathbf{W}\boldsymbol{h}_i + \mathbf{U}\boldsymbol{s}_{t-1} + \boldsymbol{b}), \quad \forall i \in [L]$$

   where $\boldsymbol{h}_i$ is the ith column of $\mathbf{F}$ (representation of the ith source word), and $\boldsymbol{v}$, $\mathbf{W}$, $\mathbf{U}$, $\boldsymbol{b}$ are parameters of the model
2. Form vector $\boldsymbol{z}_t = (z_{t,1}, \ldots, z_{t,i}, \ldots, z_{t,L})$ and compute attention $\boldsymbol{a}_t = \textbf{softmax}(\boldsymbol{z}_t)$
3. Use attention to compute input conditioning state $\boldsymbol{c}_t = \mathbf{F}\boldsymbol{a}_t$
4. Update RNN state $\boldsymbol{s_t}$ based on $\boldsymbol{s}_{t-1}, y_{t-1}, \boldsymbol{c}_t$
5. Predict $y_t \sim p(y_t \mid \boldsymbol{s}_t)$

Encoder-Decoder with Attention



(Slide credit: Chris Dyer)

Putting It All Together

```
obtain input matrix $\mathbf{F}$ with a bidirectional LSTM
$t = 0$, $y_0 = \text{START}$ (the start symbol)
$\boldsymbol{s}_0 = \boldsymbol{w}$ (learned initial state)
repeat
    $t = t + 1$
    $\boldsymbol{e}_t = \boldsymbol{v} \cdot \boldsymbol{g}(\mathbf{WF} + \mathbf{U}\boldsymbol{s}_{t-1} + \boldsymbol{b})$
    compute attention $\boldsymbol{a}_t = \mathbf{softmax}(\boldsymbol{e}_t)$
    compute input conditioning state $\boldsymbol{c}_t = \mathbf{F}\boldsymbol{a}_t$
    $\boldsymbol{s}_t = \mathbf{RNN}(\boldsymbol{s}_{t-1}, [\mathbf{E}(y_{t-1}), \boldsymbol{c}_t])$
    $y_t | y_{<t}, \boldsymbol{x} \sim \mathbf{softmax}(\mathbf{P}\boldsymbol{s}_t + \boldsymbol{b})$
until $y_t \neq \text{STOP}$
```

Attention Mechanisms

Attention is closely related to "pooling" operations in convnets (and other architectures)

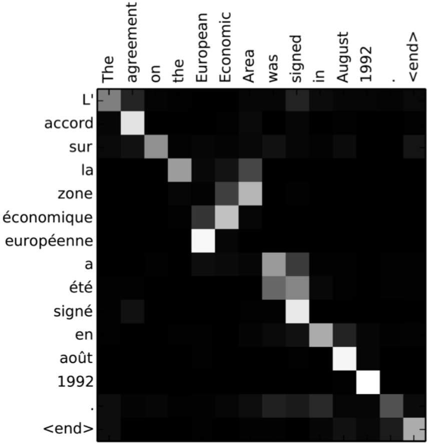
- Attention in MT plays a similar role as alignment, but leads to "soft" alignment instead of "hard" alignment
- Bahdanau et al. (2015)'s model has no bias in favor of diagonals, short jumps, fertility, etc.
- Some recent work adds some "structural" biases (Luong et al., 2015; Cohn et al., 2016)
- Other works constrains the amount of attention each word can receive (based on its fertility): Malaviya et al. (2018).

Attention is Great!

Attention significantly improves NMT performance!

- It's very useful to allow decoder to focus on certain parts of the source
- Attention solves the bottleneck problem (by allowing the decoder to look directly at source)
- Attention helps with vanishing gradient problem (provides shortcut to faraway states)
- Attention provides some interpretability (we can see what the decoder was focusing on)
- This is cool because we never explicitly trained an word aligner; the network learns it by itself!

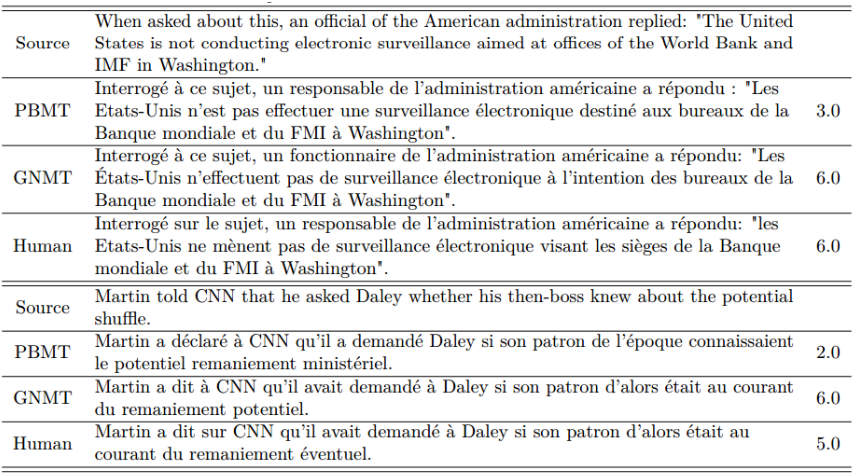
Attention Map



Dzmitry Bahdanau, KyungHuyn Cho, and Yoshua Bengio. ***Neural Machine Translation by Jointly Learning to Translate and Align****. ICLR'15.*

Example: Machine Translation

Some positive examples where NMT has impressive performance:



Source	When asked about this, an official of the American administration replied: "The United States is not conducting electronic surveillance aimed at offices of the World Bank and IMF in Washington."	
PBMT	Interrogé à ce sujet, un responsable de l'administration américaine a répondu : "Les Etats-Unis n'est pas effectuer une surveillance électronique destiné aux bureaux de la Banque mondiale et du FMI à Washington".	3.0
GNMT	Interrogé à ce sujet, un fonctionnaire de l'administration américaine a répondu: "Les États-Unis n'effectuent pas de surveillance électronique à l'intention des bureaux de la Banque mondiale et du FMI à Washington".	6.0
Human	Interrogé sur le sujet, un responsable de l'administration américaine a répondu: "les Etats-Unis ne mènent pas de surveillance électronique visant les sièges de la Banque mondiale et du FMI à Washington".	6.0
Source	Martin told CNN that he asked Daley whether his then-boss knew about the potential shuffle.	
PBMT	Martin a déclaré à CNN qu'il a demandé Daley si son patron de l'époque connaissaient le potentiel remaniement ministériel.	2.0
GNMT	Martin a dit à CNN qu'il avait demandé à Daley si son patron d'alors était au courant du remaniement potentiel.	6.0
Human	Martin a dit sur CNN qu'il avait demandé à Daley si son patron d'alors était au courant du remaniement éventuel.	5.0

(From Wu et al. (2016))

Example: Machine Translation

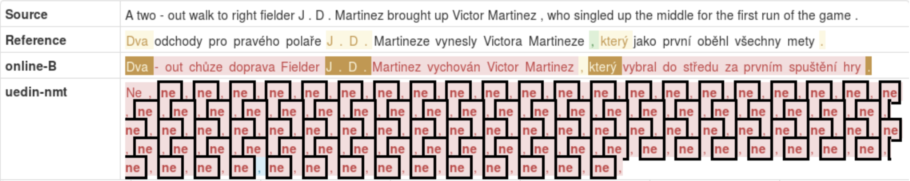
... But also some negative examples:

- Dropping source words (lack of attention)
- Repeated source words (too much attention)

Source:	1922 in Wien geboren, studierte Mang während und nach dem Zweiten Weltkrieg Architektur an der Technischen Hochschule in Wien bei Friedrich Lehmann.
Human:	Born in Vienna in 1922, Meng studied architecture at the Technical University in Vienna under Friedrich Lehmann *during and after the second World War.*
NMT:	*Born in Vienna in 1922, Mang studied architecture at the Technical College in Vienna with Friedrich Lehmann.
Source:	Es ist schon komisch, wie dies immer wieder zu dieser Jahreszeit auftaucht.
Human:	It's funny how this always comes up at *this time* of year.
NMT:	*It's funny how **this time** to come back to **this time** of year.

Example: Machine Translation

... And an example where neural MT failed miserably:

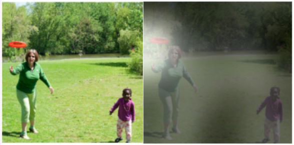


Source	A two - out walk to right fielder J . D . Martinez brought up Victor Martinez , who singled up the middle for the first run of the game .
Reference	Dva odchody pro pravého polaře J . D . Martineze vynesly Victora Martineze , který jako první oběhl všechny mety .
online-B	Dva - out chůze doprava Fielder J . D . Martinez vychován Victor Martinez , který vybral do středu za prvním spuštění hry .
uedin-nmt	Ne , ne , ne , ne , ne , ne , ne , ne , ne , ne , ne , ne , ne , ne , ne , ne , ne , ne , ne , ne , ne , ne , ne , ne , ne , ne , ne , ne , ne , ne , ne , ne , ne , ne , ne , ne , ne , ne , ne , ne , ne , ne , ne , ne , ne , ne , ne , ne , ne , ne , ne , ne , ne , ne , ne , ne , ne , ne , ne , ne , ne , ne , ne , ne , ne , ne , ne , ne , ne , ne , ne , ne , ne , ne , ne , ne , ne , ne , ne , ne , ne , ne , ne , ne , ne , ne , ne , ne , ne , ne , ne , ne , ne , ne , ne , ne , ne , ne , ne , ne ,

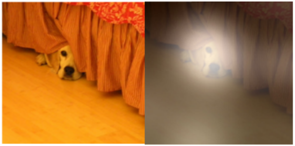
(Credit: Barry Haddow)

Example: Caption Generation

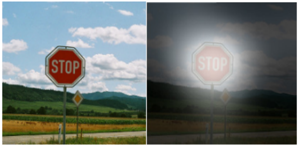
Attention over images:



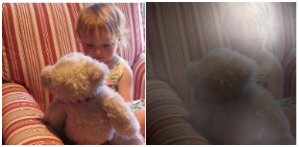
A woman is throwing a frisbee in a park.



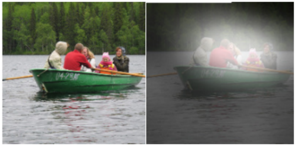
A dog is standing on a hardwood floor.



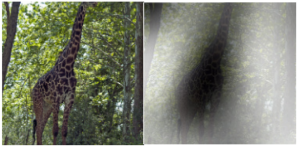
A stop sign is on a road with a mountain in the background.



A little girl sitting on a bed with a teddy bear.



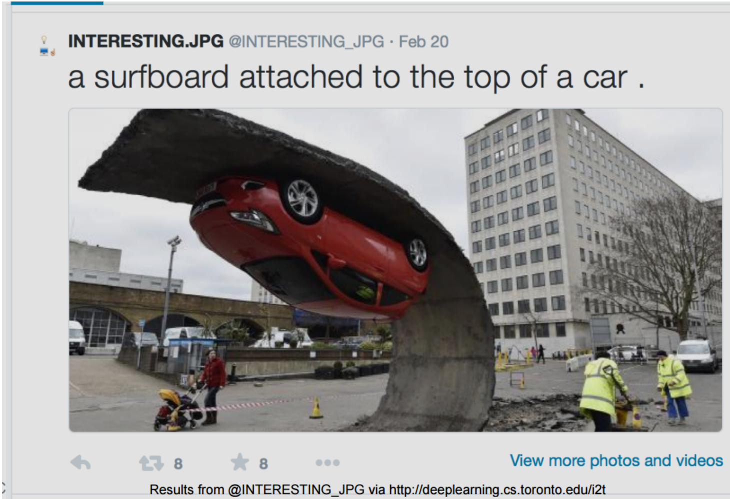
A group of people sitting on a boat in the water.



A giraffe standing in a forest with trees in the background.

(Slide credit to Yoshua Bengio)

A More Extreme Example...



INTERESTING.JPG @INTERESTING_JPG · Feb 20

a surfboard attached to the top of a car .

Results from @INTERESTING_JPG via http://deeplearning.cs.toronto.edu/i2t

(Slide credit to Dhruv Batra)

Attention and Memories

Attention is used in other problems, sometimes under different names:

- image caption generation (Xu et al., 2015)
- speech recognition (Chorowski et al., 2015)
- memory networks for reading comprehension (Sukhbaatar et al., 2015; Hermann et al., 2015)
- neural Turing machines and other "differentiable computers" (Graves et al., 2014; Grefenstette et al., 2015)

Other Attentions

Can we have more interpretable attention? Closer to hard alignments?

Can we upper bound how much attention a word receives? This may prevent a common problem in neural MT, repetitions

We'll see:

- Sparse attention via **sparsemax** (Martins and Astudillo, 2016)
- Constrained attention with constrained softmax/sparsemax (Malaviya et al., 2018)

Recap: Sparsemax (Martins and Astudillo, 2016)

A sparse-friendly alternative to softmax is **sparsemax** : $\mathbb{R}^C \to \Delta^{C-1}$:

$$\boxed{\textbf{sparsemax}(\boldsymbol{z}) := \arg\min_{\boldsymbol{p} \in \Delta^{C-1}} \|\boldsymbol{p} - \boldsymbol{z}\|^2.}$$

- In words: Euclidean projection of $\boldsymbol{z}$ onto the probability simplex
- Likely to hit the boundary of the simplex, in which case **sparsemax**($\boldsymbol{z}$) becomes sparse (hence the name)
- Retains many of the properties of softmax (e.g. differentiability), having in addition the ability of producing sparse distributions
- Projecting onto the simplex amounts to a soft-thresholding operation (next)
- Efficient forward/backward propagation.

Sparsemax in Closed Form

- Projecting onto the simplex amounts to a soft-thresholding operation:

$$\textbf{sparsemax}_i(\boldsymbol{z}) \;=\; \max\{0, z_i - \tau\}$$

  where τ is a normalizing constant such that $\sum_j \max\{0, z_j - \tau\} = 1$
- To evaluate the sparsemax, all we need is to compute τ
- Coordinates above the threshold will be shifted by this amount; the others will be truncated to zero.
- **This will result in a sparse probability vector!**

A Formal Algorithm

```
**Input:** $\boldsymbol{z} \in \mathbb{R}^C$
Sort $\boldsymbol{z}$ as $z_{(1)} \geq \ldots \geq z_{(C)}$
Find $k(\boldsymbol{z}) := \max\left\{k \in [C] \mid 1 + kz_{(k)} > \sum_{j \leq k} z_{(j)}\right\}$
Define $\tau(\boldsymbol{z}) = \frac{\left(\sum_{j \leq k(\boldsymbol{z})} z_{(j)}\right) - 1}{k(\boldsymbol{z})}$
**Output:** $\boldsymbol{p} \in \Delta^{C-1}$ s.t. $p_i = [z_i - \tau(\boldsymbol{z})]_+$.
```

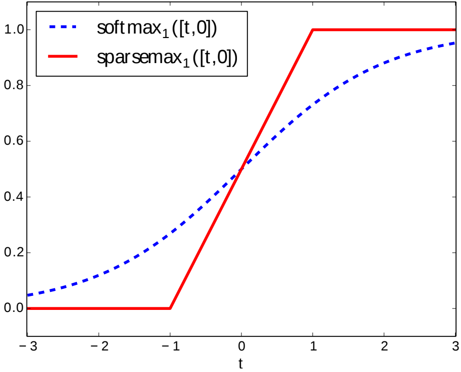
- Time complexity is $O(C \log C)$ due to the sort operation; but $O(C)$ algorithms exist based on linear-time selection.
- Note: evaluating **softmax** costs $\mathrm{O}(C)$ too.

Two Dimensions

- Parametrize $z = (t, 0)$
- The 2D **softmax** is the logistic (sigmoid) function:

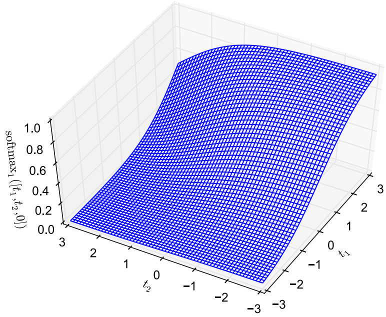
$$\textbf{softmax}_1(z) = (1 + \exp(-t))^{-1}$$

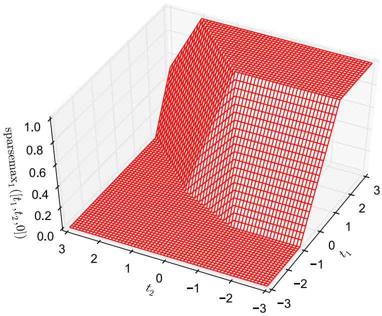
- The 2D **sparsemax** is the "hard" version of the sigmoid:



Three Dimensions

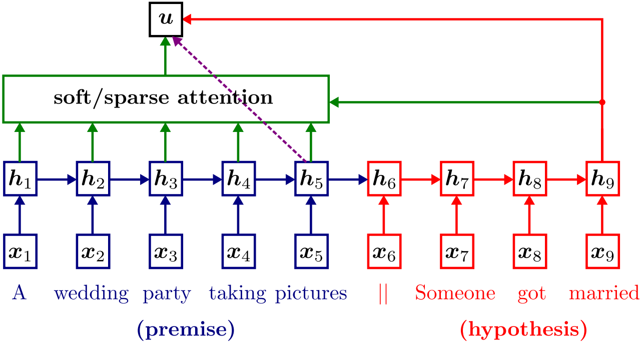
- Parameterize $\boldsymbol{z} = (t_1, t_2, 0)$ and plot $\textbf{softmax}_1(\boldsymbol{z})$ and $\textbf{sparsemax}_1(\boldsymbol{z})$ as a function of t_1 and t_2
- **sparsemax** is piecewise linear, but asymptotically similar to **softmax**





Example: Sparse Attention for Natural Language Inference

- SNLI corpus (Bowman et al., 2015): 570K sentence pairs (a premise and an hypothesis), labeled as **entailment**, **contradiction**, or **neutral**
- We used an attention-based architecture as Rocktäschel et al. (2015)


u
soft/sparse attention
h_1 h_2 h_3 h_4 h_5 h_6 h_7 h_8 h_9
x_1 x_2 x_3 x_4 x_5 x_6 x_7 x_8 x_9
A wedding party taking pictures || Someone got married
(premise)
(hypothesis)

Example: Sparse Attention for Natural Language Inference

- *In blue*, the premise words selected by the sparse attention mechanism
- In red, the hypothesis
- Only a few words are selected, which are key for the system's decision
- The sparsemax activation yields a compact and more interpretable selection, which can be particularly useful in long sentences

A boy *rides on* a *camel* in a crowded area while talking on his cellphone.
—— A boy is riding an animal. [entailment]

A young girl wearing *a pink coat* plays with a *yellow* toy golf club.
—— A girl is wearing a blue jacket. [contradiction]

Two black dogs are *frolicking* around the *grass together*.
—— Two dogs swim in the lake. [contradiction]

A man wearing a yellow striped shirt *laughs* while *seated next* to another *man* who is wearing a light blue shirt and *clasping* his hands together.
—— Two mimes sit in complete silence. [contradiction]

Constrained Softmax

Constrained softmax resembles softmax, but it allows imposing hard constraints on the maximal probability assigned to each word

- Given scores $\boldsymbol{z} \in \mathbb{R}^C$ and **upper bounds** $\boldsymbol{u} \in \mathbb{R}^C$:

$$\begin{aligned} \textbf{csoftmax}(\boldsymbol{z};\boldsymbol{u}) \quad = \quad & \operatorname*{arg\,min}_{\boldsymbol{p}\in\Delta^{C-1}} \textbf{KL}(\boldsymbol{p} \,\|\, \textbf{softmax}(\boldsymbol{z})) \\ & \quad\quad \text{s.t.} \quad \boldsymbol{p} \le \boldsymbol{u} \end{aligned}$$

- Related to **posterior regularization** (Ganchev et al., 2010)

Particular cases:

- If $\boldsymbol{u} \geq \mathbf{1}$, all constraints are loose and this reduces to softmax
- If $\boldsymbol{u} \in \Delta^{C-1}$, they are tight and we must have $\boldsymbol{p} = \boldsymbol{u}$

How to Evaluate?

Forward computation takes $O(C \log C)$ **time** (Martins and Kreutzer, 2017):

- Let $\mathcal{A} = \{i \in [C] \mid p_i^\star < u_i\}$ be the **constraints that are met strictly**
- Then by writing the KKT conditions we can express the solution as:

$$p_i^\star = \min\left\{\frac{\exp(z_i)}{Z}, u_i\right\} \quad \forall i \in [C], \quad \text{where } Z = \frac{\sum_{i \in \mathcal{A}} \exp(z_i)}{1 - \sum_{i \notin \mathcal{A}} u_i}.$$

- Identifying the set $\mathcal{A}$ can be done in $O(C \log C)$ time with a sort

How to Backpropagate?

We need to compute gradients with respect to both $\boldsymbol{z}$ and $\boldsymbol{u}$

Can be done in $O(C)$ time (Martins and Kreutzer, 2017):

- Let $L(\boldsymbol{\theta})$ be a loss function, $\mathrm{d}\boldsymbol{p} = \nabla_{\boldsymbol{\alpha}} L(\boldsymbol{\theta})$ be the output gradient, and $\mathrm{d}\boldsymbol{z} = \nabla_{\boldsymbol{z}} L(\boldsymbol{\theta})$ and $\mathrm{d}\boldsymbol{u} = \nabla_{\boldsymbol{u}} L(\boldsymbol{\theta})$ be the input gradients
- Then, the input gradients are given as:

$$\begin{aligned} \mathrm{d}z_i &= \mathbb{I}(i \in \mathcal{A}) p_i (\mathrm{d}p_i - m) \\ \mathrm{d}u_i &= \mathbb{I}(i \notin \mathcal{A}) (\mathrm{d}p_i - m), \end{aligned}$$

where $m = (\sum_{i \in \mathcal{A}} p_i \, \mathrm{d}p_i)/(1 - \sum_{i \notin \mathcal{A}} u_i)$.

Constrained Sparsemax (Malaviya et al., 2018)

Similar idea, but replacing softmax by sparsemax:

- Given scores $\boldsymbol{z} \in \mathbb{R}^C$ and **upper bounds** $\boldsymbol{u} \in \mathbb{R}^C$:

$$
\begin{aligned}
\textbf{csparsemax}(\boldsymbol{z};\boldsymbol{u}) \quad = \quad & \operatorname*{arg\,min}_{\boldsymbol{p}\in\Delta^{C-1}} \|\boldsymbol{p}-\boldsymbol{z}\|^2 \\
& \qquad \text{s.t.} \quad \boldsymbol{p} \le \boldsymbol{u}
\end{aligned}
$$

- Both sparse and upper bounded
- If $\boldsymbol{u} \ge \mathbf{1}$, all constraints are loose and this reduces to sparsemax
- If $\boldsymbol{u} \in \Delta^{C-1}$, they are tight and we must have $\boldsymbol{p} = \boldsymbol{u}$

How to Evaluate?

Forward computation can be done with a sort in $O(C \log C)$ time

Can be reduced to $O(C)$ (Malaviya et al., 2018; Pardalos and Kovoor, 1990):

- Let $\mathcal{A} = \{i \in [C] \mid 0 < p_i^\star < u_i\}$ be the **constraints that are met strictly**
- Let $\mathcal{A}_R = \{i \in [C] \mid p_i^\star = u_i\}$
- Then by writing the KKT conditions we can express the solution as:

$$p_i^\star = \max\{0, \min\{u_i, z_i - \tau\}\} \quad \forall i \in [C], \quad \text{where } \tau \text{ is a constant.}$$

- Identifying the sets $\mathcal{A}$ and $\mathcal{A}_R$ can be done in $O(C \log C)$ time with a sort

How to Backpropagate?

We need to compute gradients with respect to both $\boldsymbol{z}$ and $\boldsymbol{u}$

Can be done in sublinear time $O(|\mathcal{A}| + |\mathcal{A}_R|)$ (Malaviya et al., 2018):

- Let $L(\boldsymbol{\theta})$ be a loss function, $\mathrm{d}\boldsymbol{p} = \nabla_{\boldsymbol{\alpha}} L(\boldsymbol{\theta})$ be the output gradient, and $\mathrm{d}\boldsymbol{z} = \nabla_{\boldsymbol{z}} L(\boldsymbol{\theta})$ and $\mathrm{d}\boldsymbol{u} = \nabla_{\boldsymbol{u}} L(\boldsymbol{\theta})$ be the input gradients
- Then, the input gradients are given as:

$$\begin{aligned} \mathrm{d}z_i &= \mathbb{I}(i \in \mathcal{A})(\mathrm{d}p_i - m) \\ \mathrm{d}u_i &= \mathbb{I}(i \in \mathcal{A}_R)(\mathrm{d}p_i - m), \end{aligned}$$

where $m = \frac{1}{|\mathcal{A}|} \sum_{i \in \mathcal{A}} \mathrm{d}p_i$.

Next, we show how to use these constrained attentions in neural machine translation decoders, inspired by the idea of **fertility** (IBM Model 2)...

Modeling Fertility in NMT

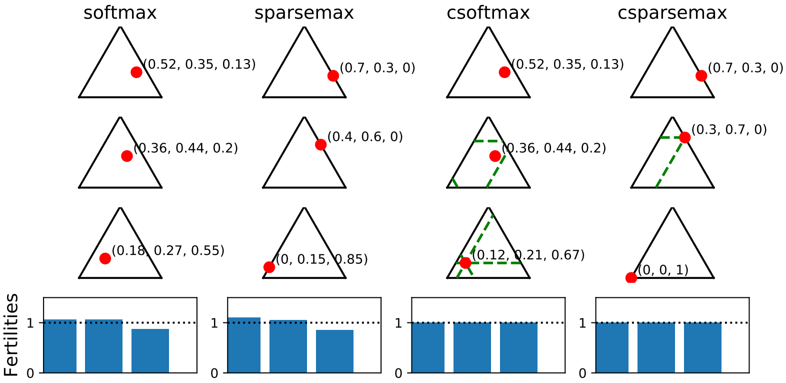
We do the following procedure:

1. Align the training data with `fast_align`
2. Train a separate BILSTM to predict fertility f_i for each word
3. At each decoder step, use upper bound $u_i = f_i - \beta_i$ for each word, where β_i is the cumulative attention

See Malaviya et al. (2018) for more details.

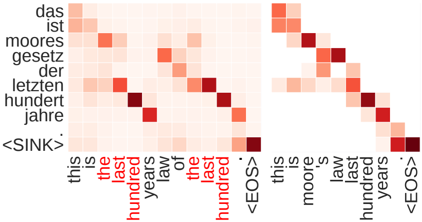
Example: Source Sentence with Three Words

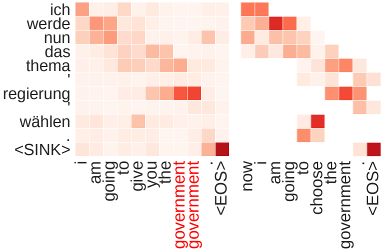
Assume each word is given fertility 1:



Attention Maps

Softmax (left) vs Constrained Sparsemax (right) for De-En:





Sentence Examples

input	so ungefähr , sie wissen schon .
reference	*like that , you know* .
softmax	**so , you know , you know** .
sparsemax	**so , you know , you know** .
csoftmax	**so , you know , you know** .
csparsemax	like that , you know .

input	und wir benutzen dieses wort mit solcher verachtung .
reference	and we say that word *with such contempt* .
softmax	and we use this word with such **contempt contempt** .
sparsemax	and we use this word with such contempt .
csoftmax	and we use this word with **like this** .
csparsemax	and we use this word with such contempt .

Outline

1. Statistical Machine Translation

2. **Neural Machine Translation**
   - Encoder-Decoder Architecture
   - Encoder-Decoder with Attention
   - Convolutional Encoder-Decoder
   - Self-Attention and Transformer Networks

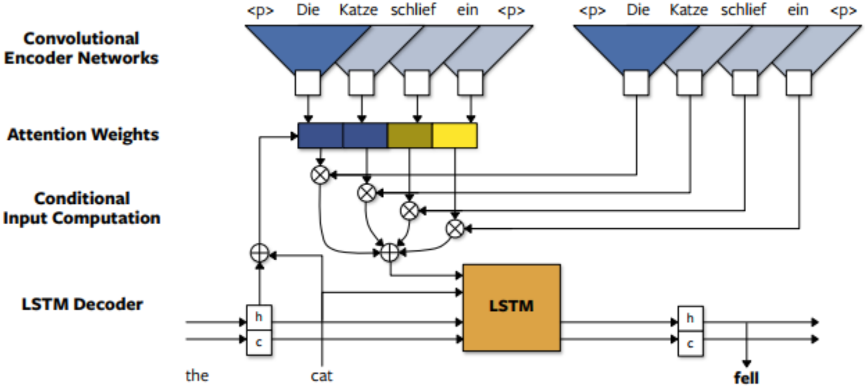
3. Transfer Learning with Transformers

4. Conclusions

Disadvantages of the RNN Architecture

- Sequential computation prevents parallelization
- Long-range dependencies between words that are far apart involve too many computation steps (information will be dropped, even with GRUs or LSTMs)
- Solution: replace the RNN encoder by a hierarchical CNN!

Convolutional Encoder

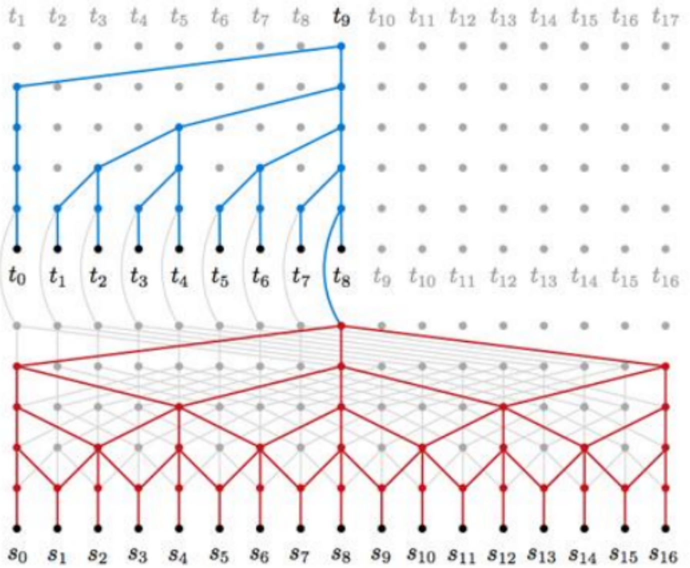


(Gehring et al., 2017)

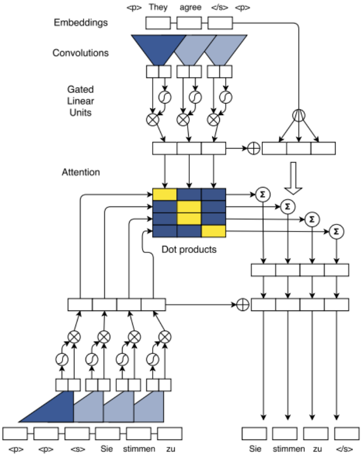
Fully Convolutional

- Can have a CNN decoder too!
- Convolutions will be over output prefixes only
- Encoder is parallelizable, but decoder still requires sequential computation (the model is still auto-regressive)

Convolutional Sequence-to-Sequence


t_1 t_2 t_3 t_4 t_5 t_6 t_7 t_8 t_9 t_{10} t_{11} t_{12} t_{13} t_{14} t_{15} t_{16} t_{17}
t_0 t_1 t_2 t_3 t_4 t_5 t_6 t_7 t_8 t_9 t_{10} t_{11} t_{12} t_{13} t_{14} t_{15} t_{16}
s_0 s_1 s_2 s_3 s_4 s_5 s_6 s_7 s_8 s_9 s_{10} s_{11} s_{12} s_{13} s_{14} s_{15} s_{16}

Convolutional Sequence-to-Sequence



(Gehring et al., 2017)

Outline

1. Statistical Machine Translation

2. Neural Machine Translation

    Encoder-Decoder Architecture

    Encoder-Decoder with Attention

    Convolutional Encoder-Decoder

    Self-Attention and Transformer Networks

3. Transfer Learning with Transformers

4. Conclusions

Self-Attention

- Both RNN and CNN decoders require an attention mechanism
- Attention allows focusing on an arbitrary position in the source sentence, shortcutting the computation graph
- But if attention gives us access to any state... maybe we don't need the RNN?

Transformer (Vaswani et al., 2017)

- **Key idea:** instead of RNN/CNNs, use self-attention in the encoder
- Each word state attends to all the other words
- Each self-attention is followed by a feed-forward transformation
- Do several layers of this
- Do the same for the decoder, attending only to already generated words.

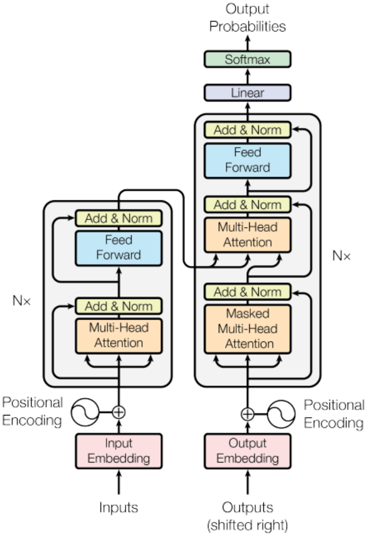


Figure 1: The Transformer - model architecture.

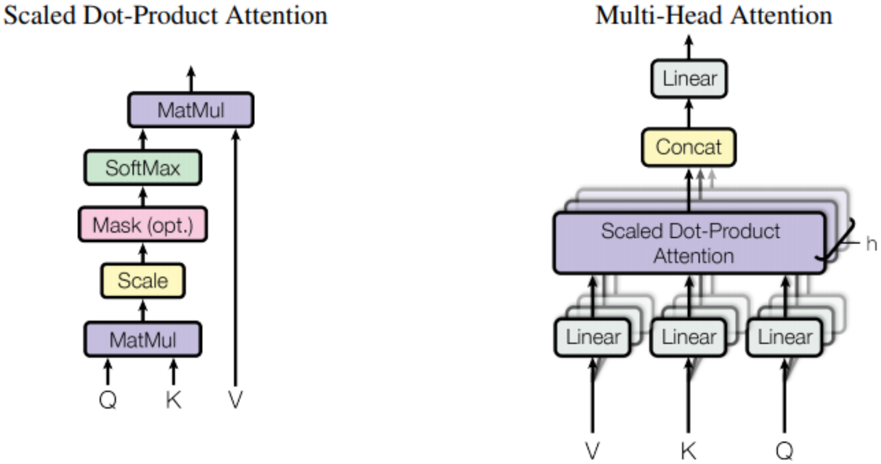
Transformer Basics

Let's define the basic building blocks of transformer networks first: new attention layers!

Two innovations:

- scaled dot-product attention
- multi-head attention

Scaled Dot-Product and Multi-Head Attention



(Vaswani et al., 2017)

Scaled Dot-Product Attention

Inputs:

- A query vector $\boldsymbol{q}$ (e.g. the decoder state)
- A matrix $\mathbf{K}$ whose columns are key vectors (e.g. the encoder states)
- A matrix $\mathbf{V}$ whose columns are value vectors (e.g. the encoder states)

When discussing attention with RNNs, we assume the key and value vectors were the same, but they don't need to!

Output: the weighted sum of values, where each weight is computed by a dot product between the query and the corresponding key:

$$\boldsymbol{a} = \mathbf{softmax}(\mathbf{K}\boldsymbol{q}), \qquad \bar{\boldsymbol{v}} = \mathbf{V}\boldsymbol{a}.$$

With multiple queries,

$$\bar{\mathbf{V}} = \mathbf{softmax}(\mathbf{Q}\mathbf{K}^\top)\mathbf{V}, \qquad \mathbf{Q} \in \mathbb{R}^{|Q| \times d_k}, \mathbf{K} \in \mathbb{R}^{|K| \times d_k}, \mathbf{V} \in \mathbb{R}^{|K| \times d_v}.$$

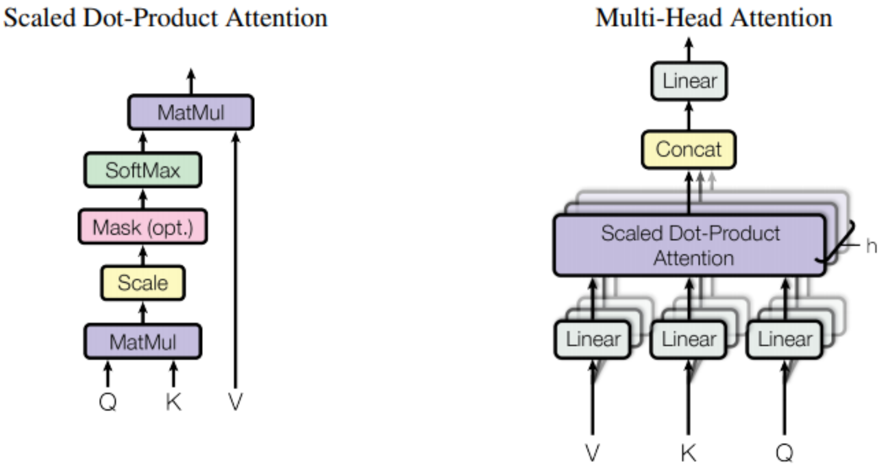
Scaled Dot-Product Attention

Problem: As d_k gets large, the variance of $\boldsymbol{q}^\top \boldsymbol{k}$ increases, the softmax gets very peaked, hence its gradient gets smaller.

Solution: scale by length of query/key vectors:

$$\bar{\mathbf{V}} = \mathbf{softmax}\left(\frac{\mathbf{QK}^\top}{\sqrt{d_k}}\right)\mathbf{V}.$$

Scaled Dot-Product and Multi-Head Attention



Scaled Dot-Product Attention

Multi-Head Attention

(Vaswani et al., 2017)

Multi-Head Attention

Self-attention lets each word state form a query vector and attend to the other words' key vectors

This is vaguely similar to a 1D convolution, but where the filter weights are "dynamic" is the window size spans the entire sentence!

Problem: only one channel for words to interact with one-another

Solution: multi-head attention!

- first project $\mathbf{Q}$, $\mathbf{K}$, and $\mathbf{V}$ into lower dimensional spaces
- then apply attention in multiple channels, concatenate the outputs and pipe through linear layer:

$$\mathrm{MultiHead}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \mathrm{Concat}(\mathrm{head}_1, \ldots, \mathrm{head}_h)\mathbf{W}^O,$$

where $\mathrm{head}_i = \mathrm{Attention}(\mathbf{Q}\mathbf{W}_i^Q, \mathbf{K}\mathbf{W}_i^K, \mathbf{V}\mathbf{W}_i^V)$.

Other Tricks

- Self-attention blocks are repeated 6 times
- Residual connections on each attention block
- Positional encodings (to distinguish word positions)
- Layer normalization

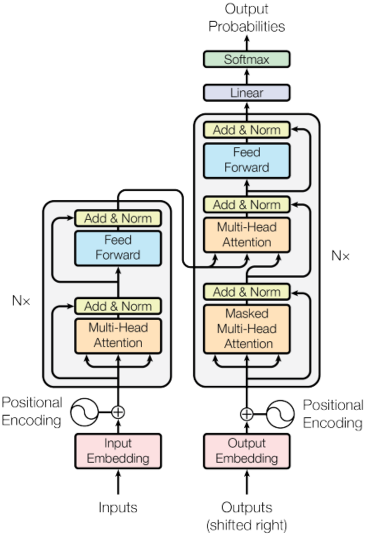
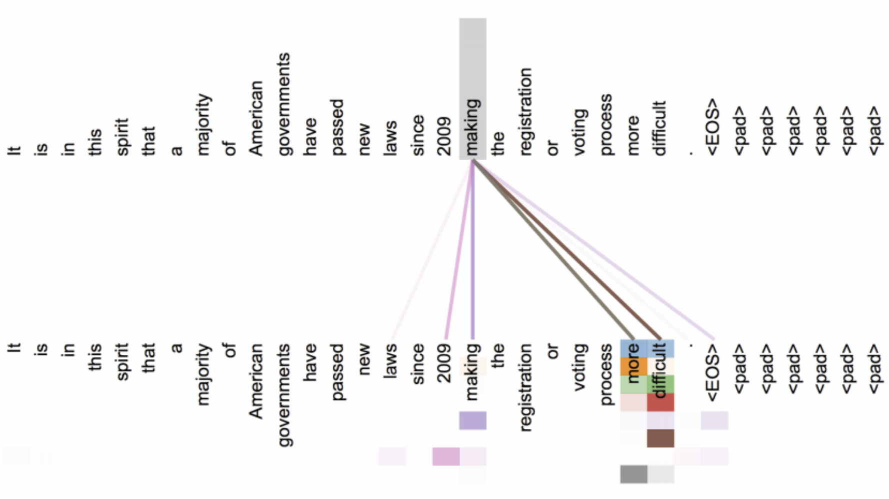
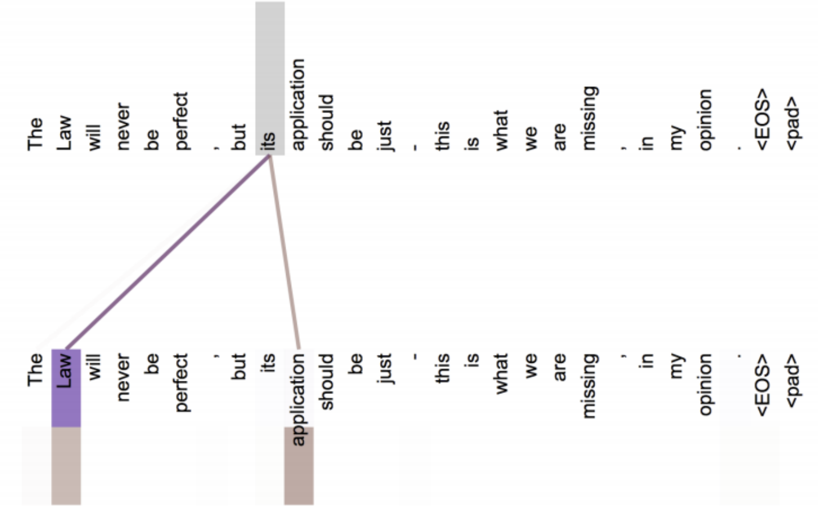


Figure 1: The Transformer - model architecture.

Attention Visualization Layer 5



Implicit Anaphora Resolution



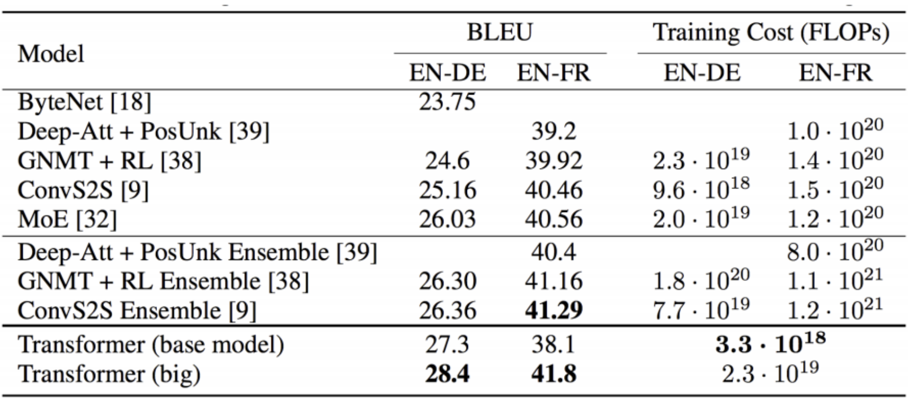
More Transformer Tricks

- Subword units
- Checkpoint averaging
- ADAM optimizer with non-standard learning rate schedules
- Label smoothing
- Auto-regressive decoding with beam search and length penalties

Overall, transformers are harder to optimize than RNN sequence-to-sequence models

They don't work out of the box: hyperparameter tuning is very important.

Transformer Results



Model	BLEU		Training Cost (FLOPs)	
	EN-DE	EN-FR	EN-DE	EN-FR
ByteNet [18]	23.75			
Deep-Att + PosUnk [39]		39.2		$1.0 \cdot 10^{20}$
GNMT + RL [38]	24.6	39.92	$2.3 \cdot 10^{19}$	$1.4 \cdot 10^{20}$
ConvS2S [9]	25.16	40.46	$9.6 \cdot 10^{18}$	$1.5 \cdot 10^{20}$
MoE [32]	26.03	40.56	$2.0 \cdot 10^{19}$	$1.2 \cdot 10^{20}$
Deep-Att + PosUnk Ensemble [39]		40.4		$8.0 \cdot 10^{20}$
GNMT + RL Ensemble [38]	26.30	41.16	$1.8 \cdot 10^{20}$	$1.1 \cdot 10^{21}$
ConvS2S Ensemble [9]	26.36	**41.29**	$7.7 \cdot 10^{19}$	$1.2 \cdot 10^{21}$
Transformer (base model)	27.3	38.1	$\mathbf{3.3 \cdot 10^{18}}$	
Transformer (big)	**28.4**	**41.8**	$2.3 \cdot 10^{19}$	

Outline

1. Statistical Machine Translation

2. Neural Machine Translation

   Encoder-Decoder Architecture

   Encoder-Decoder with Attention

   Convolutional Encoder-Decoder

   Self-Attention and Transformer Networks

3. **Transfer Learning with Transformers**

4. Conclusions

GPT-2

Transformers can be used for language generation.

Example: OpenAI's GPT-2:

- A transformer-based language model with 1.5 billion parameters, trained on 8 million web pages
- Training objective: predict the next word, given all of the previous words
- Test time: top-k truncated random sampling
- Lots of hype!

GPT-2 Generated Text

A machine learning school was held in Bangkok early August.

`https://talktotransformer.com/`

GPT-2 Generated Text

A machine learning school was held in Bangkok early August. I had originally expected a much more "procedural and algorithmic" approach as well, but was ple asantly surprised by the level of expertise of the students. These were both very experienced mathematicians, but also had an appreciation for machine learning techniques. These ar e the same school that created and operates the very popular Algorithmia blog , so they aren't exactly new to me, and are well known to other Algorithmia grads at the University o f Central Europe.

https://talktotransformer.com/

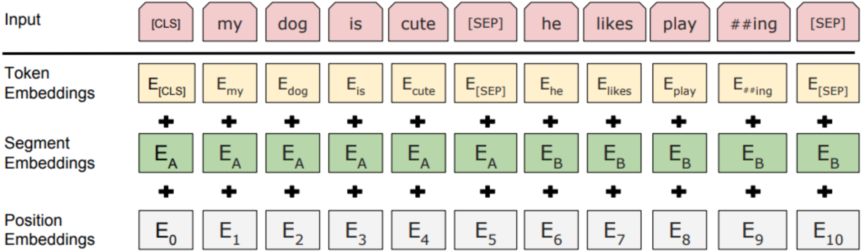
BERT

Transformers can also be used to pre-train big models and fine-tune them for specific tasks.

Example: BERT (Devlin et al., 2018)

- Randomly mask some words of the input and train a Transformer to recover those words from the context
- Both left and right context, used simultaneously!
- In doing so, learn contextual word representations (just like ELMo)
- Can use this as a pre-trained model and fine-tune it to any downstream task
- Extremely effective! Achieved SOTA on 11 NLP tasks (7.7% absolute point improvement on GLUE score).

BERT



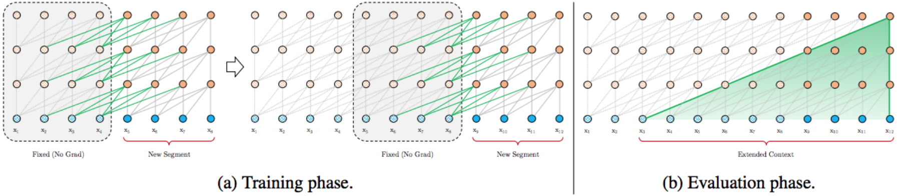
Input	[CLS]	my	dog	is	cute	[SEP]	he	likes	play	##ing	[SEP]
Token Embeddings	$E_{[CLS]}$	E_{my}	E_{dog}	E_{is}	E_{cute}	$E_{[SEP]}$	E_{he}	E_{likes}	E_{play}	$E_{\#\#ing}$	$E_{[SEP]}$
Segment Embeddings	E_A	E_A	E_A	E_A	E_A	E_A	E_B	E_B	E_B	E_B	E_B
Position Embeddings	E_0	E_1	E_2	E_3	E_4	E_5	E_6	E_7	E_8	E_9	E_{10}

(Devlin et al., 2018)

TransformerXL and XLNet

Big transformers can look at larger contexts.

TransformerXL: enables going beyond a fixed length without disrupting temporal coherence:



x_1 x_2 x_3 x_4 x_5 x_6 x_7 x_8
Fixed (No Grad)
New Segment
x_1 x_2 x_3 x_4 x_5 x_6 x_7 x_8 x_9 x_{10} x_{11} x_{12}
Fixed (No Grad)
New Segment
x_1 x_2 x_3 x_4 x_5 x_6 x_7 x_8 x_9 x_{10} x_{11} x_{12}
Extended Context

(a) Training phase.

(b) Evaluation phase.

(Dai et al., 2019)

XLNet: uses TransformerXL for transfer learning, replacing BERT's masking by sampling different generation orderings (Yang et al., 2019)

Outline

1. Statistical Machine Translation

2. Neural Machine Translation

   Encoder-Decoder Architecture

   Encoder-Decoder with Attention

   Convolutional Encoder-Decoder

   Self-Attention and Transformer Networks

3. Transfer Learning with Transformers

4. **Conclusions**

Conclusions

- Machine translation is a key problem in AI since the 1950s
- Neural machine translation with sequence-to-sequence models was a breakthrough
- Representing a full sentence with a single vector is a bottleneck
- Attention mechanisms allow focusing on different parts of the input and solve the bottleneck problem
- Encoders/decoders can be RNNs, CNNs, or self-attention layers
- Transformer networks are the current state of the art in this task
- Other applications beyond MT: speech recognition, image captioning, etc.
- Code available and more info: `https://github.com/tensorflow/tensor2tensor`.

Thank you!

Questions?

References I

Bahdanau, D., Cho, K., and Bengio, Y. (2015). Neural Machine Translation by Jointly Learning to Align and Translate. In *International Conference on Learning Representations*.

Bowman, S. R., Angeli, G., Potts, C., and Manning, C. D. (2015). A Large Annotated Corpus for Learning Natural Language Inference. In *Proc. of Empirical Methods in Natural Language Processing*.

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning Phrase Representations Using RNN Encoder-Decoder for Statistical Machine Translation. In *Proc. of Empirical Methods in Natural Language Processing*.

Chorowski, J. K., Bahdanau, D., Serdyuk, D., Cho, K., and Bengio, Y. (2015). Attention-based Models for Speech Recognition. In *Advances in Neural Information Processing Systems*, pages 577–585.

Cohn, T., Hoang, C. D. V., Vymolova, E., Yao, K., Dyer, C., and Haffari, G. (2016). Incorporating structural alignment biases into an attentional neural translation model. *arXiv preprint arXiv:1601.01085*.

Dai, Z., Yang, Z., Yang, Y., Cohen, W. W., Carbonell, J., Le, Q. V., and Salakhutdinov, R. (2019). Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Ganchev, K., Graca, J., Gillenwater, J., and Taskar, B. (2010). Posterior regularization for structured latent variable models. *Journal of Machine Learning Research*, 11:2001–2049.

Gehring, J., Auli, M., Grangier, D., Yarats, D., and Dauphin, Y. N. (2017). Convolutional sequence to sequence learning. *arXiv preprint arXiv:1705.03122*.

Graves, A., Wayne, G., and Danihelka, I. (2014). Neural Turing Machines. *arXiv preprint arXiv:1410.5401*.

Grefenstette, E., Hermann, K. M., Suleyman, M., and Blunsom, P. (2015). Learning to Transduce with Unbounded Memory. In *Advances in Neural Information Processing Systems*, pages 1819–1827.

Hermann, K. M., Kocisky, T., Grefenstette, E., Espeholt, L., Kay, W., Suleyman, M., and Blunsom, P. (2015). Teaching Machines to Read and Comprehend. In *Advances in Neural Information Processing Systems*, pages 1684–1692.

References II

Kalchbrenner, N., Grefenstette, E., and Blunsom, P. (2014). A convolutional neural network for modelling sentences. *arXiv preprint arXiv:1404.2188*.

Luong, M.-T., Pham, H., and Manning, C. D. (2015). Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*.

Malaviya, C., Ferreira, P., and Martins, A. F. T. (2018). Sparse and constrained attention for neural machine translation. In *Proc. of the Annual Meeting of the Association for Computational Linguistics*.

Martins, A. F. T. and Astudillo, R. (2016). From Softmax to Sparsemax: A Sparse Model of Attention and Multi-Label Classification. In *Proc. of the International Conference on Machine Learning*.

Martins, A. F. T. and Kreutzer, J. (2017). Fully differentiable neural easy-first taggers. In *Proc. of Empirical Methods for Natural Language Processing*.

Pardalos, P. M. and Kovoor, N. (1990). An algorithm for a singly constrained class of quadratic programs subject to upper and lower bounds. *Mathematical Programming*, 46(1):321–328.

Rocktäschel, T., Grefenstette, E., Hermann, K. M., Kočiskỳ, T., and Blunsom, P. (2015). Reasoning about Entailment with Neural Attention. *arXiv preprint arXiv:1509.06664*.

Shannon, C. E. and Weaver, W. (1949). The mathematical theory of communication. *Urbana: University of Illinois Press*, 29.

Sukhbaatar, S., Szlam, A., Weston, J., and Fergus, R. (2015). End-to-End Memory Networks. In *Advances in Neural Information Processing Systems*, pages 2431–2439.

Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, pages 3104–3112.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. (2017). Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., et al. (2016). Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Xu, K., Ba, J., Kiros, R., Courville, A., Salakhutdinov, R., Zemel, R., and Bengio, Y. (2015). Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. In *International Conference on Machine Learning*.

Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R., and Le, Q. V. (2019). Xlnet: Generalized autoregressive pretraining for language understanding. *arXiv preprint arXiv:1906.08237*.