Inspire…Educate…Transform.

Deep Learning for NLP

Dr. Anand Narasimhamurthy

Senior Associate Professor
International School of Engineering

Acknowledgement : A significant amount of material is drawn from slides by Dr. Sreerama Murthy

Outline

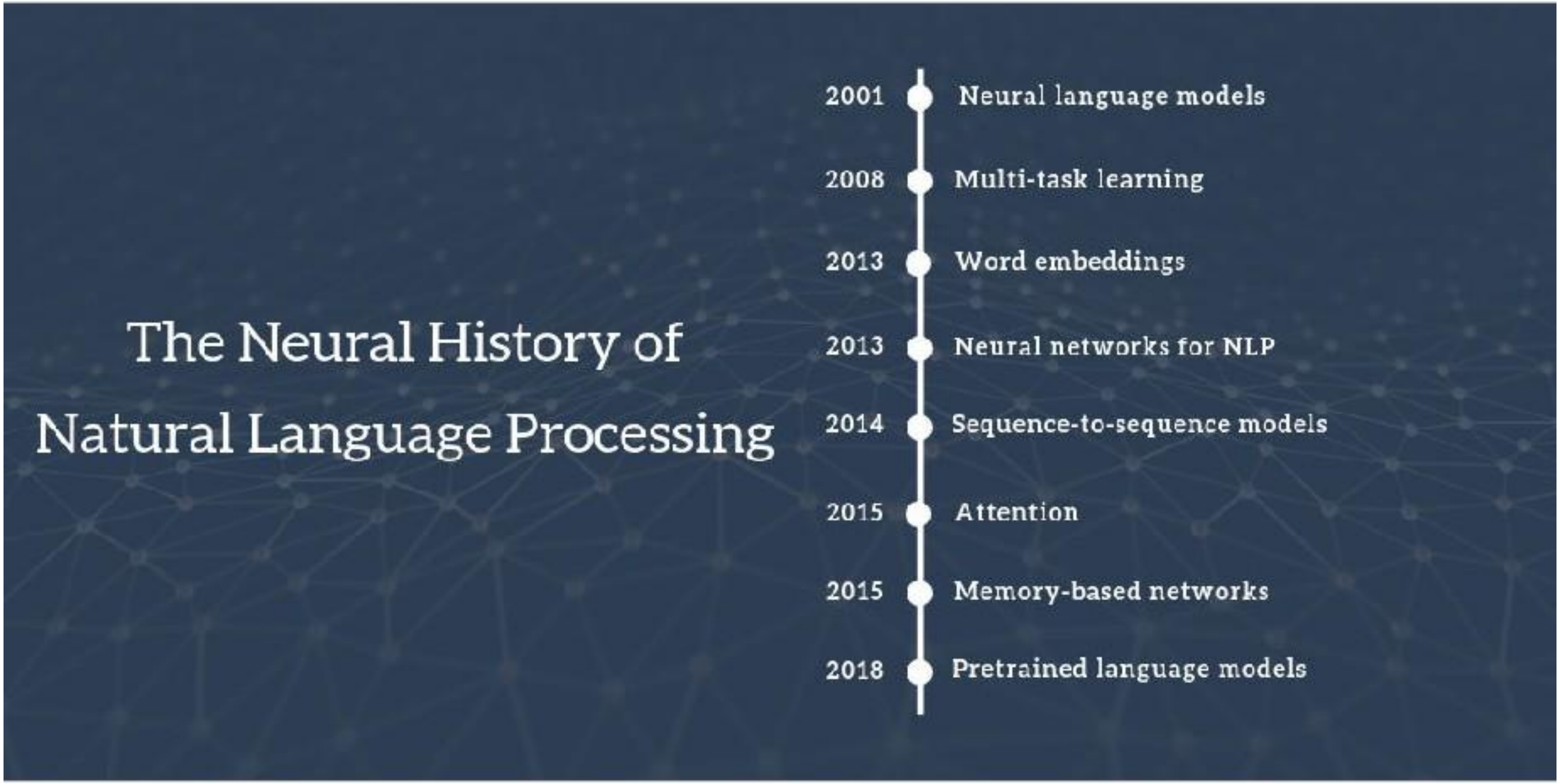
- A little more Attention
- Transformers
- Transfer learning and ULMFit
- BERT, GPT

Recap of architectures covered thus far

- Recurrent Neural Networks (RNN)
- Long Short Term Memory Networks (LSTM)
- Gated Recurrent Unit (GRU)

Other architectures

- Bidirectional models
- Encoder Decoder
- Attention mechanism



The Neural History of Natural Language Processing

2001	Neural language models
2008	Multi-task learning
2013	Word embeddings
2013	Neural networks for NLP
2014	Sequence-to-sequence models
2015	Attention
2015	Memory-based networks
2018	Pretrained language models

RNN Summary

Promise

RNN can encode sentence meaning

Can predict next word given a sequence of words.

Can be trained to learn a language model

Problems

- Hard to parallelize efficiently
- Back propagation through variable length sequence
- Transmitting information through one bottleneck (final hidden state)
- longer sentences lose context (e.g., I was born in France..I speak fluent French)

(There are improvements to RNN structure that retain context)

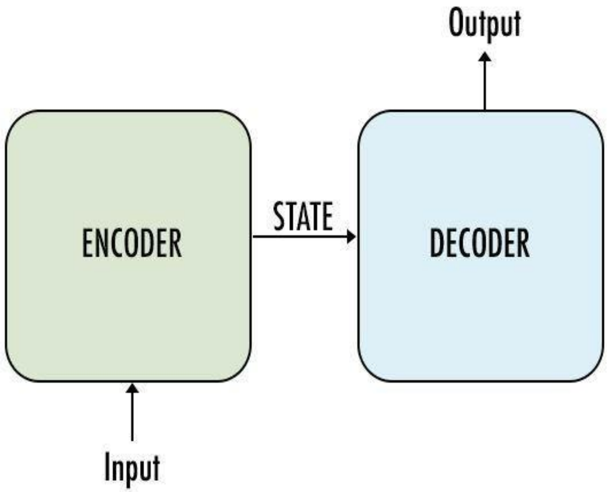
Encoder – Decoder Summary

Two RNNs jointly trained.

The first RNN creates context

The second RNN uses the final context of the first RNN.

Training using Parallel corpora



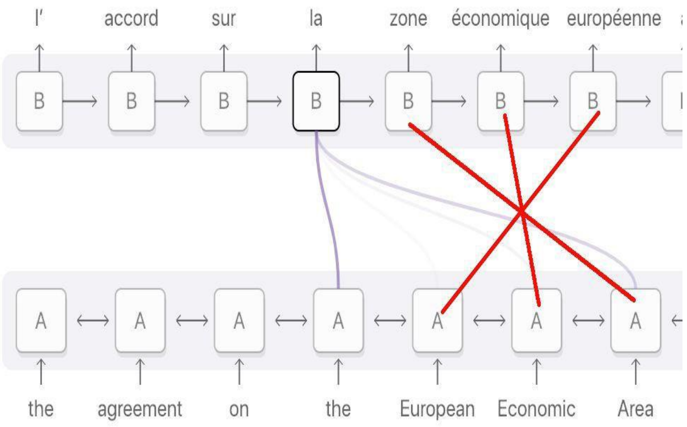
- 2014 -Google replaced their Statistical model with Neural Machine Translation, using E-D.

- Due to its flexibility, it is the go-to framework for NLG with different models taking roles of encoder and decoder

- The decoder can not only be conditioned on a sequence but on arbitrary representation, enabling many use cases (e.g., generating caption from image)

Issues

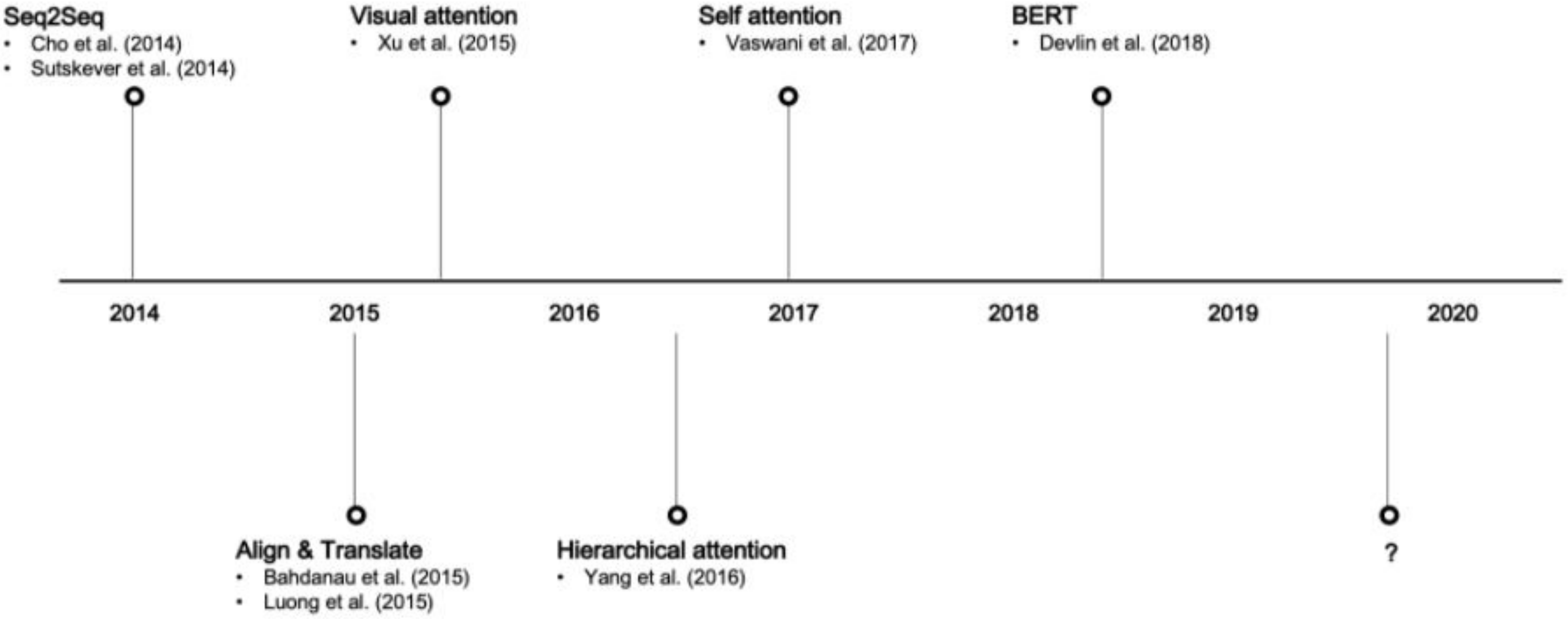
- Meaning is crammed
- Long term dependencies
- Word alignment
- Word interdependencies
- The model cannot remember enough
- Only final Encoder state is used



Some more attention

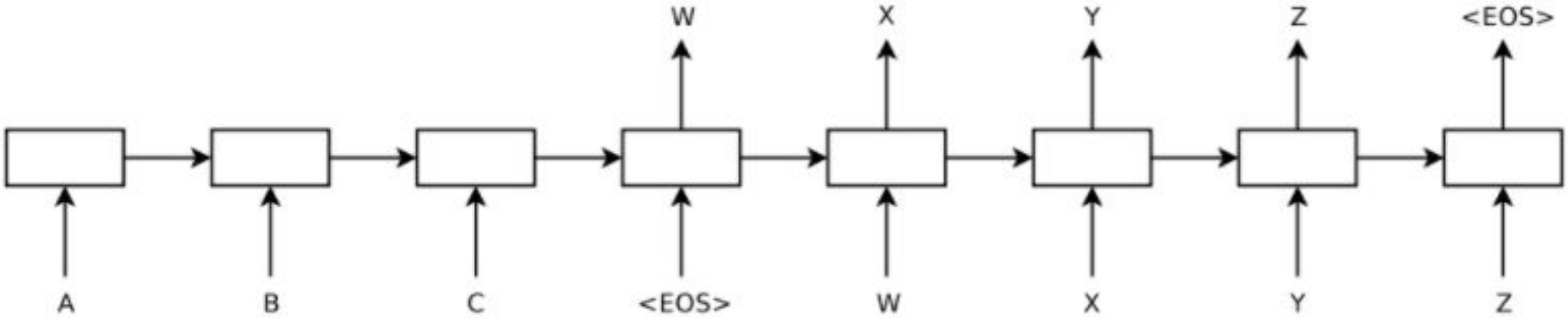
Attention is arguably one of the most powerful concepts in the deep learning field now. It is based on a commonsensical intuition that we **"attend to"** a certain part when processing a large amount of information.

Key Developments in Attention



https://buomsoo-kim.github.io/attention/2020/01/01/Attention-mechanism-1.md/

Seq2Seq (Sequence to sequence)



A potential problem of the vanilla Seq2Seq architecture is that some information might not be captured by a fixed-length vector, i.e., the final hidden state from the encoder (h_t).
This can be especially problematic when processing long sentences where RNN is unable to send adequate information to the end of the sentences due to gradient exploding, etc.

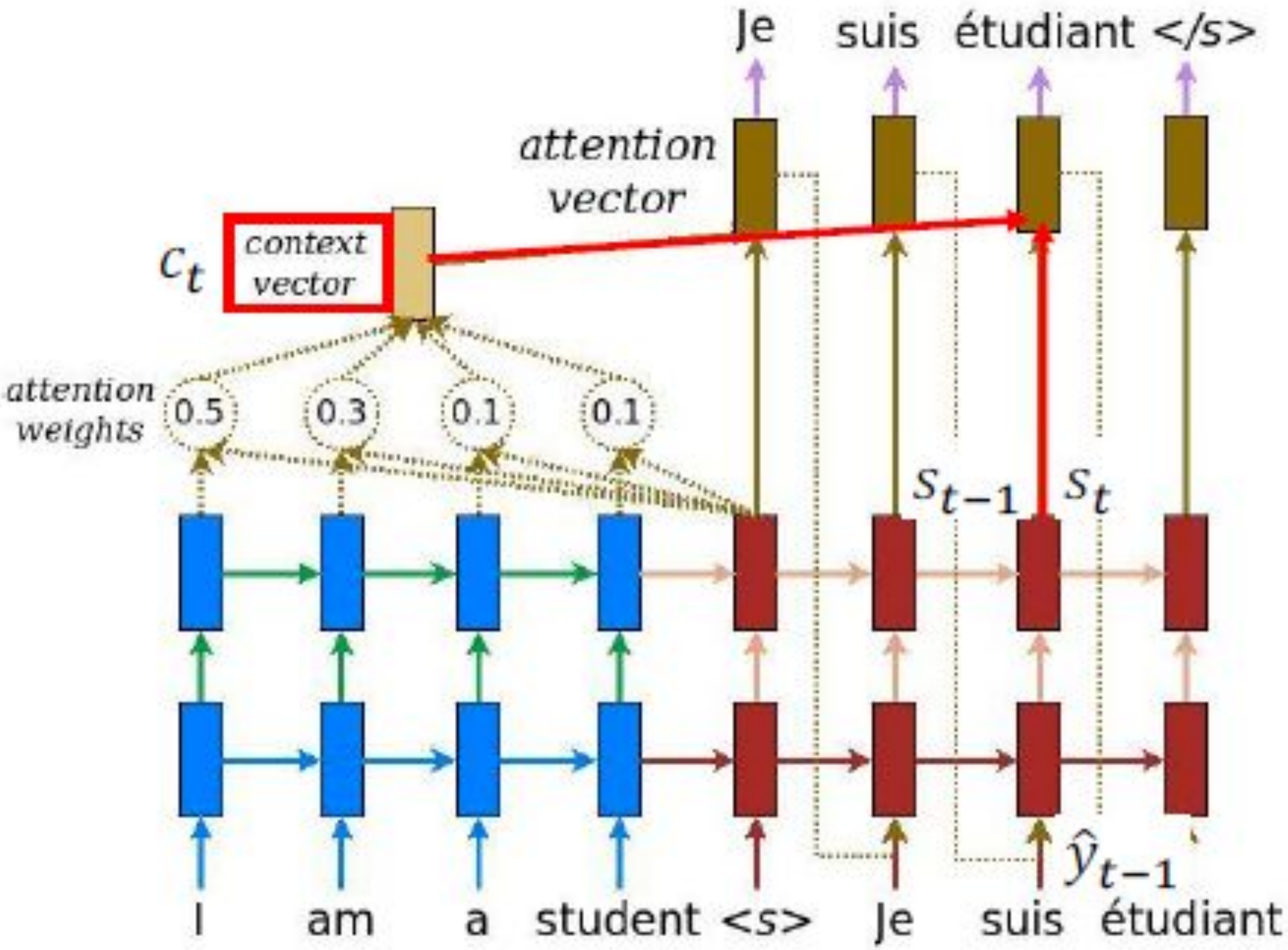
Attention

Motivation:

- Previous models **summarize** inputs into a single feature vector
- Hence, the model forgets old inputs, especially for long sequences
- Idea: Use input features, but attend on the most important features

*Source:

Attention



- Now the **decoder hidden state** s_t is a function of previous state s_{t-1}, current input $\hat{y}_{t-1}$, and context vector c_t, i.e., $s_t = f(s_{t-1}, \hat{y}_{t-1}, c_t)$

Attention Modeling

- Method:
  - **Task:** Translate source sequence $[x_1, \ldots, x_n]$ to target sequence $[y_1, \ldots, y_m]$
  - Now the **decoder hidden state** s_t is a function of previous state s_{t-1}, current input $\hat{y}_{t-1}$, and context vector c_t, i.e., $s_t = f(s_{t-1}, \hat{y}_{t-1}, c_t)$
  - The context vector c_t is **linear combination** of input hidden features $[h_1, \ldots, h_n]$

$$c_t = \sum_{i=1}^{n} \alpha_{t,i} h_i$$

  - Here, the weight $\alpha_{t,i}$ is alignment score of two words y_t and x_i

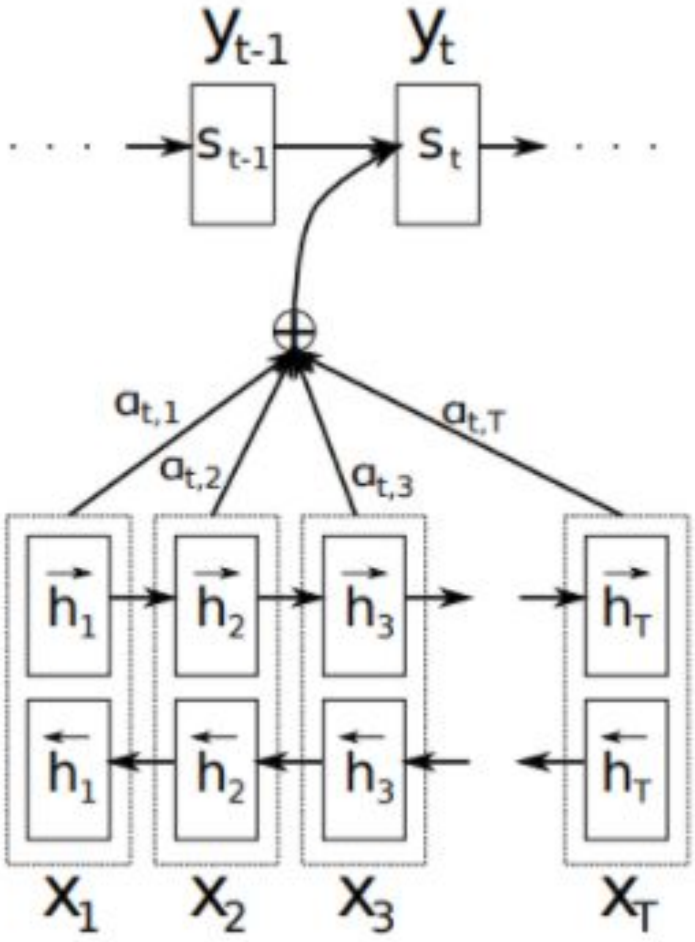
$$\alpha_{t,i} = \text{align}(y_t, x_i) = \frac{\text{score}(s_{t-1}, h_i)}{\sum_{i'} \text{score}(s_{t-1}, h_{i'})}$$

  where score is also jointly trained, e.g., $\text{score}(s_t, h_i) = \mathbf{v}^T \tanh(\mathbf{W}[s_t; h_i])$

Align & Translate – Core Idea

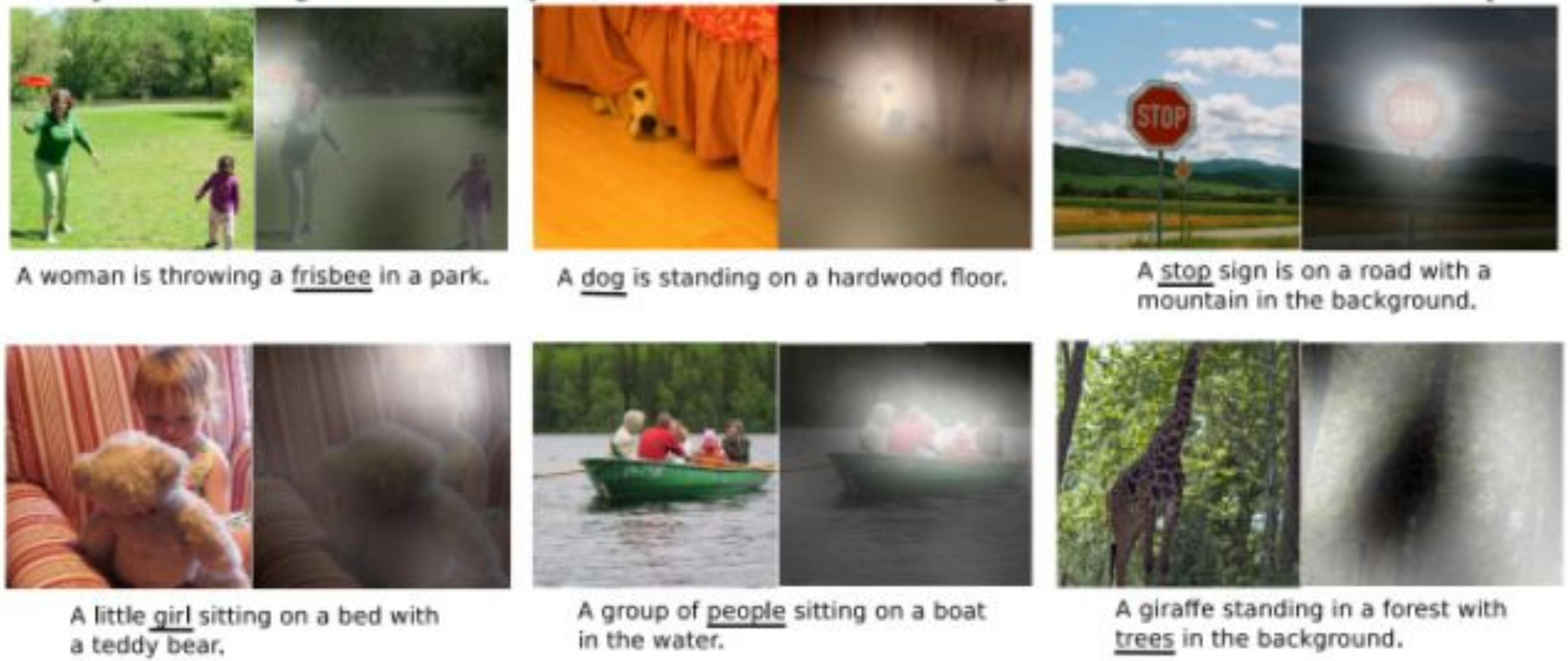
The context vector preserves information from all hidden states from encoder cells and aligns them with the current target output. By doing so, the model is able to *"attend to"* a certain part of the source inputs and learn the complex relationship between the source and target better.

Luong et al. (2015) outlines various types of attention models to align the source and target.



Application of Attention to image data

Figure 4. Examples of attending to the correct object (*white* indicates the attended regions, *underlines* indicated the corresponding word)



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.

Xu et al. (2015) proposed an attention framework that extends beyond the conventional Seq2Seq architecture. Their framework attempts to align the input image and output word, tackling the image captioning problem.

Hierarchical Attention

Yang et al. (2016) demonstrated with their hierarchical attention network (HAN) that attention can be effectively used on various levels.

Also, they showed that attention mechanism applicable to the classification problem, not just sequence generation.

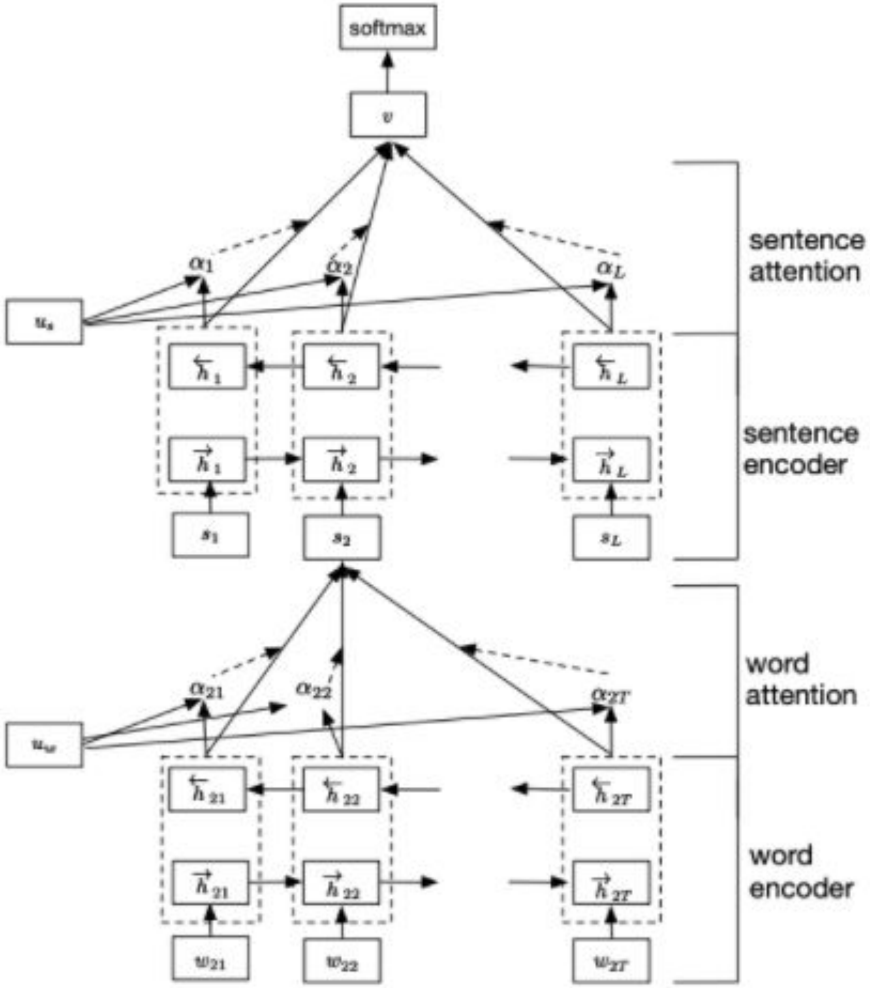
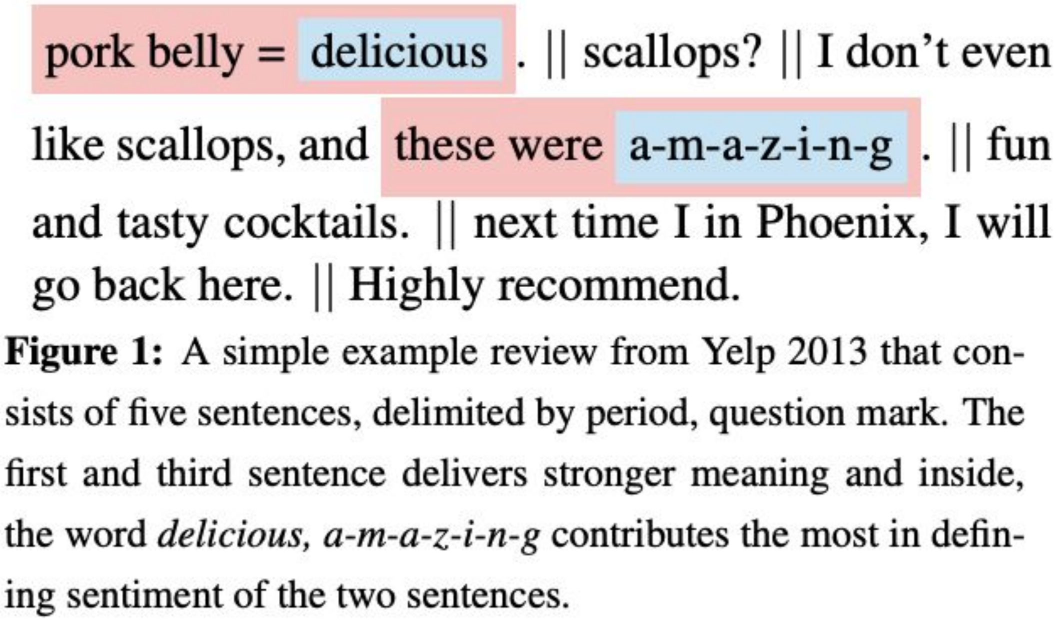


Figure 2: Hierarchical Attention Network.

Classification with HAN



pork belly = delicious . || scallops? || I don’t even like scallops, and these were a-m-a-z-i-n-g . || fun and tasty cocktails. || next time I in Phoenix, I will go back here. || Highly recommend.

Figure 1: A simple example review from Yelp 2013 that consists of five sentences, delimited by period, question mark. The first and third sentence delivers stronger meaning and inside, the word *delicious, a-m-a-z-i-n-g* contributes the most in defining sentiment of the two sentences.

HAN enables hierarchical interpretation of results as below. The user can understand

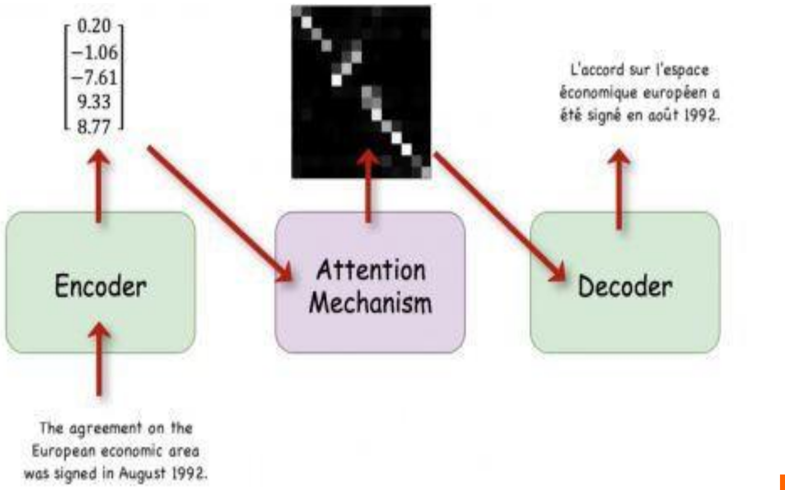
(1) which sentence is crucial in classifying the document and

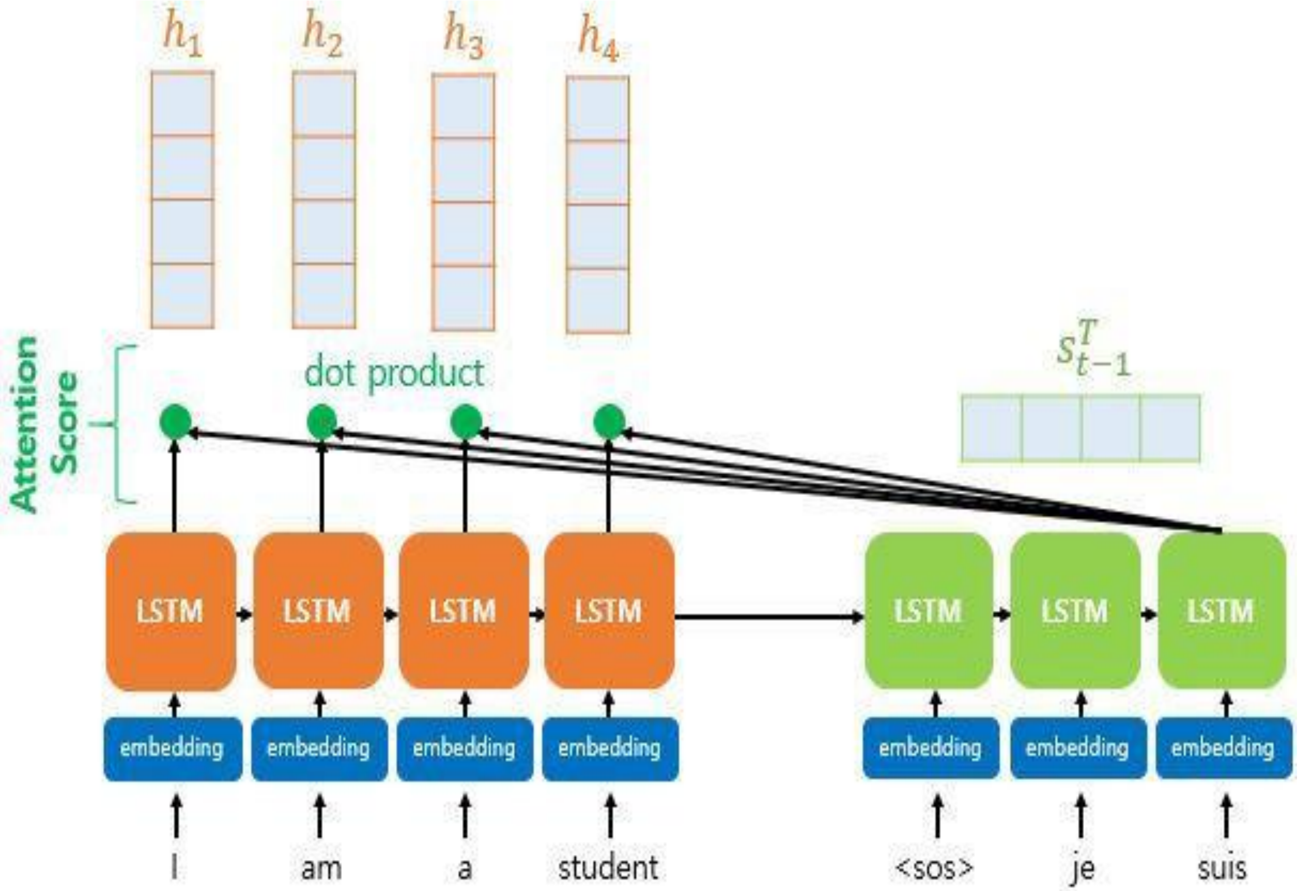
(2) which part of the sentence, i.e., which words, are salient in that sentence.

Attention: Summary

Attention removes bottleneck of Encoder-Decoder model

- Pay selective attention to relevant parts of input
- Compute weighted sum of all hidden states (called attention weight). Use attention weights while decoding
- Attention takes care of alignment while translating


0.20
−1.06
−7.61
9.33
8.77
L'accord sur l'espace économique européen a été signé en août 1992.
Encoder
Attention Mechanism
Decoder
The agreement on the European economic area was signed in August 1992.



Transformers

Transformer is a deep learning model for sequence to sequence learning.

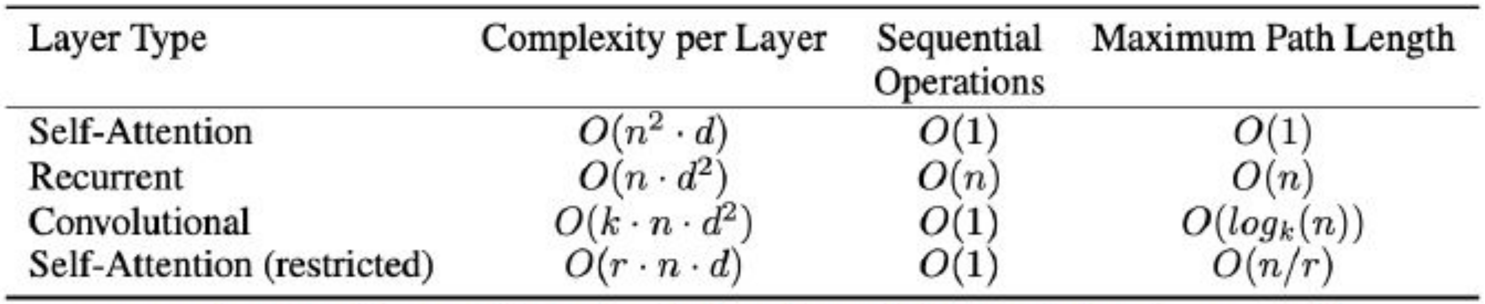
It was first proposed in the paper “**Attention is all you need**” by Vaswani et al. (Advances in Neural Information Processing Systems (NIPS), Volume 30, 2017)

Transformer is an architecture for transforming one sequence into another one with the help of two parts (Encoder and Decoder), but it differs from the previously described/existing sequence-to-sequence models because it does not imply any Recurrent Networks (GRU, LSTM, etc.).

Transformer – Core Idea

- Motivation:
  - Prior works use RNN/CNN to solve **sequence-to-sequence** problems
  - **Attention** already handles *arbitrary length* of sequences, *easy to parallelize*, and not suffer from *forgetting* problems… Why should one use RNN/CNN modules?

- Idea:
  - Design architecture **only using** attention modules
  - To extract features, the authors use self-attention, that features **attend on itself**
    - Self-attention has many advantages over RNN/CNN blocks



Layer Type	Complexity per Layer	Sequential Operations	Maximum Path Length
Self-Attention	$O(n^2 \cdot d)$	$O(1)$	$O(1)$
Recurrent	$O(n \cdot d^2)$	$O(n)$	$O(n)$
Convolutional	$O(k \cdot n \cdot d^2)$	$O(1)$	$O(log_k(n))$
Self-Attention (restricted)	$O(r \cdot n \cdot d)$	$O(1)$	$O(n/r)$

n: sequence length, d: feature dimension, k: (conv) kernel size, r: window size to consider

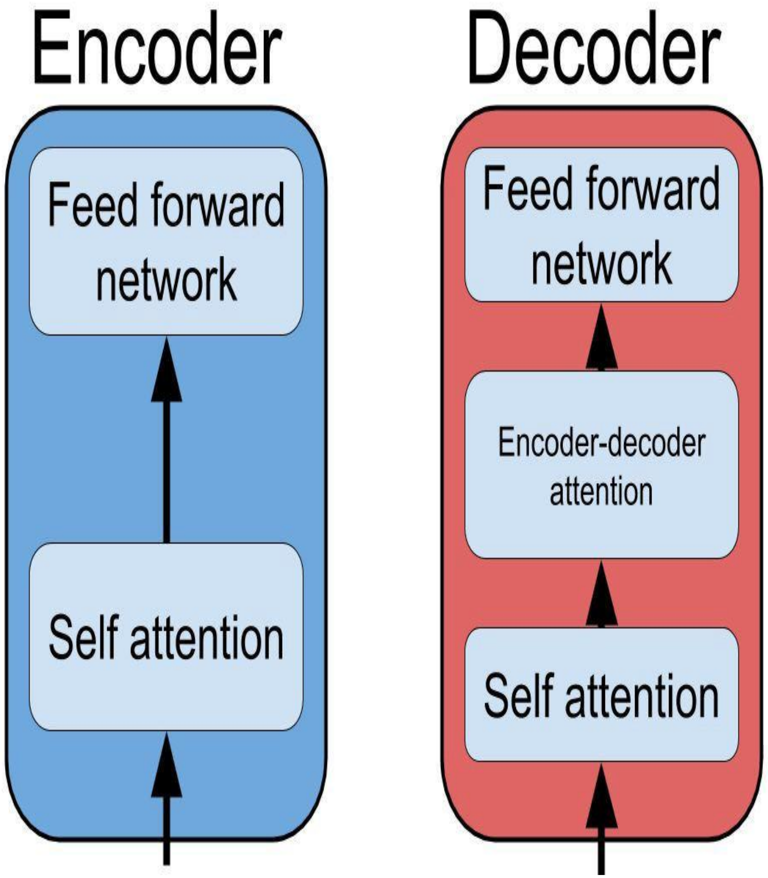
Maximum path length: maximum traversal between any two input/outputs **(lower is better)**

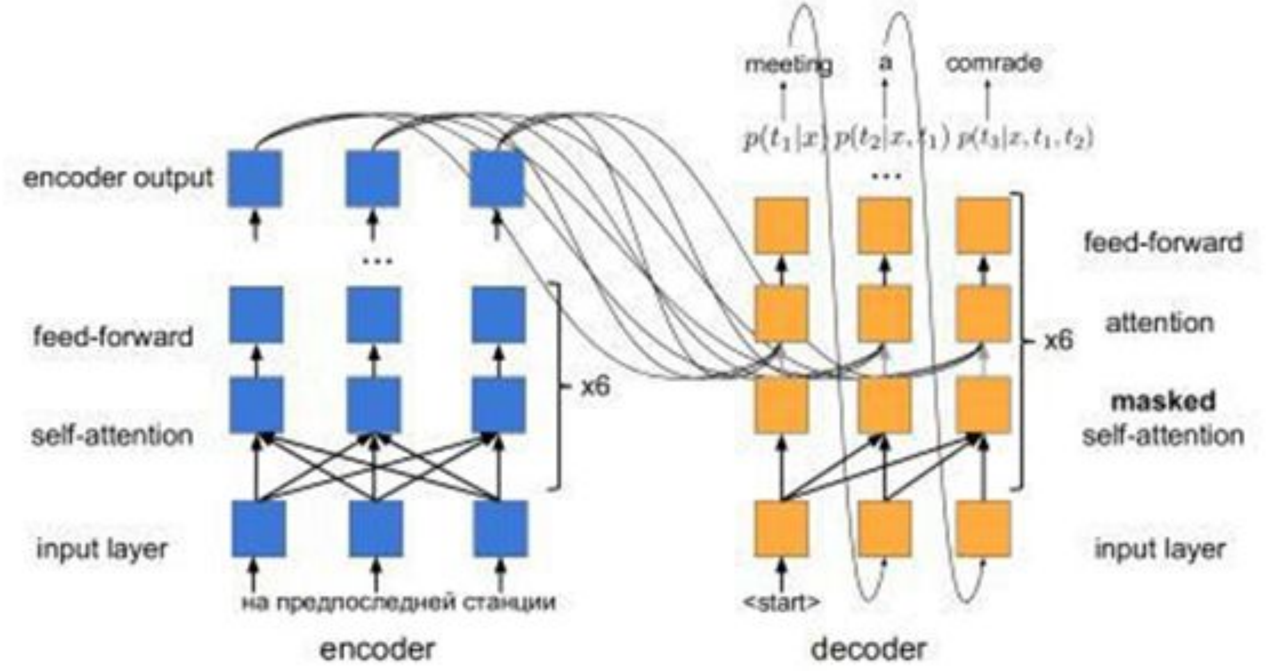
Transformer – Core Idea

The **multi-head self-attention** layer in **Transformer** aligns words in a sequence with other words in the sequence, thereby calculating a representation of the sequence.

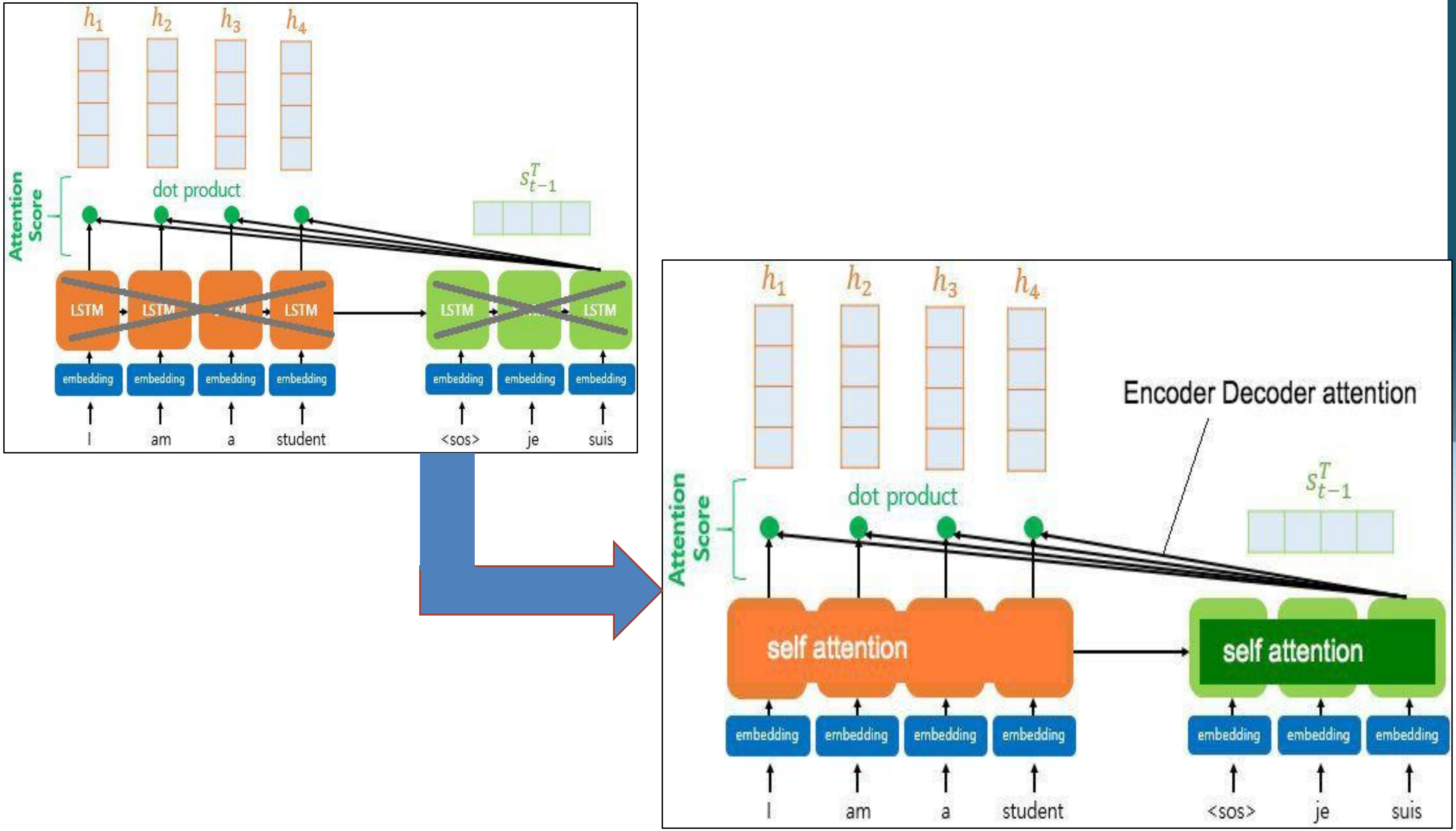
It is not only more effective in representation, but also more computationally efficient compared to convolution and recursive operations.

Transformer





Transformers: Get rid of RNNs completely



Transformer – Details

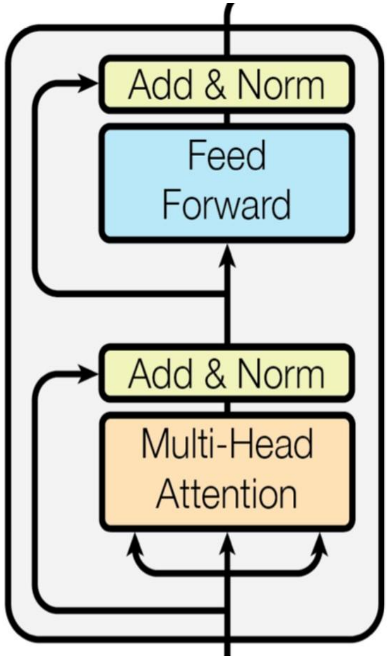
Encoder

A stack of N=6 identical layers.
All layers and sublayers are 512-dimensional

Each layer consists of two sublayers

- one multi-headed self attention layer
- one position-wise fully connected layer

Each sublayer has a residual connection and is normalized:
LayerNorm(x + Sublayer(x))



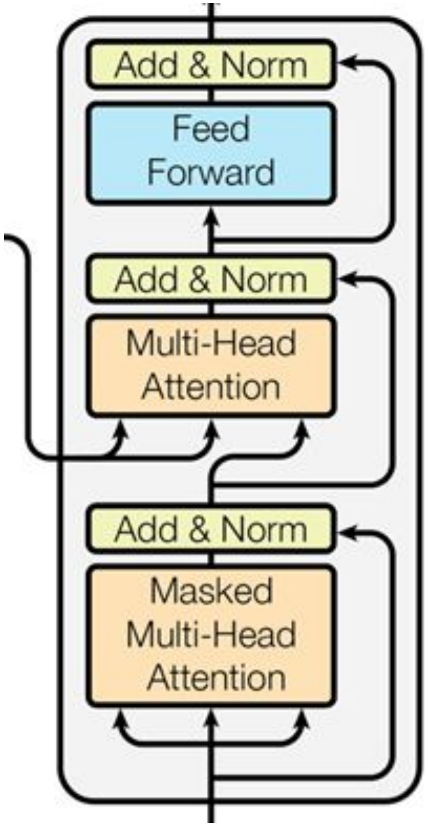
Decoder

A stack of N=6 identical layers
All layers and sublayers are 512-d

Each layer consists of three sublayers:

- one multi-headed self attention layer over decoder output (ignoring future tokens)
- one multi-headed attention layer over encoder output
- one position-wise fully connected layer

Each sublayer has a residual connection and is normalized:
LayerNorm(x + Sublayer(x))



Transformer

The final **transformer** model is built upon the (multi-head) attention blocks

- First, extract features with **self-attention** (lower part of the block)

- Then decode feature with usual **attention** (middle part of the block)

- Since the model does not have a sequential structure, the authors give **position embedding** (handcrafted features that represent the location in sequence)

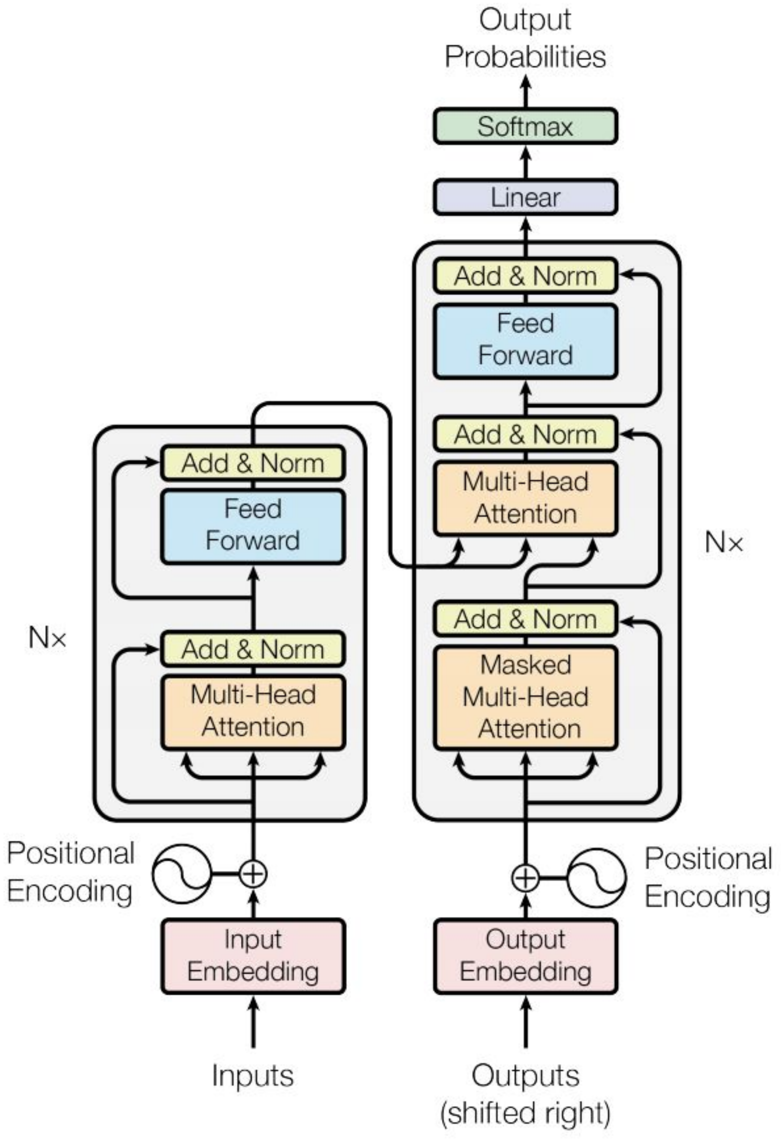


Figure 1: The Transformer - model architecture.

Transfer learning in NLP

Transfer learning is a method where a model developed for a task is reused as the starting point for a model on a second related task.

It is a popular approach in deep learning where pre-trained models are used as the starting point on computer vision and natural language processing tasks given the vast compute and time resources required to develop neural network models on these problems and from the huge jumps in skill that they provide on related problems.

Paraphrased from https://machinelearningmastery.com/transfer-learning-for-deep-learning/

Transfer learning using neural networks

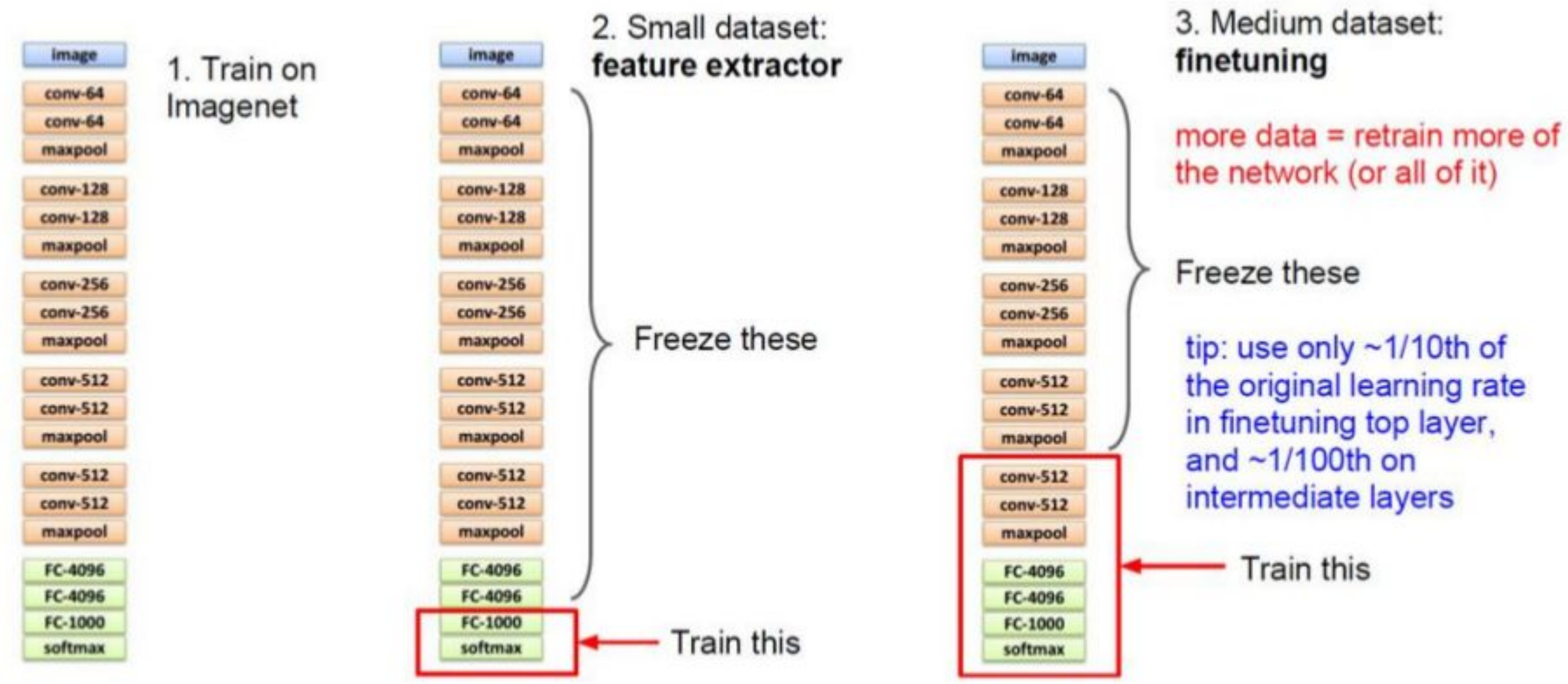
Typically, the neural network trained on a large corpus of data is made available publicly. This is referred to as the **pre-trained** model.

The pre-trained model is then used as the initialized neural network model for the user's task (which is similar to the task from which the pre-trained model was obtained).

This neural network is now trained using the user's data on the user's task. Typically, many weights are kept unchanged and only those in a small portion of the network (typically closer to the output layer) are changed in the training.

Transfer learning has been used commonly in Computer Vision and a number of pre-trained models are available.

Illustration of transfer learning in image classification



1. Train on Imagenet

2. Small dataset: **feature extractor**

Freeze these

Train this

3. Medium dataset: **finetuning**

more data = retrain more of the network (or all of it)

Freeze these

tip: use only ~1/10th of the original learning rate in finetuning top layer, and ~1/100th on intermediate layers

Train this

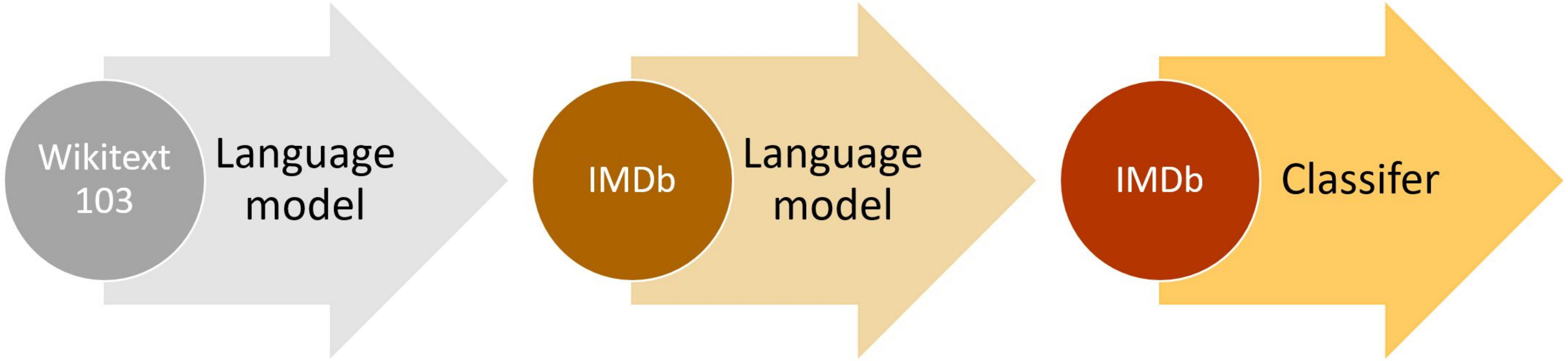
Transfer learning in NLP

Unlike in Computer Vision, in Natural Language Processing (NLP), pre-trained models only became widely available recently.

- A major development was Universal Language Model Fine-Tuning (ULMFiT) in 2018. This set the base for transfer learning for NLP and paved the way for ELMo, GPT, GPT-2, BERT and XLNet.

- The release of Bidirectional Encoder Representations from Transformers (BERT) in 2018 acted as a catalyst in many respects..
  - Two pre-trained models of BERT are available, namely base (110 million parameters) and large (345 million parameters).

ULM Fit: Transfer + Predict



Wikitext 103

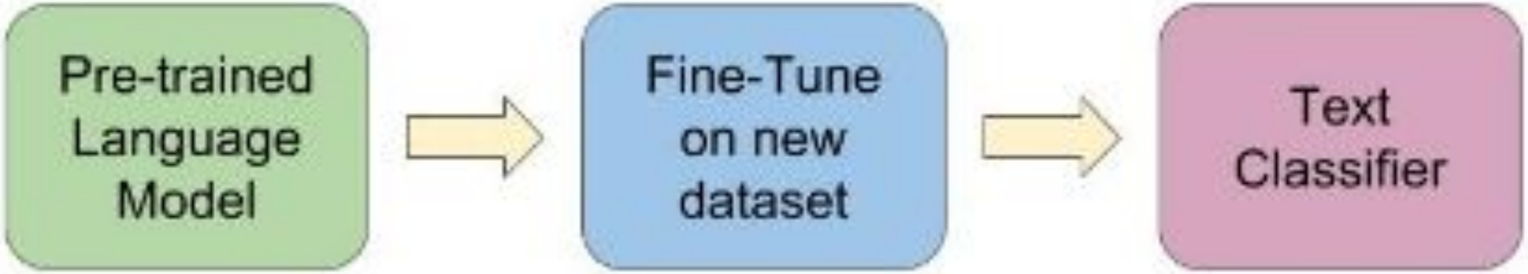
Language model

IMDb

Language model

IMDb

Classifer



Pre-trained Language Model

Fine-Tune on new dataset

Text Classifier

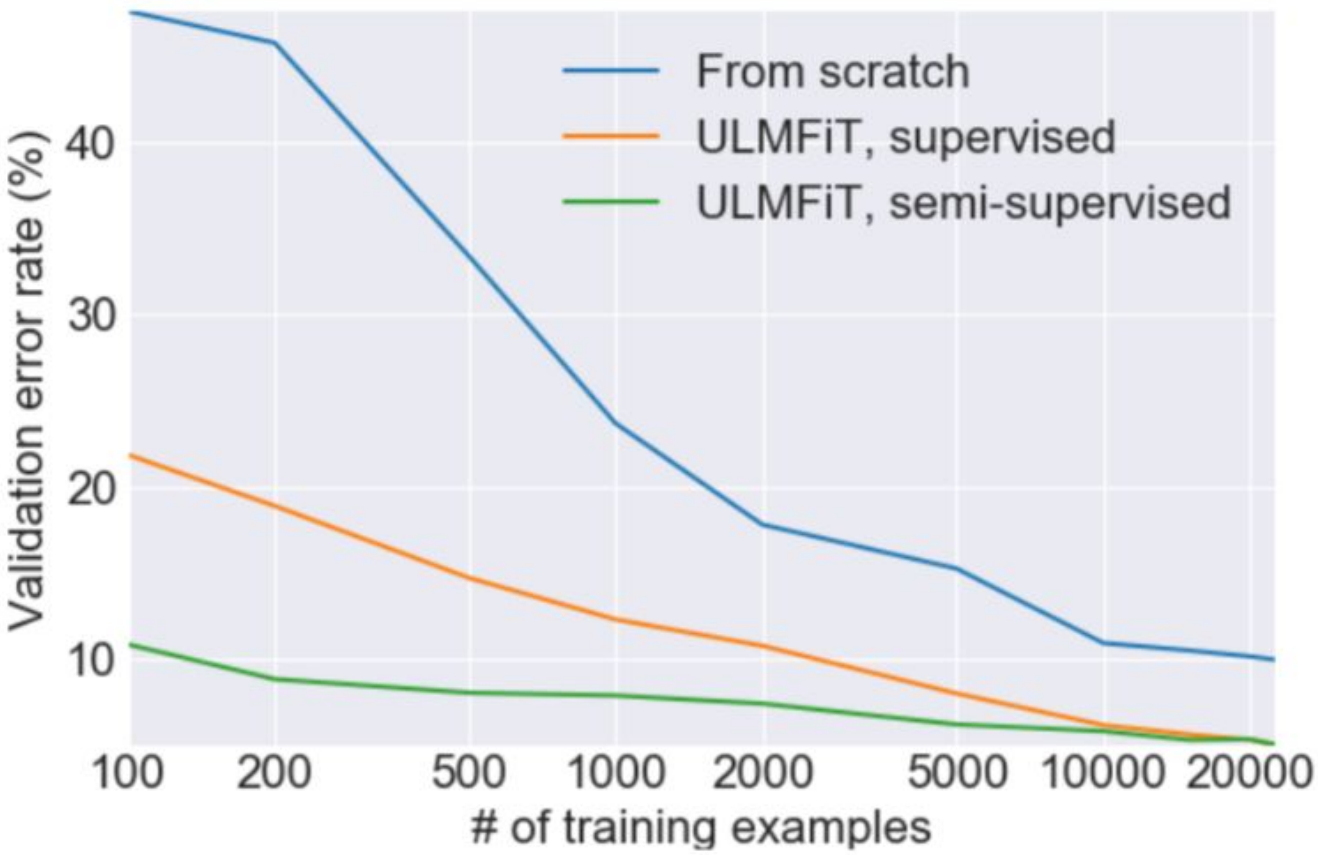
ULM Fit

Base model : AWD-LSTM (Merity et al., 2017a), a regular LSTM (with no attention, short-cut connections, or other sophisticated additions).

Hacks exploited

Dropouts, variational length back propagation, learning rate differentials

Advantages of ULM Fit



ULMFiT-based models (which have been pre-trained) perform very well even on small and medium datasets compared to models trained from scratch on the corresponding dataset.

This is because they have already captured the properties of the language during pre-training.

Image source : https://nlp.fast.ai/images/ulmfit_imdb.png

BERT and GPT

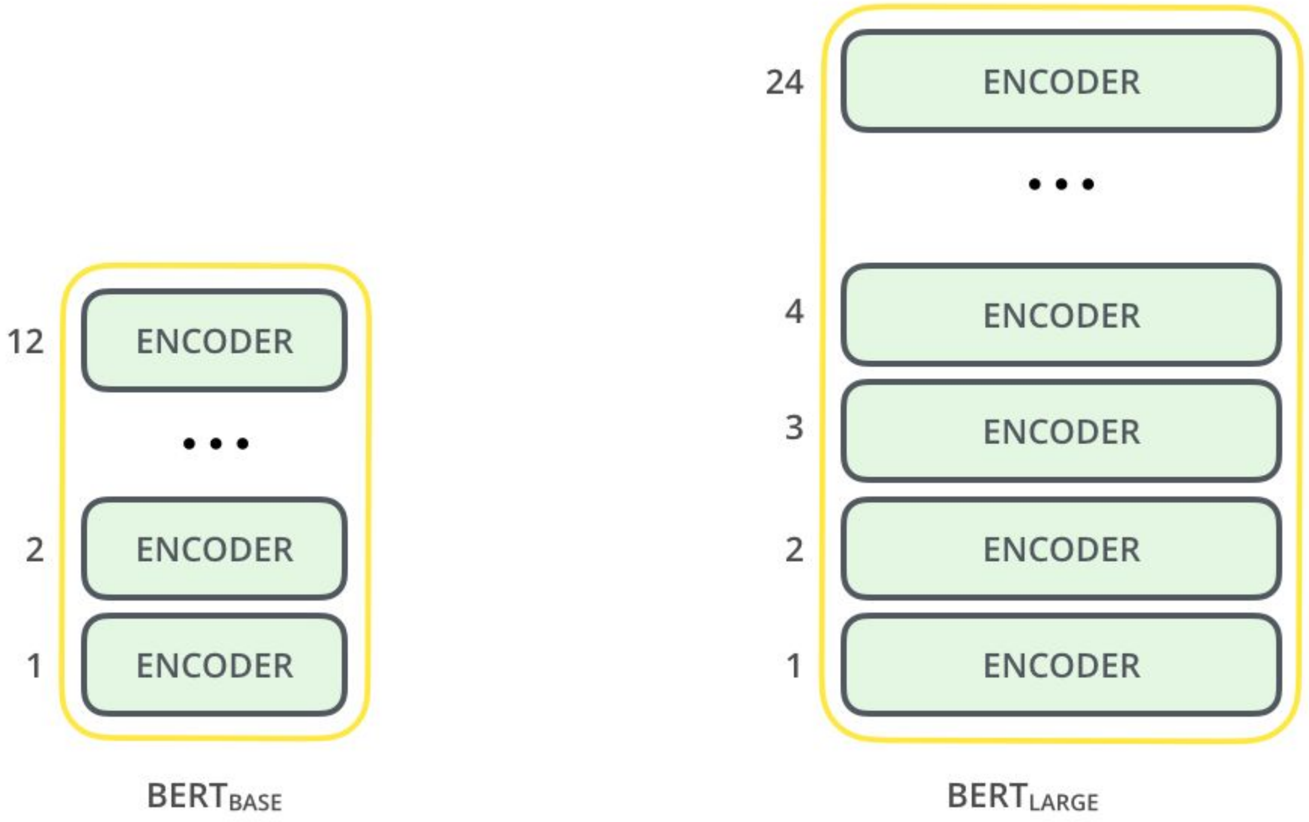
Bidirectional **E**ncoder **R**epresentations from **T**ransformers (BERT)

BERT was published in 2018 by Jacob Devlin and his colleagues at Google AI Language.

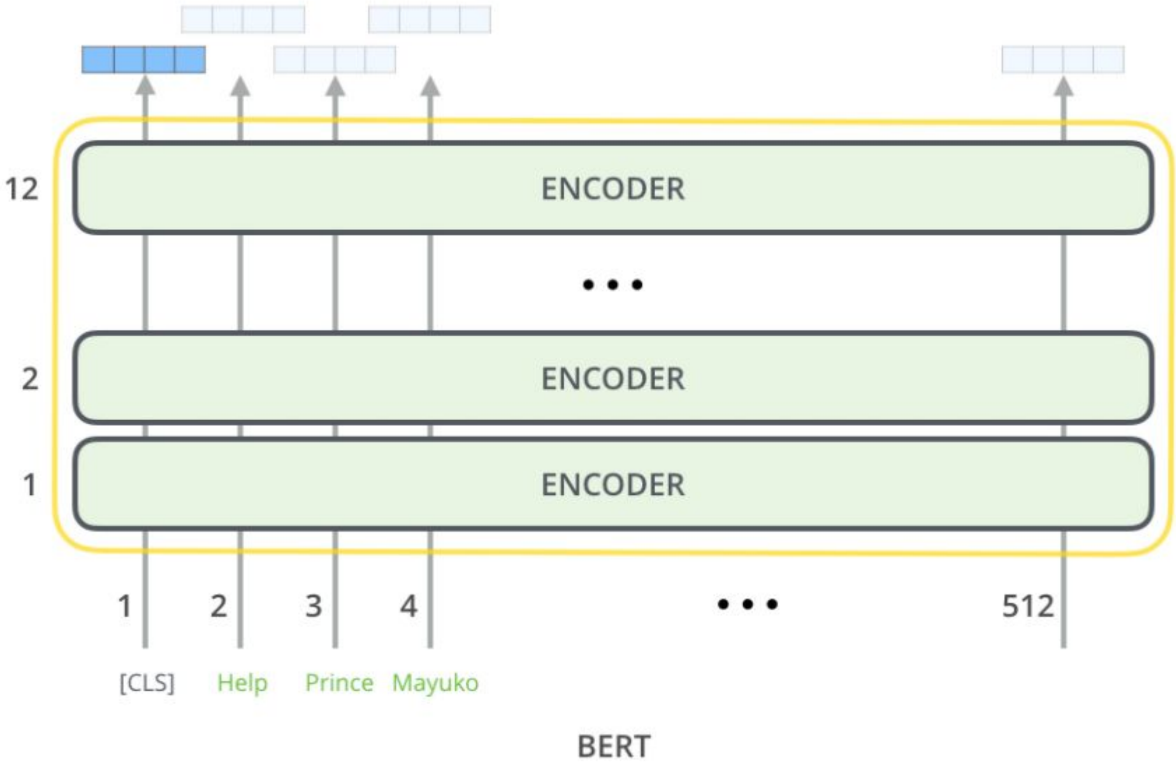
It generated a lot of interest in NLP community, since it yielded state-of-the-art results in a wide variety of NLP tasks, including Question Answering (SQuAD v1.1), Natural Language Inference (MNLI), and others.

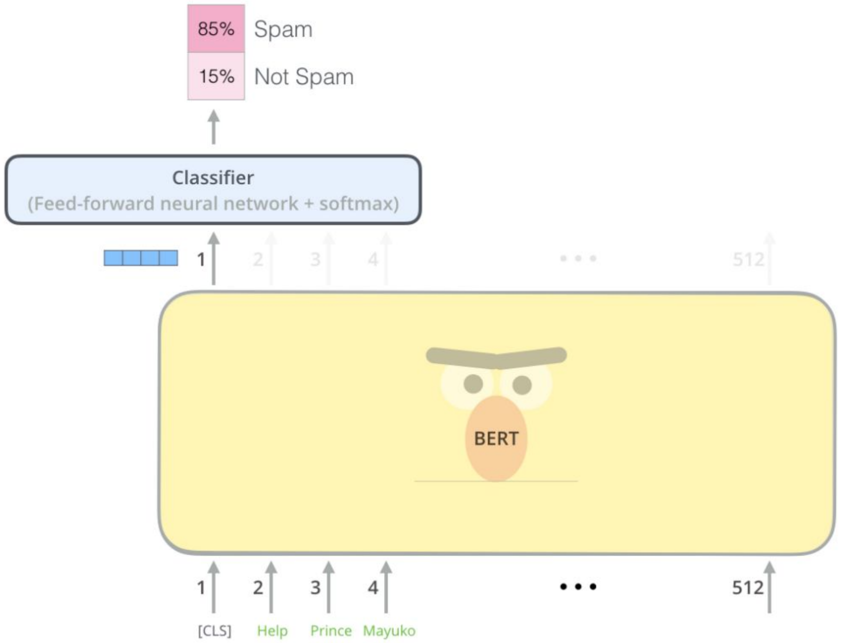
Two models of BERT are available, namely base (110 million parameters) and large (345 million parameters).

BERT builds on top of a number of clever ideas that have been bubbling up in the NLP community recently – including but not limited to Semi-supervised Sequence Learning (by Andrew Dai and Quoc Le), ELMo (by Matthew Peters and researchers from AI2 and UW CSE), ULMFiT (by fast.ai founder Jeremy Howard and Sebastian Ruder), the OpenAI transformer (by OpenAI researchers Radford, Narasimhan, Salimans, and Sutskever), and the Transformer (Vaswani et al).



How BERT Works





Each position outputs a vector of size *hidden_size* (768 in BERT Base).

Why BERT works?

- Leveraging huge unlabeled and high quality data: 7000 books + Wikipedia (together 3300M words)

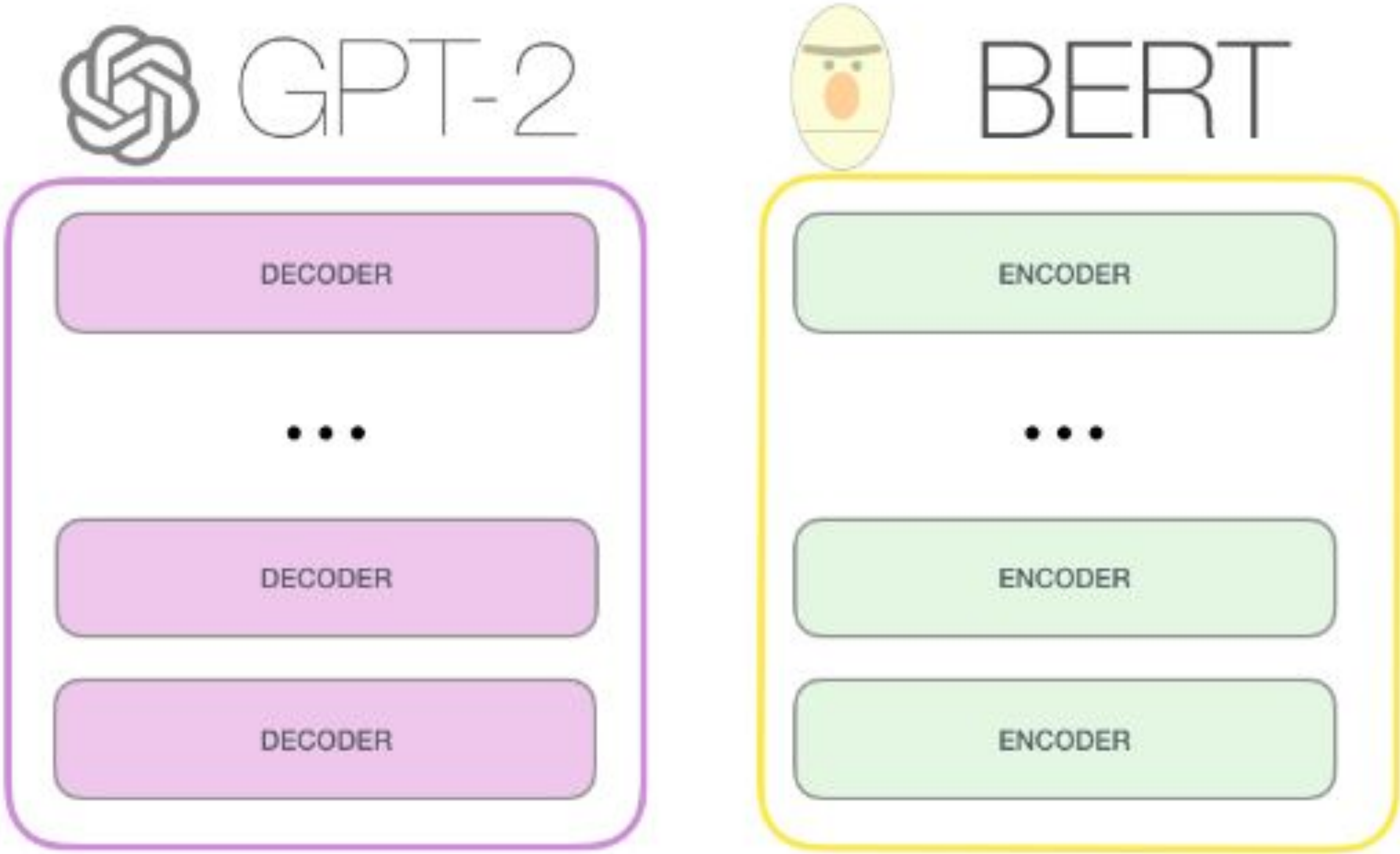
- Multi-head self-attention blocks in Transformer:
    - modelling the intra- and extra- word-word relations
    - parallelable within instance and thus efficient

- Task similarity: masked language modelling + next sentence prediction

GPT and BERT

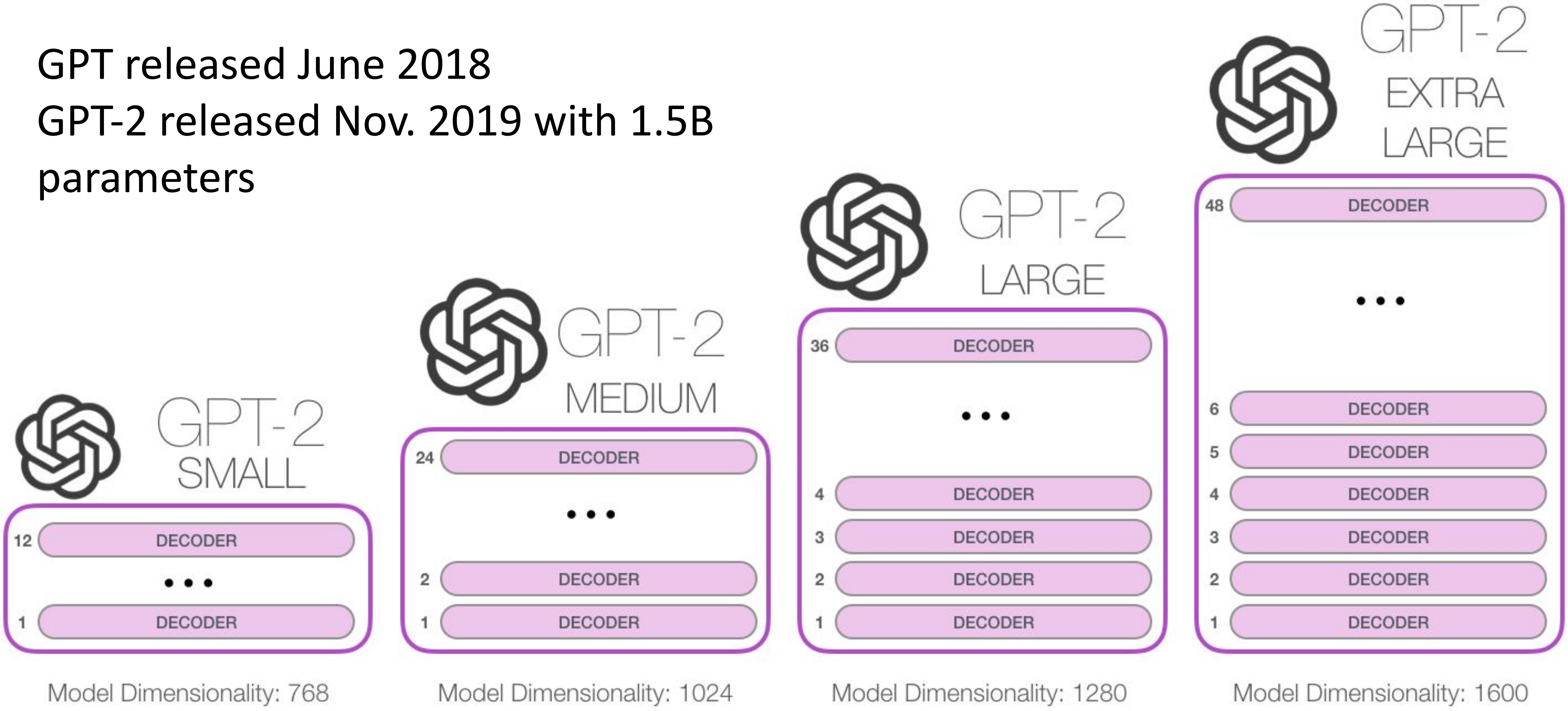
1. A transformer uses Encoder stack to model input, and uses Decoder stack to model output (using input information from encoder side).
2. But if we do not have input, we just want to model the “next word”, we can get rid of the Encoder side of a transformer and output “next word” one by one. This gives us **GPT (Generative Pre-trained transformer)**.
3. If we are only interested in training a language model for the input for some other tasks, then we do not need the Decoder of the transformer, that gives us **BERT**.

GPT and BERT


GPT-2
BERT
DECODER
• • •
DECODER
DECODER
ENCODER
• • •
ENCODER
ENCODER

GPT released June 2018

GPT-2 released Nov. 2019 with 1.5B parameters



117M parameters

345M

762M

1542M

A few sample applications

A story generated by GPT-2

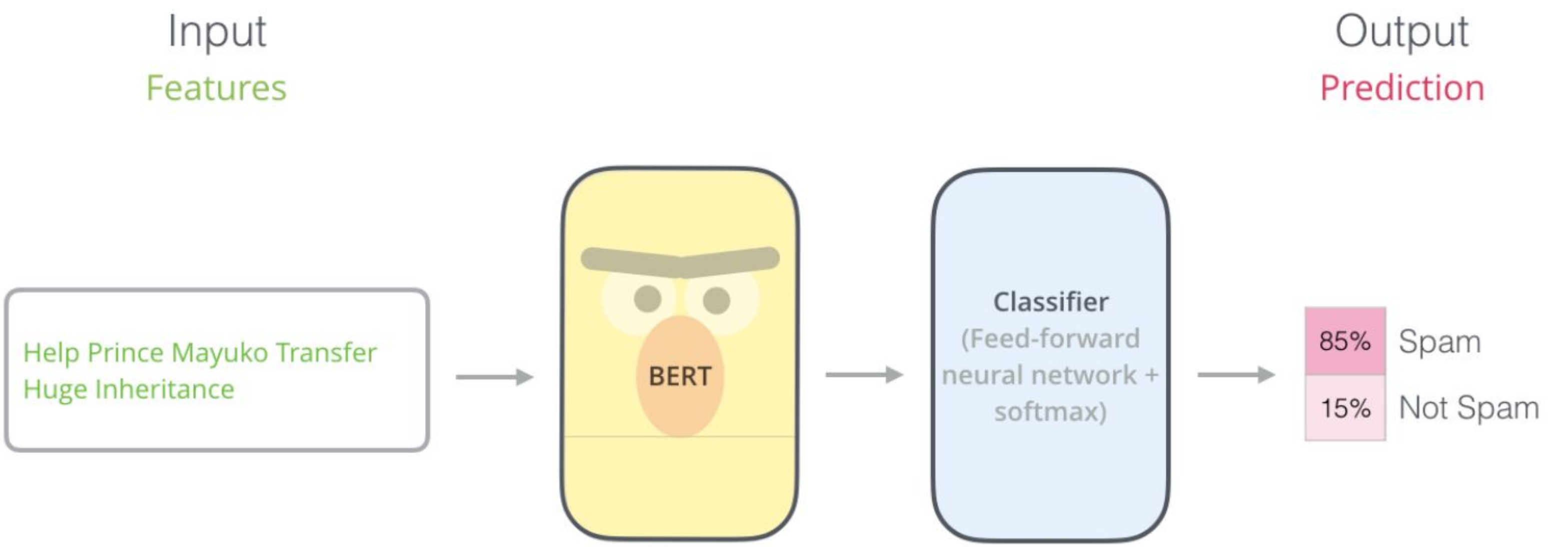
“The scientist named the population, after their distinctive horn, Ovid’s Unicorn. These four-horned, silver-white unicorns were previously unknown to science.

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.

Pérez and the others then ventured further into the valley. `By the time we reached the top of one peak, the water looked blue, with some crystals on top,’ said Pérez.

Pérez and his friends were astonished to see the unicorn herd. These creatures could be seen from the air without having to move too much to see them – they were so close they could touch their horns."

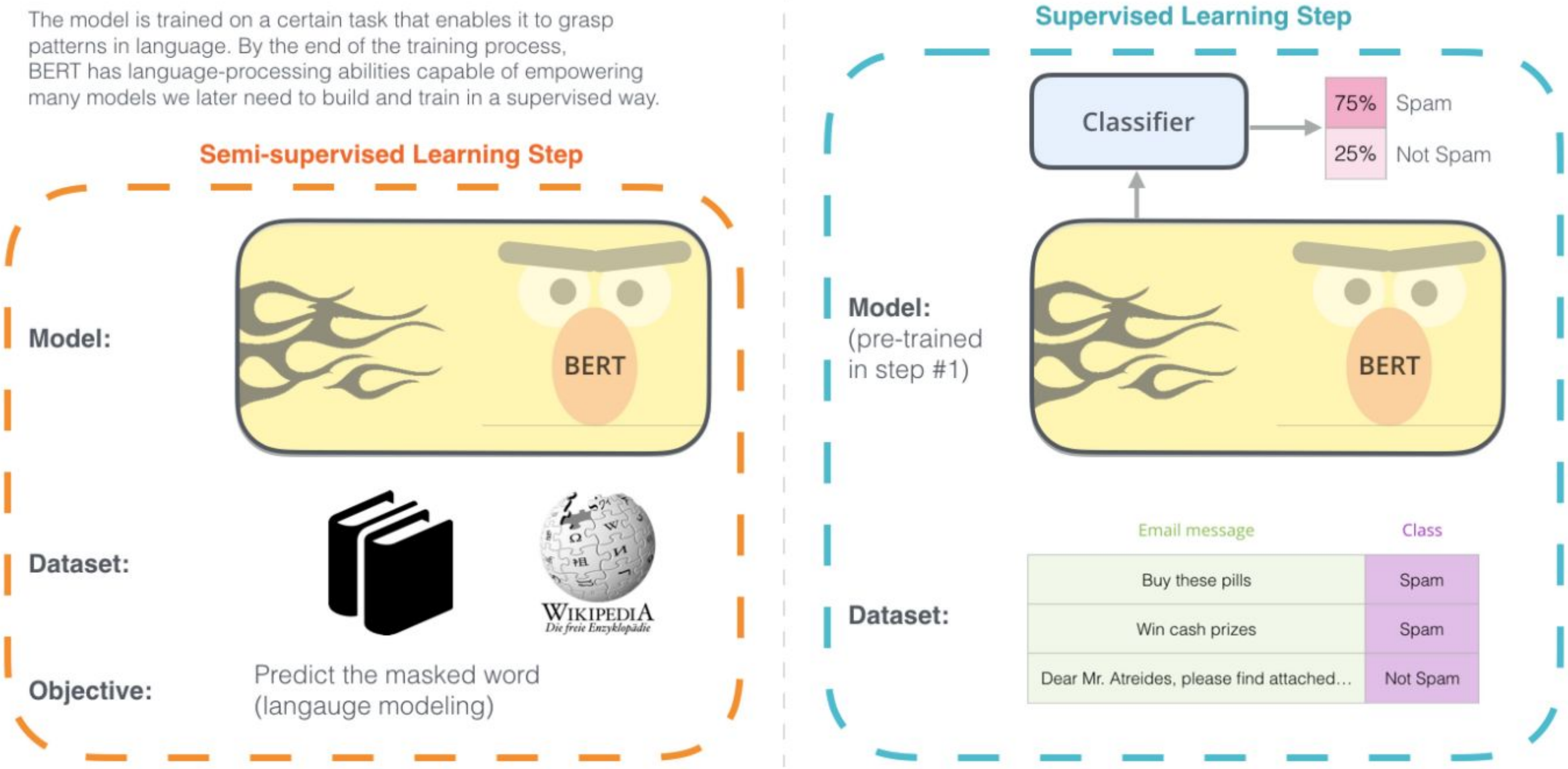

Input
Features
Output
Prediction
Help Prince Mayuko Transfer Huge Inheritance
BERT
Classifier
(Feed-forward neural network + softmax)
85%
Spam
15%
Not Spam

- **Sentiment analysis**
  - Input: Movie/Product review. Output: is the review positive or negative?
  - Example dataset: SST
- **Fact-checking**
  - Input: sentence. Output: “Claim” or “Not Claim”
  - More ambitious/futuristic example:
    - Input: Claim sentence. Output: “True” or “False”

1 - Semi-supervised training on large amounts of text (books, wikipedia..etc).

The model is trained on a certain task that enables it to grasp patterns in language. By the end of the training process, BERT has language-processing abilities capable of empowering many models we later need to build and train in a supervised way.

2 - Supervised training on a specific task with a labeled dataset.

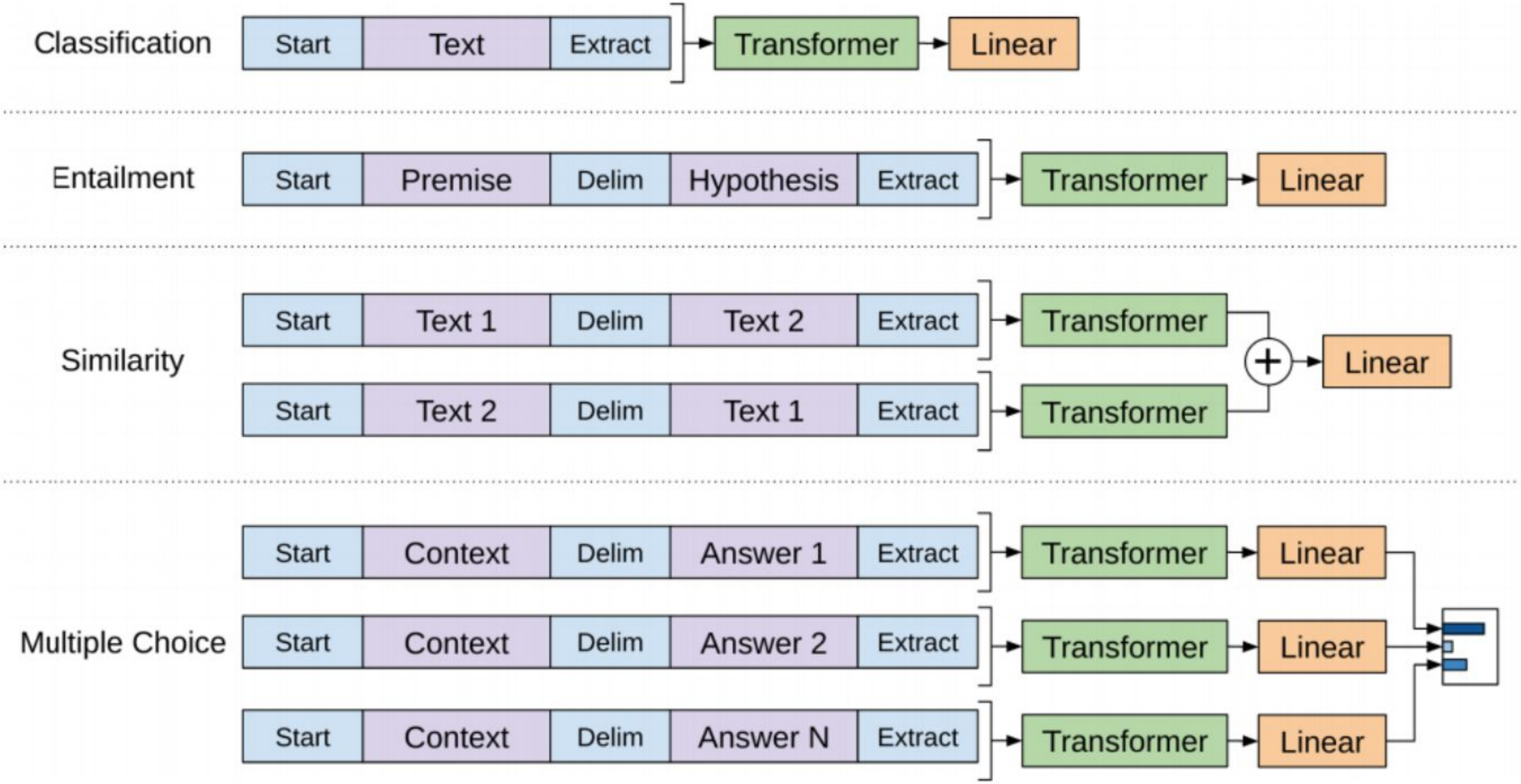


Email message	Class
Buy these pills	Spam
Win cash prizes	Spam
Dear Mr. Atreides, please find attached…	Not Spam

The two steps of how BERT is developed. You can download the model pre-trained in step 1 (trained on un-annotated data)

http://jalammar.github.io/illustrated-bert/

Use of Transformers for Multiple Applications



References

- **Attention is all you need**,
  Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin,
  Advances in Neural Information Processing Systems (NIPS), Volume 30, 2017

- **BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding,**
  Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova, 2018

https://arxiv.org/abs/1810.04805

- **Effective Approaches to Attention-based Neural Machine Translation**, Minh-Thang Luong, Hieu Pham, Christopher D. Manning https://arxiv.org/abs/1508.04025

- **Universal language model fine-tuning for text classification**,

  J. Howeard, S. Ruder., 2018

- **Neural Machine Translation by Jointly Learning to Align and Translate**

Dzmitry Bahdanau, Kyunghyun Cho, Yoshua Bengio, https://arxiv.org/abs/1409.0473, 2015

Additional links

https://en.wikipedia.org/wiki/BERT_(language_model)

BERT implementation : https://github.com/google-research/bert

Hierarchical Attention Networks for document classification, Yang et al., https://www.aclweb.org/anthology/N16-1174.pdf

Transformers in NLP, BERT, RoBERTa & GPT-3

https://towardsdatascience.com/transformers-in-nlp-7c164291326d

OpenAI GPT-2 implementation: https://github.com/openai/gpt-2

ELMo paper: M. Peters, et al, Deep contextualized word representation, 2018

Jay Alammar, The illustrated GPT-2, https://jalammar.github.io/illustrated-gpt2/

INSOFE

INSOFE's Vision

The BEST GLOBAL DESTINATION for individuals and organizations to learn and adopt disruptive technologies for solving business and society's challenges.

GLASGOW
TROY
OTTAWA
RENNES
CLEVELAND
ROPAR
MUMBAI
BENGALURU

INSOFE - HYDERABAD
2nd Floor, Jyothi Imperial, Vamsiram Builders
Janardana Hills, Gachibowli
Hyderabad – 500032
☎ +91 93199 77257

INSOFE - BENGALURU
Floors 1-3, L77, 15th Cross Road
Sector 6, HSR Layout
Bengaluru - 560102
☎ +91 93199 77267

INSOFE - MUMBAI
4th Floor - A Wing, Spaces - Kanaki
Andheri-Kurla Road, Chakala
Andheri East, Mumbai - 400093
☎ +91 93199 77269

20+
PhDs

30+
Doctoral students

75+
Patents

300+
Publications

10000+
Data Science Alumni