Natural Language Processing

Advanced Neural Network Language Models

Tianxing He
goosehe@cs.washington.edu

Outline

- Continue on RNN
- LSTM/GRU
- Seq2seq Model
- Attention
- (Briefly) Transformers and pretrained LMs

From last lecture, if you feel deep learning is magical…

- You are probably right!
- Nobody thinks DNN would work until Geoffrey Hinton.
- Nobody thinks RNN would work until Tomas Mikolov (gradient clipping).
- Dropout is originated from a bug in the code…
- A lot of other examples…
- A ton of receipe and tricks in this field simply because *it worked*.
- Just grasp the definitions of different NN modules, at some point, you will get used to these.

Heads-up

- In this lecture we will switch context between classification or (autoregressive) generation. Please be prepared.

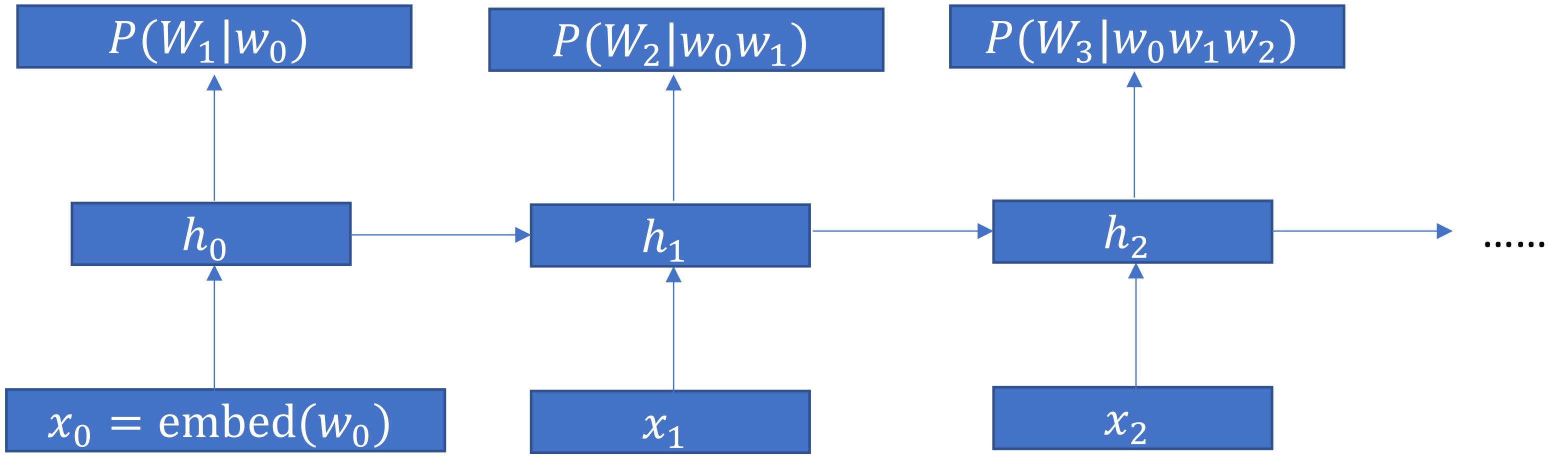
- For example, BERT is more about classification (sentence encoding), and GPT is more about autoregressive generation.

Recap: Recurrent neural network language model

- Complete formulation:

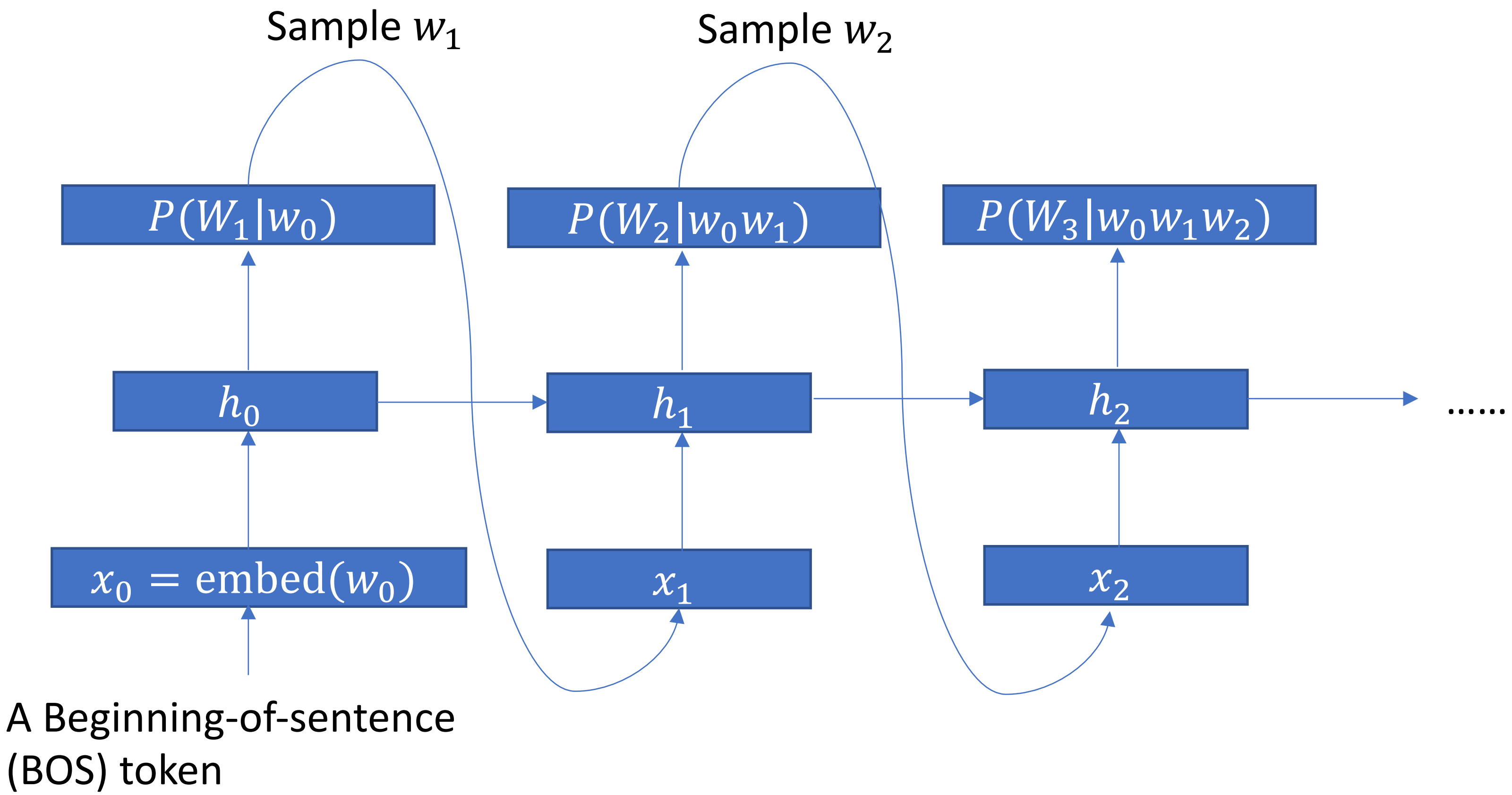
$$h_t = \sigma(W_{ih}x_t + W_{hh}h_{t-1} + b_h)$$
$$\mathrm{y}_t = \mathrm{softmax}(W_{ho}h_t + b_o)$$
$$\mathrm{L(w)} = \sum_i -\log P(w_i|w_{0..i-1})$$

- It's efficient: During training, we just feed the sequence (sentence) once into the RNN, and we get the output (loss) on every timestep.



Generation with RNNLM

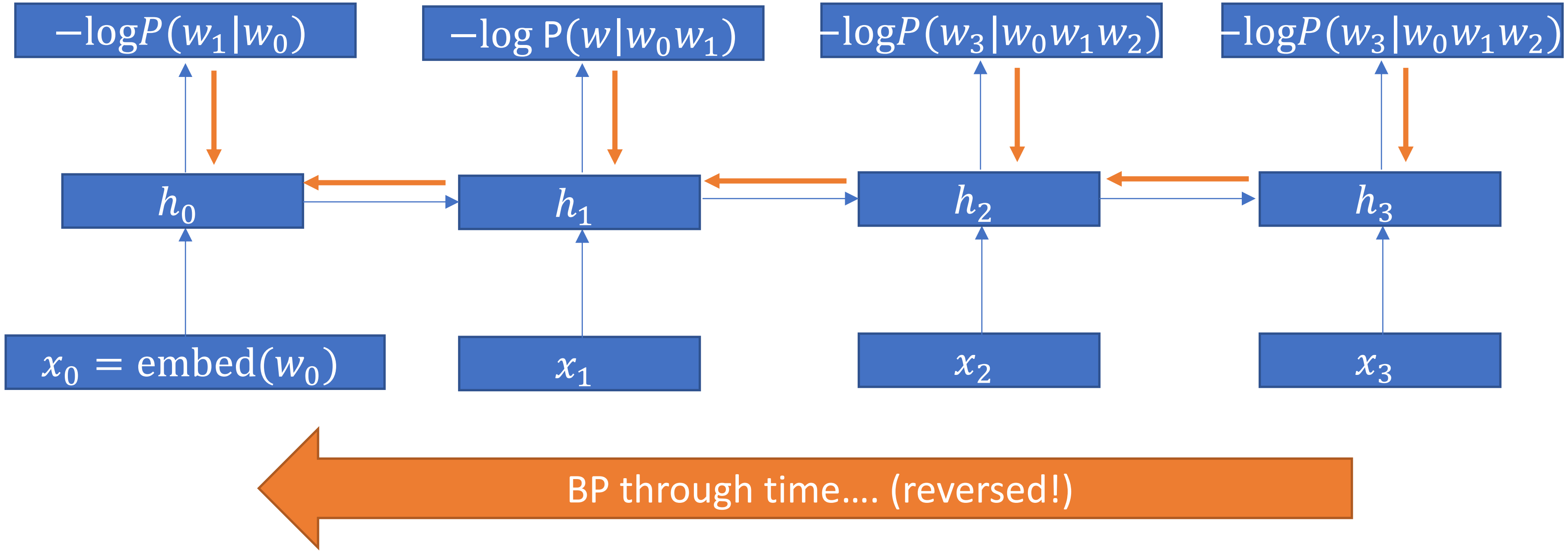
- We can do text generation with a trained RNNLM:
- At each time step t, we sample w_t from $P(W_t| \ldots)$, and feed it to the next timestep!
- LM with this kind of generation process is called autoregressive LM.



Gradient exploding and gradient vanishing

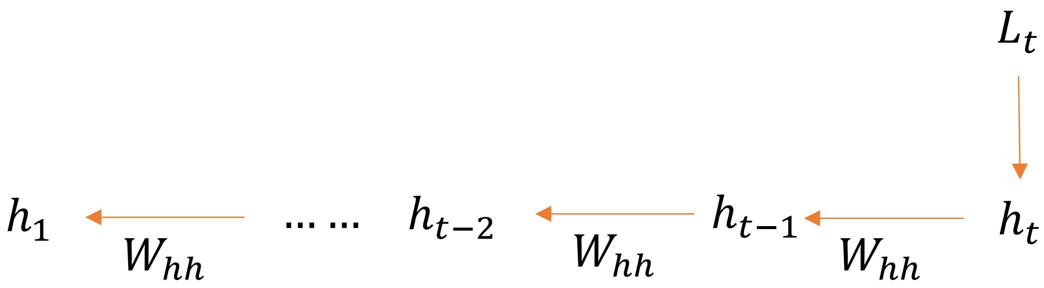
- In BPTT, we could meet two serious problems. They are called gradient exploding (error vector become too large) and gradient vanishing (error vector become too small).

- Gradient exploding is more serious because it makes training impossible.



Intuition: Gradient exploding and gradient vanishing

To give some intuition about the reason, we make two crude simplifications:
(1) Ignore the activation function: $h_t = W_{hh} h_{t-1} + W_{ih} x_t$ (2) only consider loss at time t: L_t



we get the following during error vector derivation:

$$\frac{\partial L_t}{\partial h_1} = \frac{\partial L_t}{\partial h_t} {W_{hh}}^{t-1}.$$

Further approximation! just think everything (especially W_{hh}) as a scalar…

when t is large and $W_{hh} < 1$: Gradient Vanishing!

when t is large and $W_{hh} > 1$: Gradient Exploding!

Gradient clipping for the exploding problem

It's simple!

Assume we want to set the maximum norm of gradient to be γ

$$\text{clip}(\nabla L) = \min\left\{1, \frac{\gamma}{||\nabla L||_2}\right\}\nabla L.$$

In practice, γ is a hyper-parameter, and is usually set to be 1 or 0.5.

(Brief) Word2vec

- Diverge topic a bit….
- The Word2vec project shows that if *we just want the word embeddings*, it can be trained in a very efficient way.
- Its training adopts the principle of *distributional hypothesis:*

“The meaning of a word is its use in the language”

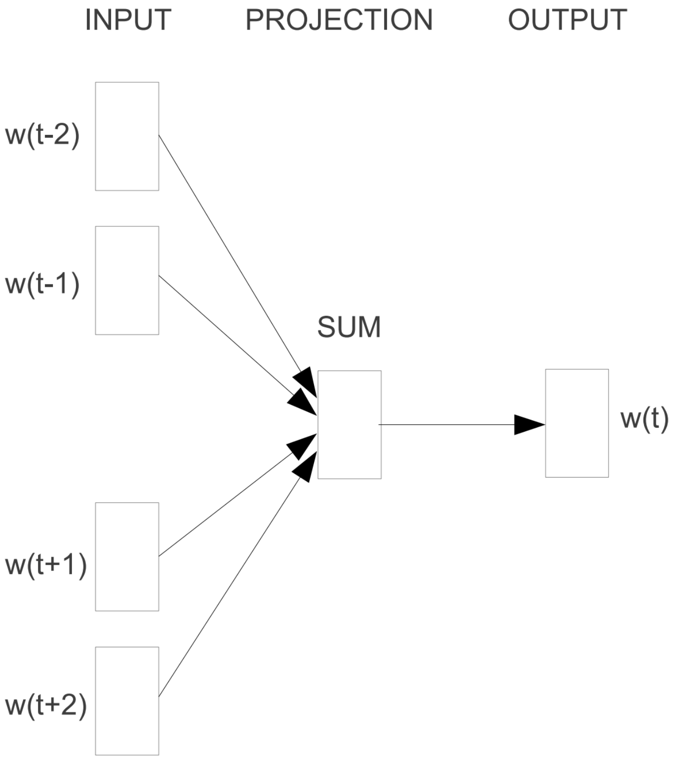
[Wittgenstein PI 43]

“You shall know a word by the company it keeps”

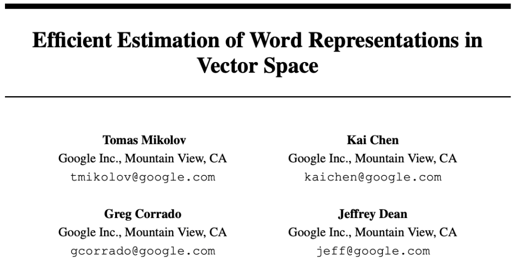
[Firth 1957]

If A and B have almost identical environments we say that they are synonyms.

[Harris 1954]



CBOW


Efficient Estimation of Word Representations in Vector Space

Tomas Mikolov
Google Inc., Mountain View, CA
tmikolov@google.com

Kai Chen
Google Inc., Mountain View, CA
kaichen@google.com

Greg Corrado
Google Inc., Mountain View, CA
gcorrado@google.com

Jeffrey Dean
Google Inc., Mountain View, CA
jeff@google.com

Word2Vec: The vector arithmetic

- We found the trained embeddings have amazing arithmetic properties.

- For example:
- emb(king)-emb(man)+emb(woman)=emb(queen)!

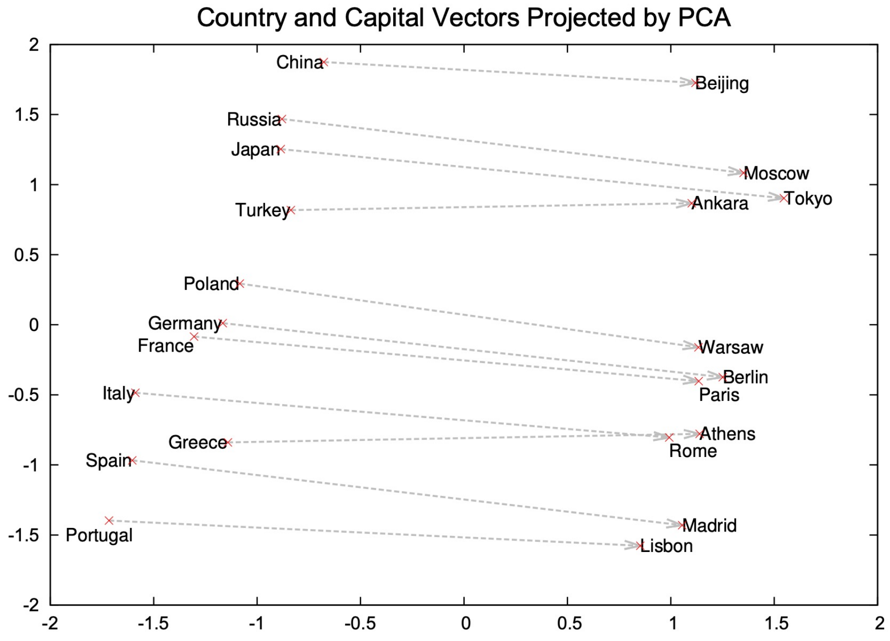


Figure 2: Two-dimensional PCA projection of the 1000-dimensional Skip-gram vectors of countries and their capital cities. The figure illustrates ability of the model to automatically organize concepts and learn implicitly the relationships between them, as during the training we did not provide any supervised information about what a capital city means.

Distributed Representations of Words and Phrases and their Compositionality

Tomas Mikolov
Google Inc.
Mountain View
mikolov@google.com

Ilya Sutskever
Google Inc.
Mountain View
ilyasu@google.com

Kai Chen
Google Inc.
Mountain View
kai@google.com

Greg Corrado
Google Inc.
Mountain View
gcorrado@google.com

Jeffrey Dean
Google Inc.
Mountain View
jeff@google.com

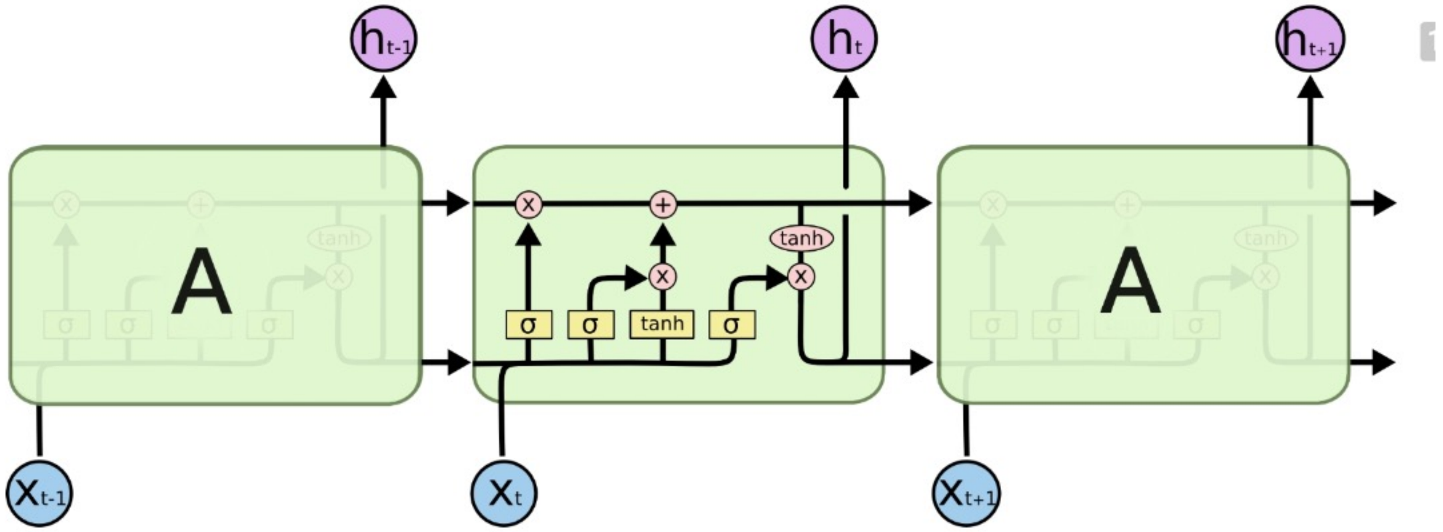
Word2Vec for initialization

- The training of word2vec can be done very efficiently on large unsupervised data (due to speed-up techniques like negative sampling).

- A good strategy: First pretrain a set of good word embeddings with a very large corpora. Then use it to initialize the embedding layer of your NN model. And finally finetune it on labeled data (e.g., for classification).

LSTM(skip) or GRU for gradient vanishing

- Historical note: The LSTM (long-short term memory) network was first used in (Sundermeyer et.al. 2012), dealing with the g-vanishing problem.
- Then, GRU (gated recurrent unit) is proposed as a simplification of LSTM.
- We will discuss GRU because it’s simpler and has the same core idea.

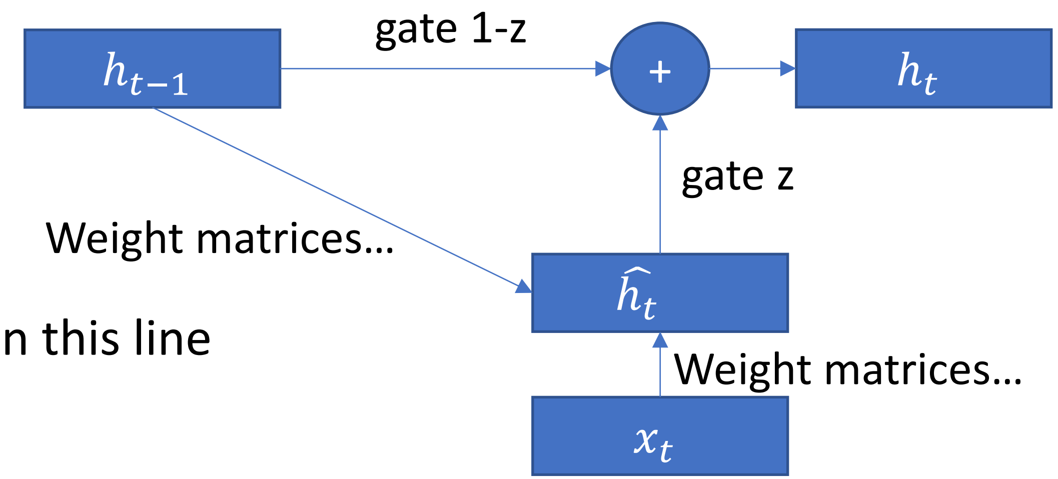


Recommend: Colah’s blog on Understanding LSTM Networks

Gated recurrent unit for gradient vanishing

GRU is by itself, a small neural network, input: x_t,h_{t-1} , output: h_t

$$z_t = \sigma_g(W_z x_t + U_z h_{t-1} + b_z)$$
$$r_t = \sigma_g(W_r x_t + U_r h_{t-1} + b_r)$$
$$\hat{h}_t = \phi_h(W_h x_t + U_h(r_t \odot h_{t-1}) + b_h)$$
$$h_t = z_t \odot \hat{h}_t + (1 - z_t) \odot h_{t-1}$$ <-Let's just focus on this line



Variables

- x_t: input vector
- h_t: output vector
- $\hat{h}_t$: candidate activation vector
- z_t: update gate vector
- r_t: reset gate vector
- W, U and b: parameter matrices and vector

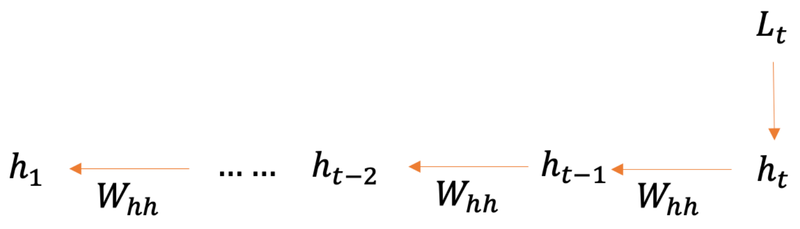
**Empirical Evaluation of
Gated Recurrent Neural Networks
on Sequence Modeling**

Junyoung Chung **Caglar Gulcehre** **KyungHyun Cho**
Université de Montréal

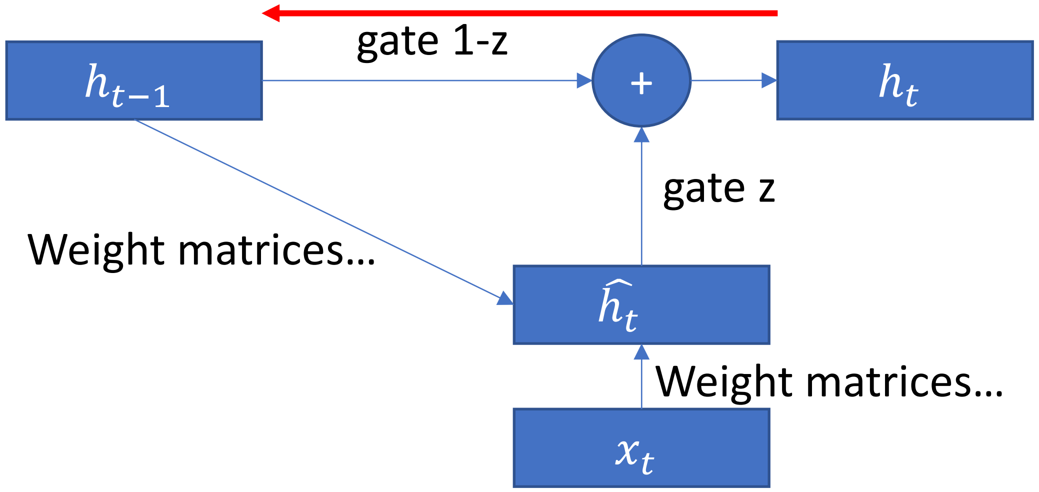
Yoshua Bengio
Université de Montréal
CIFAR Senior Fellow

Gated recurrent unit for gradient vanishing

- Think about back-propagation from h_t to h_{t-1}.
- There will be multiple paths, and the errors will be summed up. But in the red path, it does not involve any weight matrix! It's just $(1 - z)\odot h_{t-1}$.
- This path alleviates gradient vanishing.



The RNN case for reference.

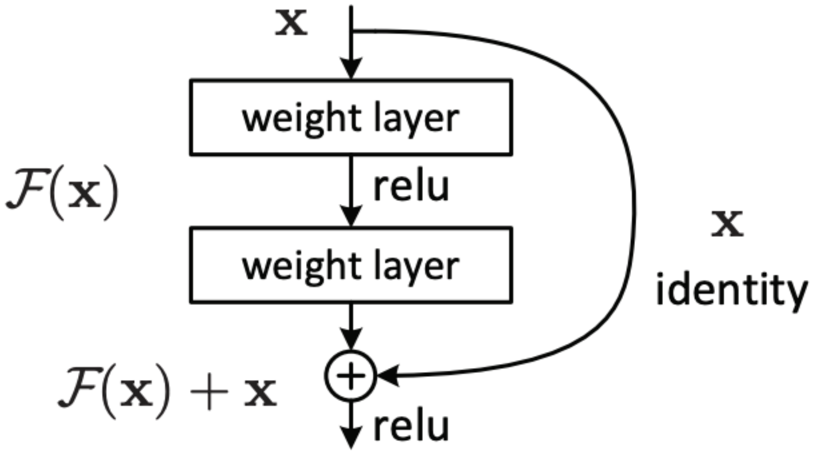


Residual connection in deep feedforward NN

- (Diverge topic a bit) Similar idea can be used to help us build deeper networks.

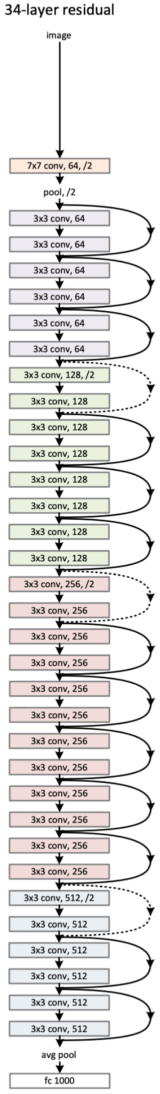
- Adding a direct link between hidden layers:
- $h_{l+1} = h_l + F(h_l)$
- F may include linear transform,ReLU, gating, etc.

- We will revisit this residual connection in transformers!



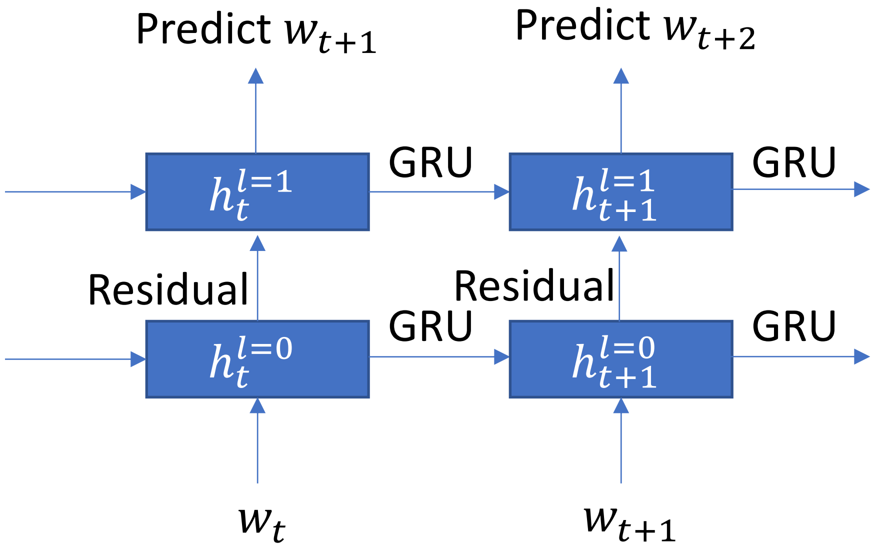
Deep Residual Learning for Image Recognition

Kaiming He Xiangyu Zhang Shaoqing Ren Jian Sun
Microsoft Research
{kahe, v-xiangz, v-shren, jiansun}@microsoft.com



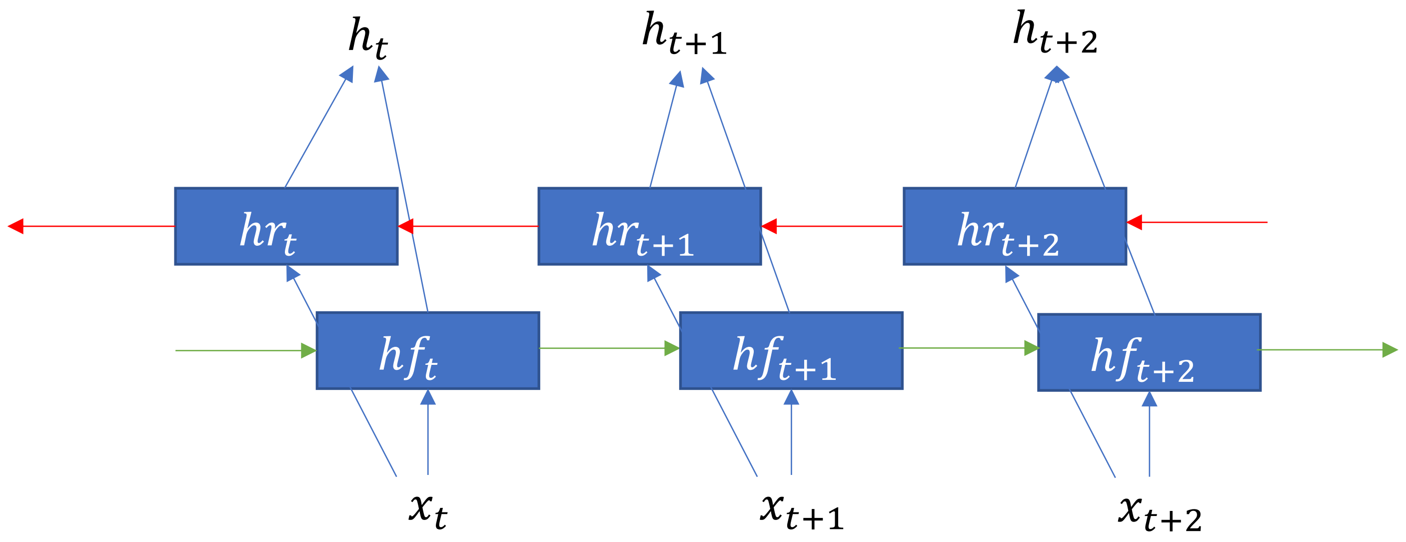
Philosophy: Combining NN modules

- We have now learnt several neural modules (rnn, lstm/gru, etc.), which are by themselves, a small neural network. We can combine different modules together to form a large neural model.
- For example, we build a AR-LM by stacking several GRU layers, and linking them with a residual link:



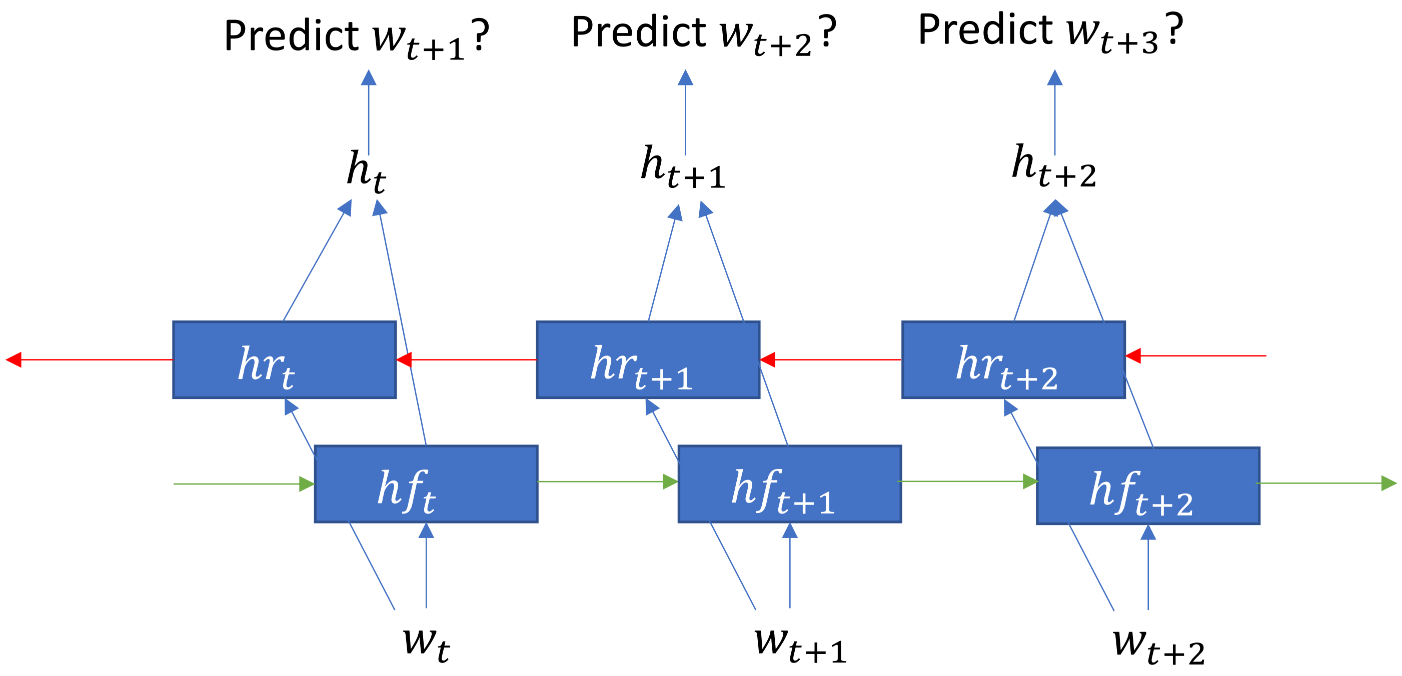
Bi-directional RNN

- In uni-directional RNN, h_t has context from the "left".
- For some applications (e.g., part-of-speech tagging), it would be useful if h_t has bi-directional context.
- We can achieve this by adding a layer of RNN with reversed direction.
- Exercise: what's the topological order of this graph (it's still a DAG!)?



Bi-directional RNN for AR-LM?

- Exercise: When we switch from a uni-rnn to a bi-rnn, and we don't change anything else, can we still do autoregressive language modelling?
- Answer: No! In autoregressive LM, we can not utilize information from the future!

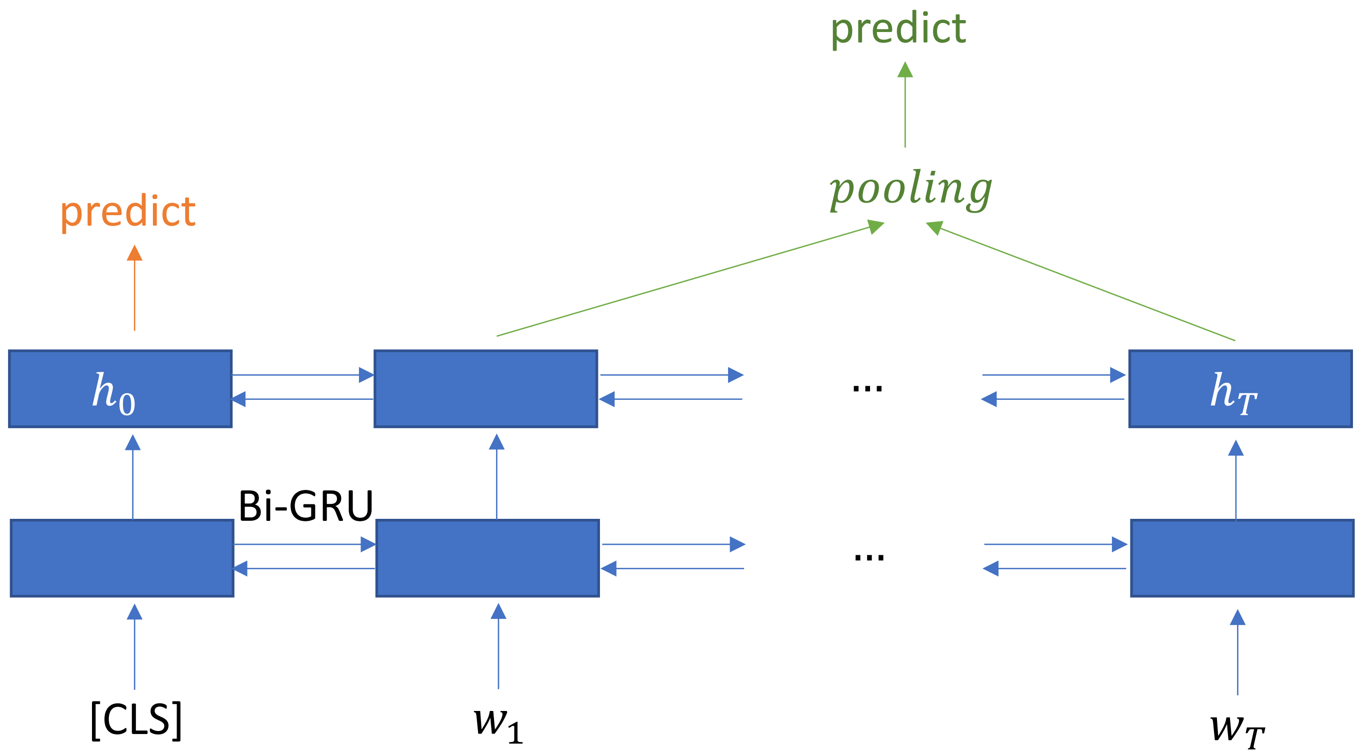


Bi-directional RNN for sentence-encoding?

There are several ways to get a sentence encoding from a bi-rnn:

Way1: add a special token to the input.

Way2: do a max-pooling or mean-pooling of the hidden states.



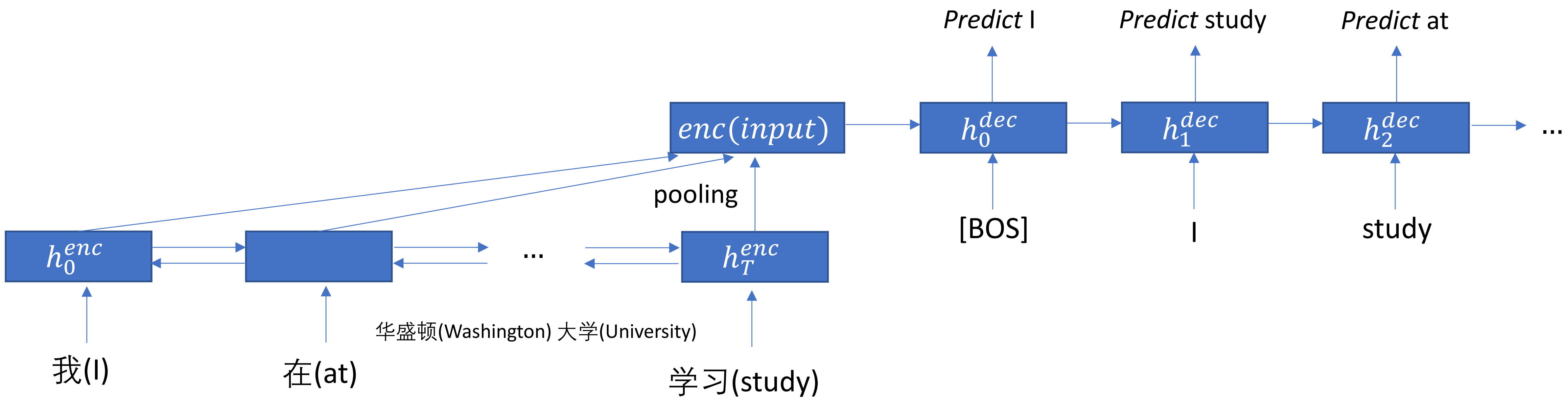
Encoder-decoder model for sequence-to-sequence (seq2seq) tasks

- Let’s switch context a bit and think about how to build a neural model for machine translation (MT, with is a seq2seq task).

- Example:
- Input: *我(I) 在(at) 华盛顿(Washington) 大学(University) 学习(study)。*
- Output: *I study at University of Washington.*

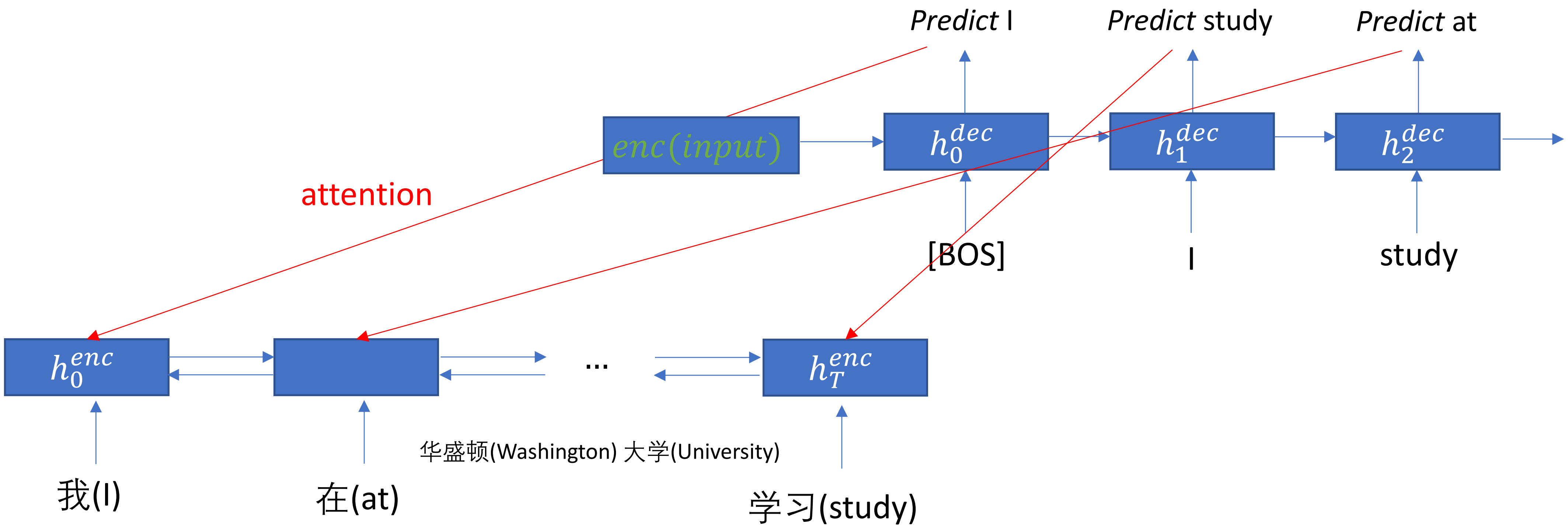
Encoder-decoder model for sequence-to-sequence (seq2seq) tasks

- We can use a bi-rnn encoder for the input sequence, and use a uni-rnn decoder for the output.
- In BP, the errors will be back-propagated from the decoder LM loss to the encoder.



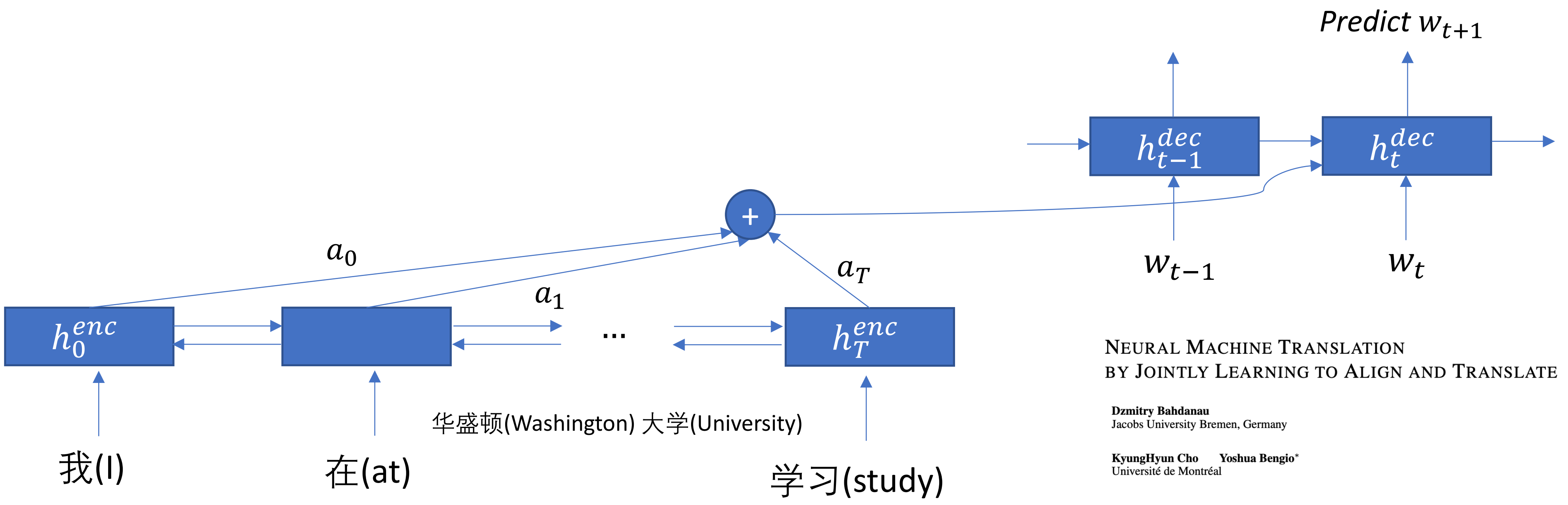
The attention mechanism: motivation

- Currently, all information in the input is condensed into a single vector.
- However, in tasks like MT, we may want to pay attention to different parts of the input in different timesteps.
- And this alignment is not trivial!
- The attention module is proposed to learn this alignment in an end-to-end fashion.



The attention mechanism

- We now focus on timestep t.
- For each encoder state h_i^{enc} , we compute an alignment score $\hat{a}_i = (h_i^{enc})^T W_a h_{t-1}^{dec}$.
- Then we get an attention distribution $a = softmax(\hat{a})$.
- We can then reweight the encoder states by a and pass $\sum_i a_i h_i^{enc}$ to the decoder.
- The parameter W_a is shared across time steps.

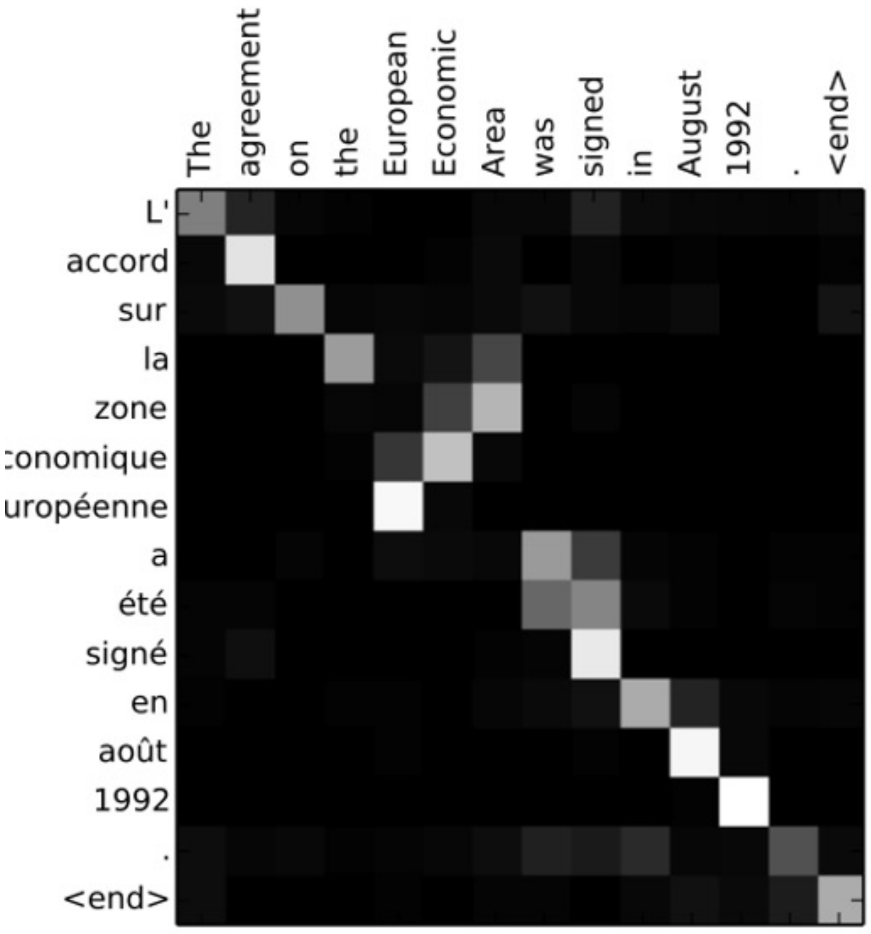


NEURAL MACHINE TRANSLATION
BY JOINTLY LEARNING TO ALIGN AND TRANSLATE

Dzmitry Bahdanau
Jacobs University Bremen, Germany

KyungHyun Cho Yoshua Bengio*
Université de Montréal

Attention: learned alignment example



(a)

Figure 3: Four sample alignments found by RNNsearch-50. The x-axis and y-axis of each plot correspond to the words in the source sentence (English) and the generated translation (French), respectively. Each pixel shows the weight α_{ij} of the annotation of the j-th source word for the i-th target word (see Eq. (6)), in grayscale (0: black, 1: white). (a) an arbitrary sentence. (b–d) three randomly selected samples among the sentences without any unknown words and of length between 10 and 20 words from the test set.

NEURAL MACHINE TRANSLATION
BY JOINTLY LEARNING TO ALIGN AND TRANSLATE

Dzmitry Bahdanau
Jacobs University Bremen, Germany

KyungHyun Cho Yoshua Bengio*
Université de Montréal

The transformer model (in a high-level)

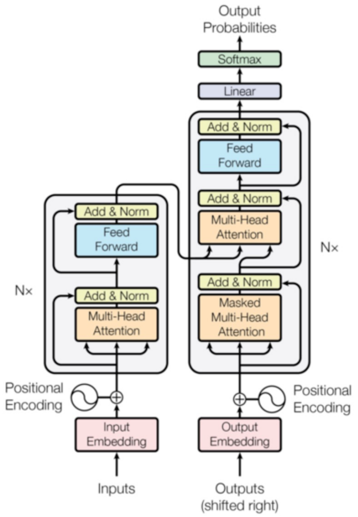
- In 2017, Google says “attention is all you need”, and proposes the transformer model.

- Over the years, it has become the most successful NN architecture in NLP.
- And it has been adopted later by the famous pretrained LM like BERT or GPT.

- We only have time to go over it in high-level today, but I recommend everyone to a great blog.
- *The Annotated Transformer*: https://nlp.seas.harvard.edu/2018/04/03/attention.html

Attention Is All You Need

Ashish Vaswani* Google Brain avaswani@google.com
Noam Shazeer* Google Brain noam@google.com
Niki Parmar* Google Research nikip@google.com
Jakob Uszkoreit* Google Research usz@google.com
Llion Jones* Google Research llion@google.com
Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu
Łukasz Kaiser* Google Brain lukaszkaiser@google.com
Illia Polosukhin* ‡ illia.polosukhin@gmail.com

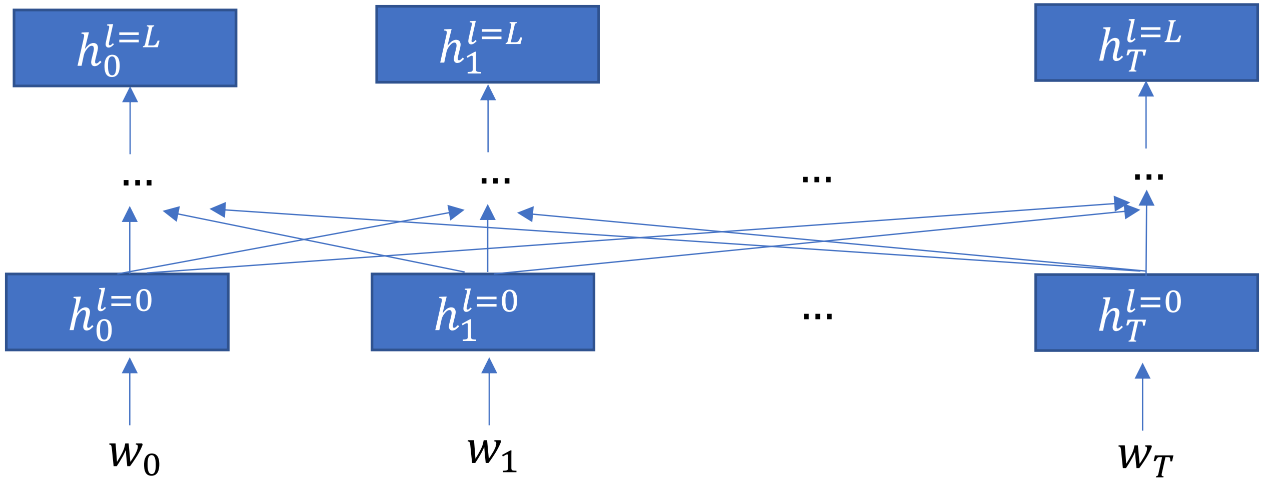


The transformer model (in a high-level)

- Actually, the transformer model is not that mysterious or magical.

- It will be easier to view it as a smart combination of modules (attention, residual block, dropout, layer-norm, etc.).

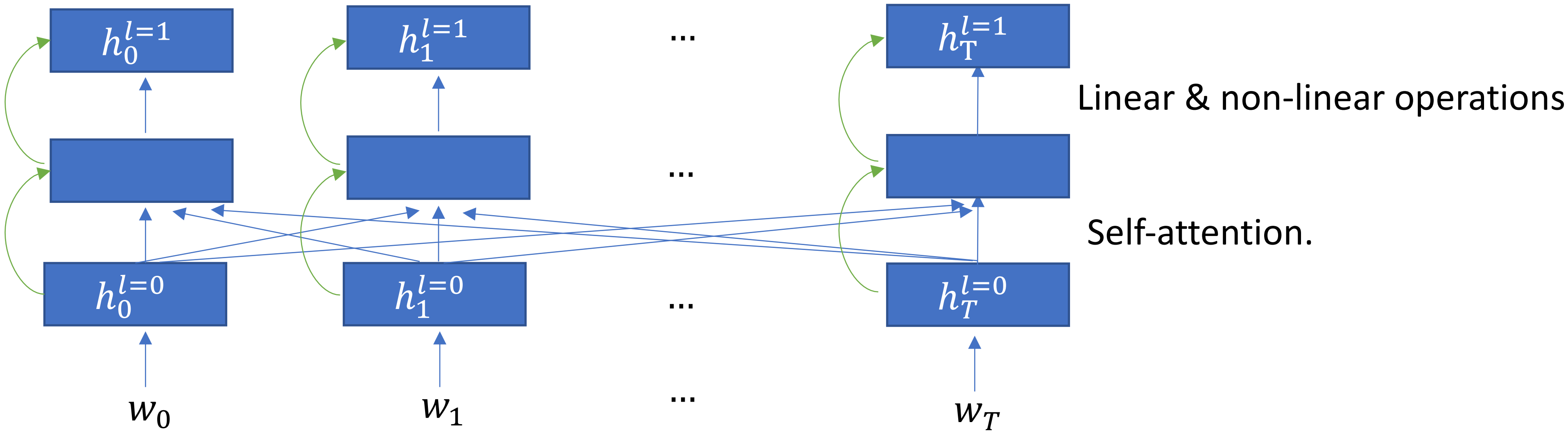
- From a very high level, the transformer is kinda like a multi-layer bi-RNN, giving encoding for each token.
- It is composed of a number of layers, each layer is called a “TF block”.



A TF block

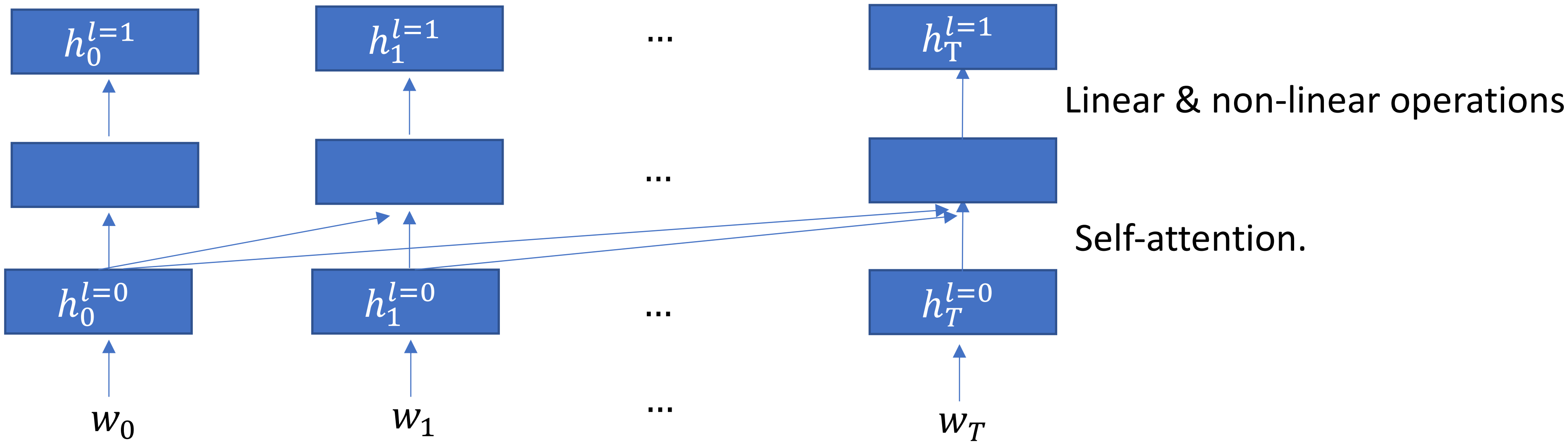
- A TF block has two sub-layers.

- First, a self-attention (a complicated version of attention) layer to exchange information among different timesteps.
- Then, a feedforward module to transform the encodings.
- Finally, there are dropout & layer-norm & residual links added to ease the optimization.



For AR-LM: Causal mask for self-attention

- To do auto-regressive LM, we need to apply a “causal” mask to self-attention, forbidding it from getting future context.
- At timestep t, we set $a_i = 0$ for $i > t$.



The transformer model (in a high-level)

Remarks:

- The name “attention is all you need” is mostly because there’s no RNN module and its job is left to the self-attention layer.

- The transformer model is good at scaling: training a deeper model with larger amounts of data, usually gives visible performance gain.

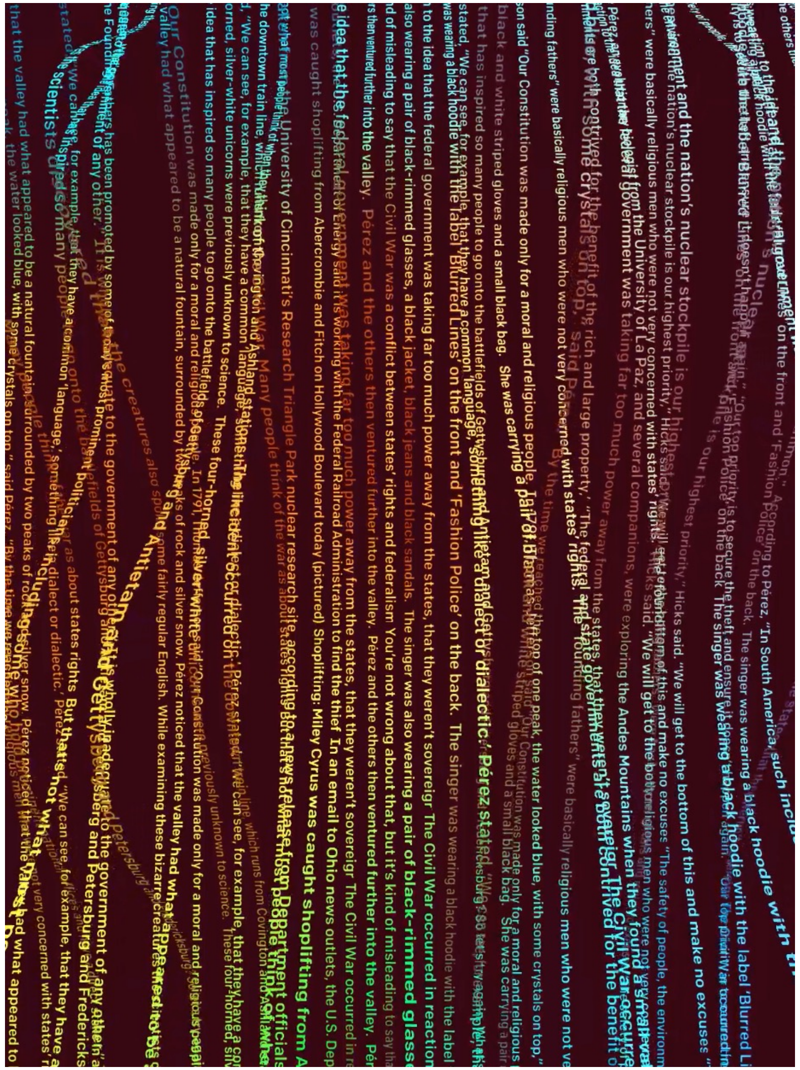
- We did not cover deep learning regularization techniques (dropout, layer-norm, etc.) Please find online resources about them, for example:
- https://medium.com/techspace-usict/normalization-techniques-in-deep-neural-networks-9121bf100d8
- https://www.deeplearningbook.org/ Chapter 7 & 8.

The GPT models from OpenAI

In recent years, OpenAI released a series of large pretrained LMs, GPT, GPT2 and GPT3. (generative pretrained transformer)

They are basically larger and larger autoregressive transformer LM trained on larger and larger amounts data.

They have shown amazing language generation capability when you give it a prompt (aka. prefix, the beginning of a paragraph).



Generation example from the GPT2 model

SYSTEM PROMPT (HUMAN-WRITTEN)	*In a shocking finding, scientist discovered a herd of unicorns living in a remote, previously unexplored valley, in the Andes Mountains. Even more surprising to the researchers was the fact that the unicorns spoke perfect English.*
MODEL COMPLETION (MACHINE-WRITTEN, 10 TRIES)	The scientist named the population, after their distinctive horn, Ovid's Unicorn. These four-horned, silver-white unicorns were previously unknown to science. Now, after almost two centuries, the mystery of what sparked this odd phenomenon is finally solved. Dr. Jorge Pérez, an evolutionary biologist from the University of La Paz, and several companions, were exploring the Andes Mountains when they found a small valley, with no other animals or humans. Pérez noticed that the valley had what appeared to be a natural fountain, surrounded by two peaks of rock and silver snow.

A sample from GPT2 (with top-k sampling)

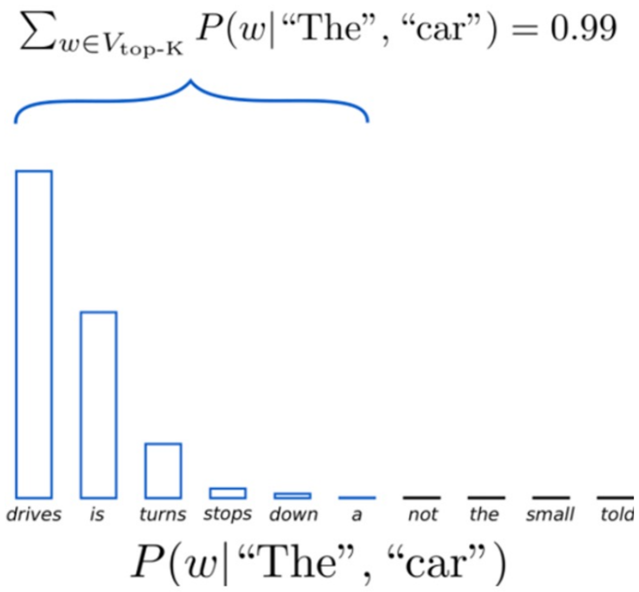
The top-K sampling algorithm

We will represent $P(\cdot\,|W_{1..i})$ by $p = (p_1, p_2, \ldots, p_{|V|})$, where the elements is sorted that $p_1 \geq p_2 \geq p_3 \ldots \geq p_{|V|}$.

Top-K sampling transforms p to $\hat{p}$ by:

$$\widehat{p_i} = \frac{p_i \cdot 1\{i \leq K\}}{Z}$$

And we sample W_{i+1} from $\hat{p}$.



<- from https://huggingface.co/blog/how-to-generate
Recommended-reading!

Examples from the GPT2 model

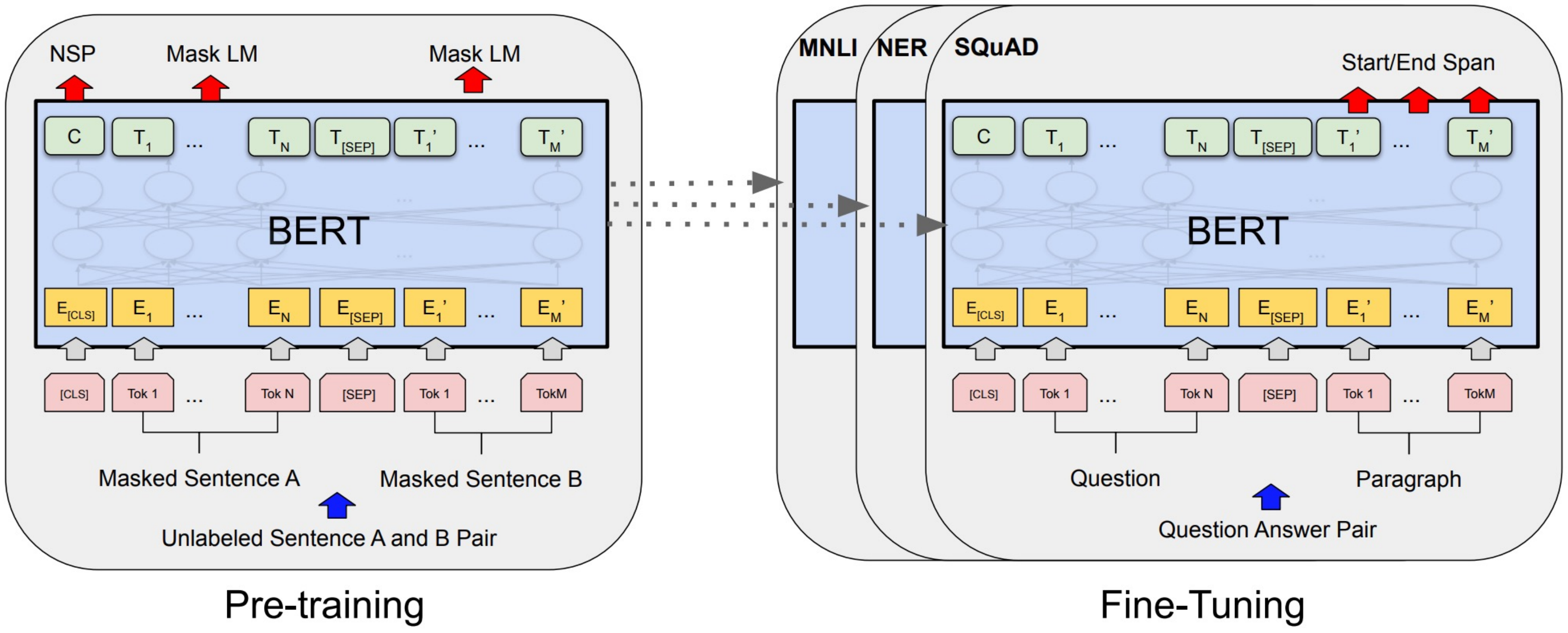
- Prompt: *MIT is a private research university in Cambridge, Massachusetts. It is one of the best universities in the U.S.,*

- GPT2 with naive sampling: but the teaching of traditional African-American studies and African-American literacy continued. Soon thereafter, MIT was renamed The International Comparative University by Lord (then), …
- GPT2 with topk40 sampling: and the home of most of the top international universities in the world. Our alumni are internationally renown, but our mission is unique. We are the only university in the world where there is a chance to take on the challenge of making an impact, …
- topk40 another sample: with a reputation for innovation and open and flexible public systems. Its principal research area deals with autonomous vehicles, robotics and artificial intelligence. To date, MIT has published 40 peer-reviewed papers on this topic, …

- Message: sampling algorithms provide a sweet quality-diversity trade-off.
- (which is the key difference to decoding e.g., beam-search)
- I did not do cherry-pick.

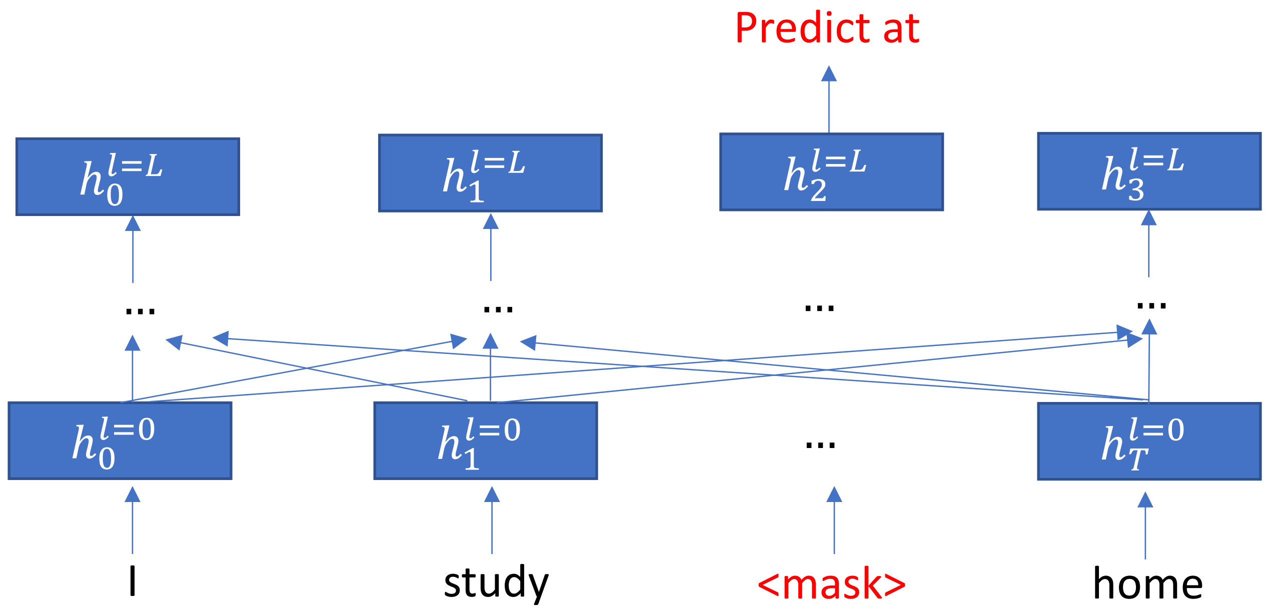
BERT

- In 2019, Google released the BERT (Bidirectional Encoder Representations from Transformers) model.
- It's a large transformer encoder pretrained on huge data with the masked LM objective.
- Superior performance is shown once finetuned on downstream NLUnderstanding tasks.



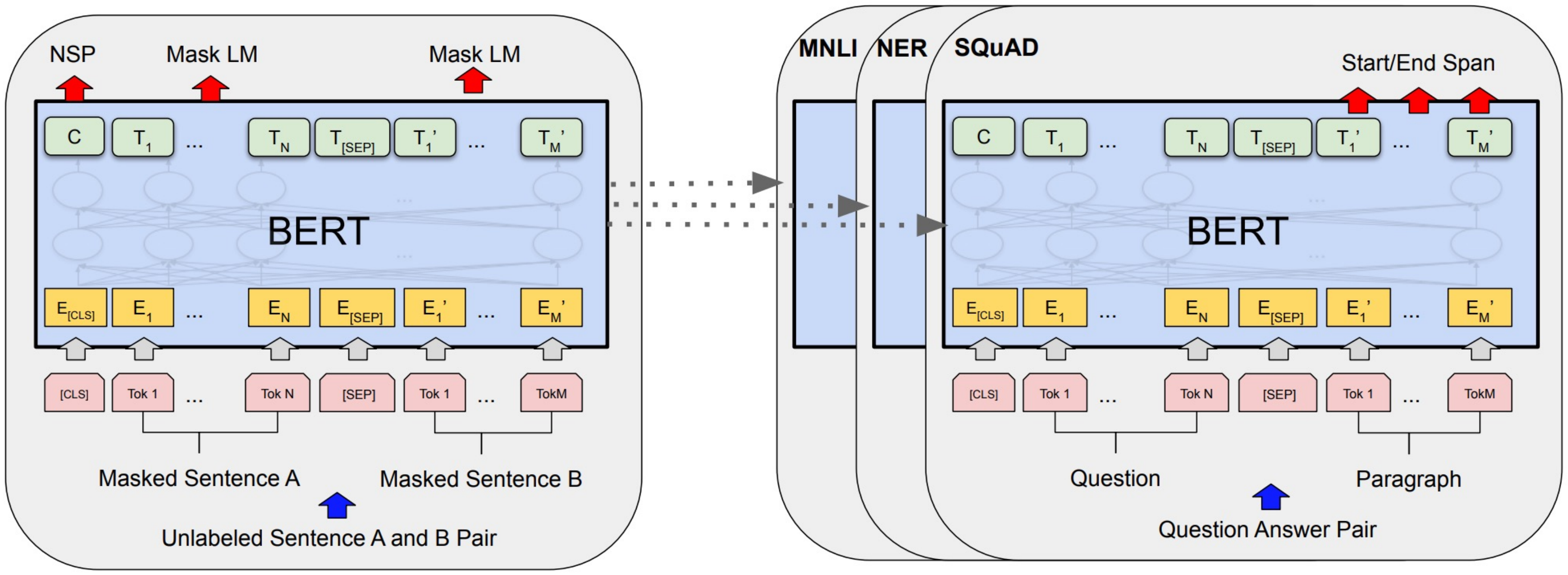
BERT: Masked LM objective

- MLM can be viewed as a bi-directional version of next-word prediction in AR-LM.
- We mask a portion of words in the sentence, pass it to the transformer encoder, and predict the masked words.



The pretrain-finetune paradigm

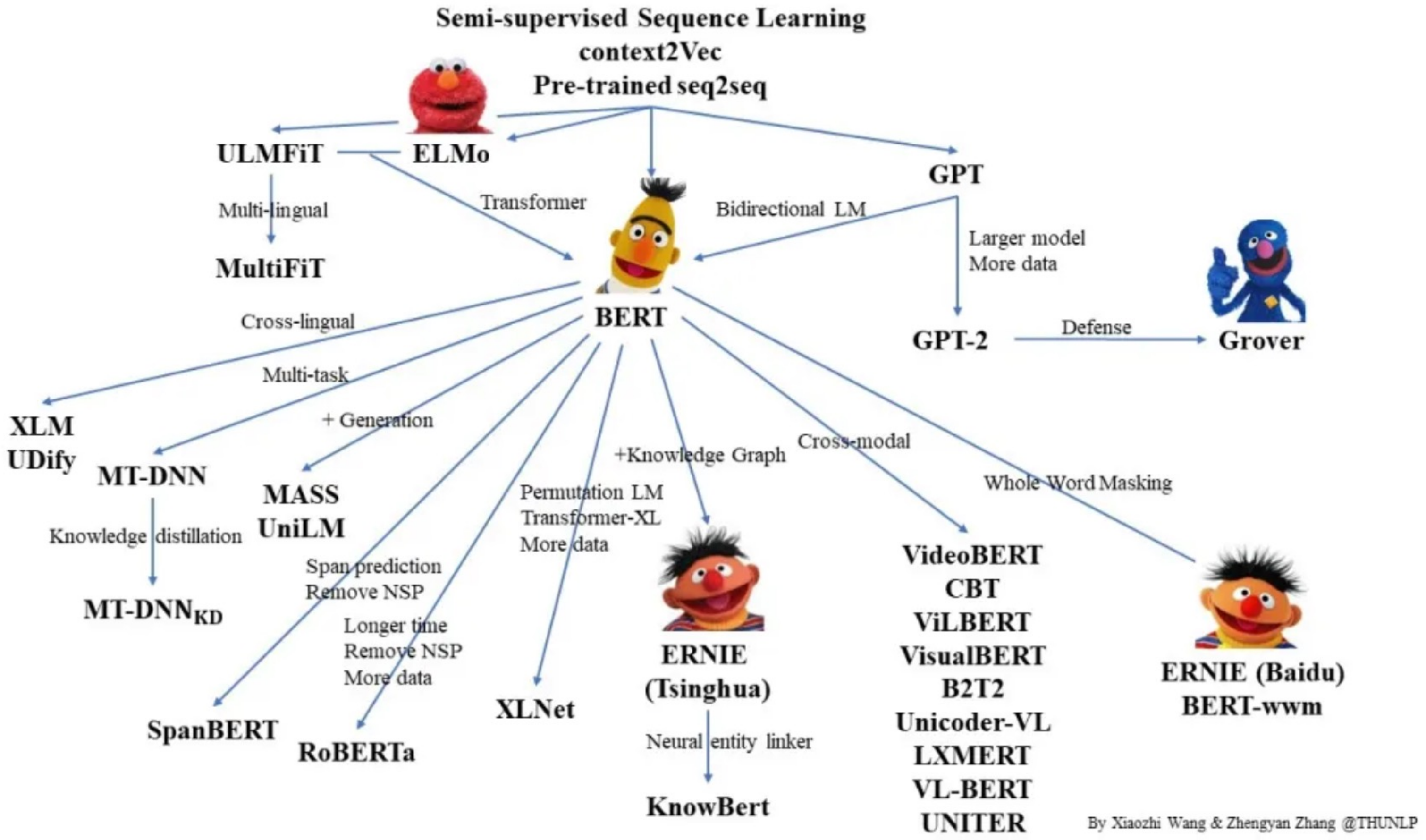
- The pretrain-finetune paradigm has become a foundational paradigm in NLP.
- Just pick a pretrained LM, and finetune (continue training) it on the downstream task you care about.



Pre-training Fine-Tuning

BERTology

- Pick your favourite PLM! (this figure is not up-to-date)



The Transformer (Muppet) family | Source: PLM Papers

Huggingface examples about finetuning PLM

Hugging Face

- Huggingface is an amazing organization trying to make the use of pretrained LM easier.
- Finetune gpt2: https://github.com/huggingface/transformers/blob/main/examples/pytorch/language-modeling/run_clm_no_trainer.py
- Finetune bert: https://github.com/huggingface/transformers/tree/main/examples/pytorch/text-classification
- This maybe the most attractive homework in this lecture!
- To run large models you will need GPU machine, the google-colab might be a good toy testbed (change the runtime to GPU).

Looking for bigger challenges? Try read some papers!

I can recommend these two papers (ofc there are many many more good papers!).

Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context

Zihang Dai[*12], Zhilin Yang[*12], Yiming Yang[1], Jaime Carbonell[1], Quoc V. Le[2], Ruslan Salakhutdinov[1]
[1]Carnegie Mellon University, [2]Google Brain
{dzihang,zhiliny,yiming,jgc,rsalakhu}@cs.cmu.edu, qvl@google.com

FUDGE: Controlled Text Generation With Future Discriminators

Kevin Yang
UC Berkeley
yangk@berkeley.edu

Dan Klein
UC Berkeley
klein@berkeley.edu

A little more about me:
I work in neural language generation.
https://people.csail.mit.edu/cloudygoose/

Tianxing He （贺天行）

Hi! I'm currently a postdoc at UW, supervised by Yulia Tsvetkov, who runs the Tsvetshop. Not long ago, I was a PhD student at MIT, supervised by Prof. James Glass, who runs the SLS group. My research interest lies in natural language processing and deep learning. Most of my works during my PhD is focused on neural language generation.

You can download my PhD defense slides here.

Thanks!