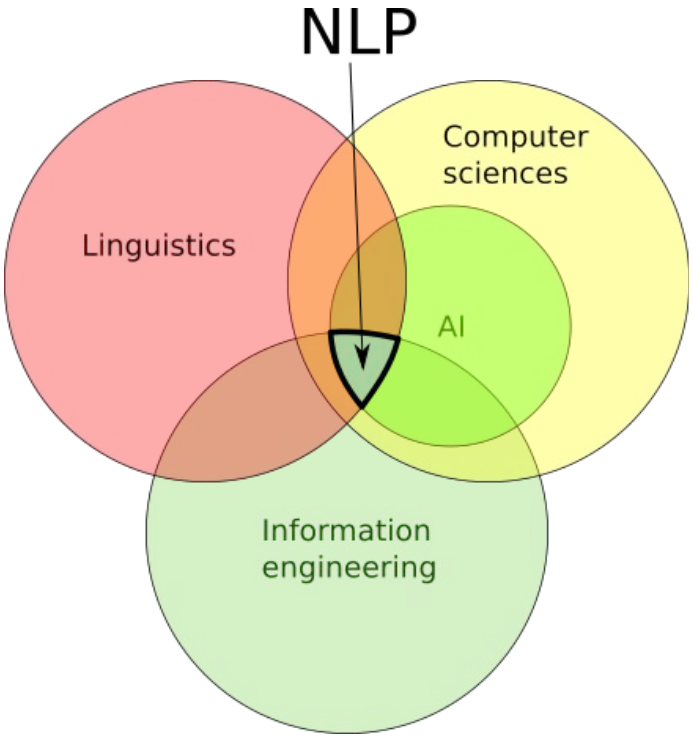
From word embeddings to transformers

Laurence Jennings

29/5/2020, Strasbourg

Natural Language Processing



Natural Language Processing - how?

Creating algorithms that understand natural language

J. R. Firth “**You shall know a word by the company it keeps”**

The probability of the i^{th} word in a sentence is a function of the i-1 words preceding it (the context)

Represent words in a way that can be treated mathematically (**word embeddings**)

Feed massive (billions of words) corpora to these algorithms to train them.

Fine tune pretrained algorithms on specific tasks (**pretrained transformers)**

Natural Language Processing - why?

Facilitate daily interaction with technology:

- TTS STT
- Improved search engine results
- Suggestions while typing on your phone or gmail

Within our lifetimes:

- Real time translation to have seamless conversations in two different languages
- Lifelike personal assistants?

Services

- Specialized Information retrieval platforms
  - Medical
  - Legal
- Marketing
  - Sentiment analysis
- Client support
  - Support chatbots

GPT-2 text generation

MODEL COMPLETION (MACHINE-WRITTEN, 10 TRIES)

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.

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

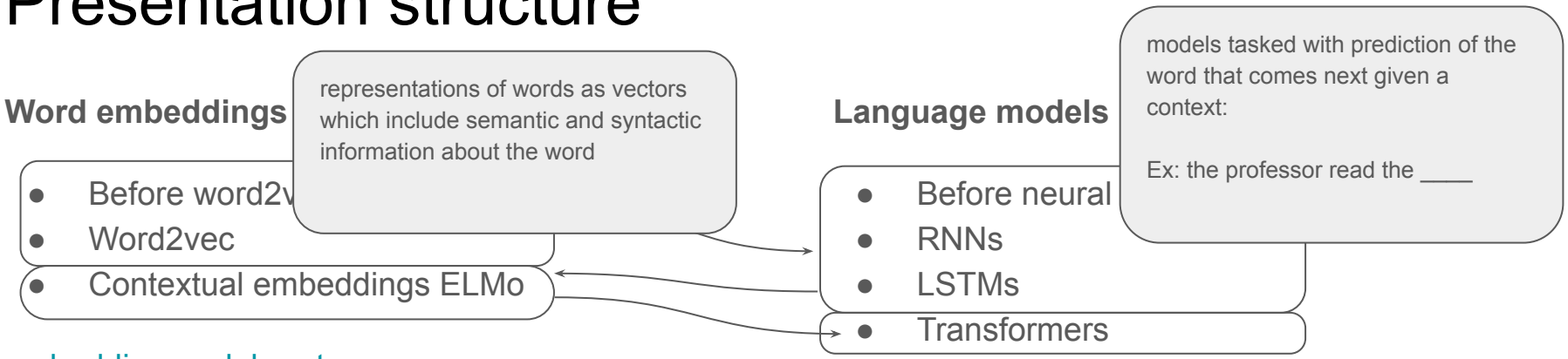
Now, after almost two centuries, the mystery of what sparked this odd phenomenon is finally solved.

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.

[...]

However, Pérez also pointed out that it is likely that the only way of knowing for sure if unicorns are indeed the descendants of a lost alien race is through DNA. "But they seem to be able to communicate in English quite well, which I believe is a sign of evolution, or at least a change in social

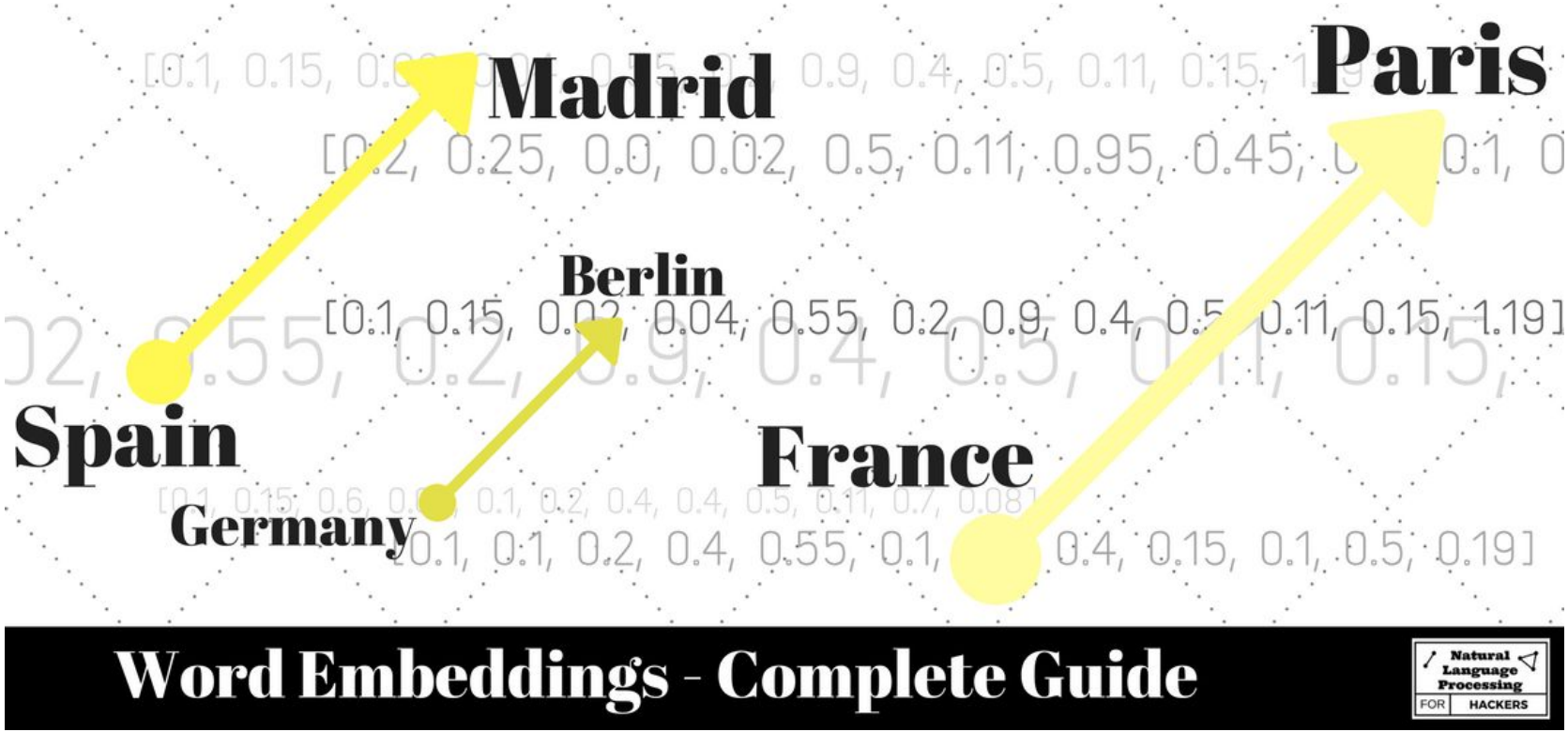
Presentation structure



[embeddings colaboratory]

[Language models colaboratory]

Word embeddings



https://nlpforhackers.io/word-embeddings/

Setting the stage

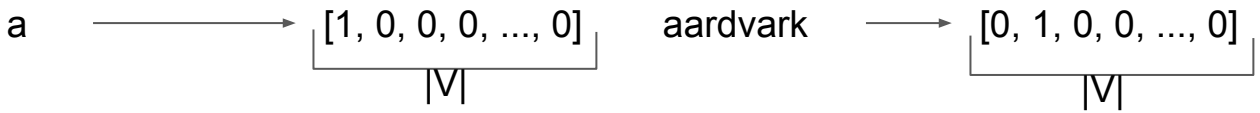
Mikolov 2010:

Language models for real-world speech recognition or machine translation systems are built on huge amounts of data, and **popular belief says that more data is all we need**.

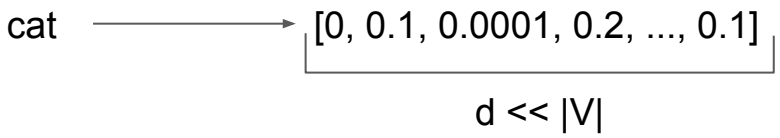
Models coming from research tend to be **complex** and often work well only for systems based on very **limited amounts of training data**. In fact, most of the proposed advanced language modeling techniques provide only tiny improvements over simple baselines, and are rarely used in practice.

Mikolov et al. Recurrent neural network based language model 2010

Word embeddings



Word embeddings introduced in 1986 by Rumelhart et al. [1]



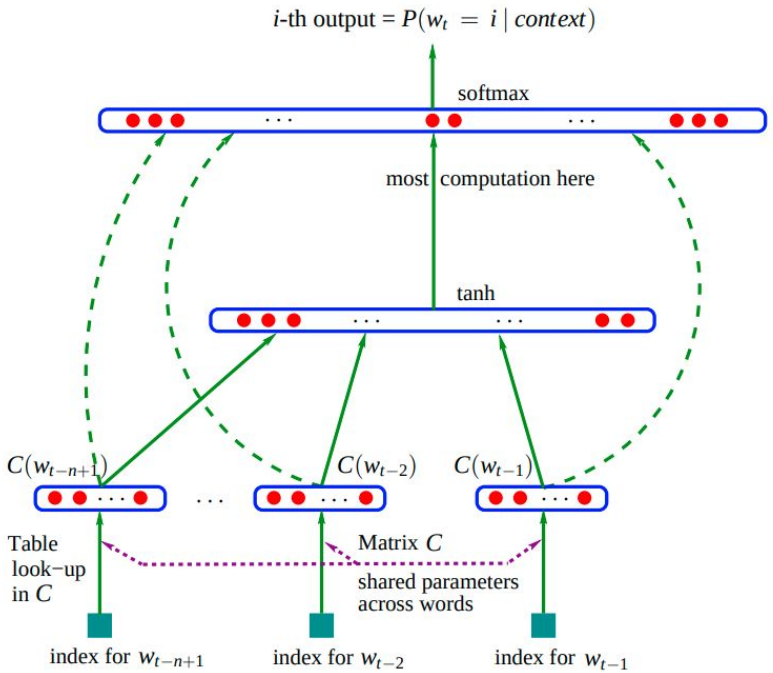
A good example of word embeddings before the introduction of word2vec was Bengio's Neural Network Language Model (NNLM) [2]

1. G.E. Hinton, J.L. McClelland, D.E. Rumelhart. Distributed representations. In: Parallel distributed processing: Explorations in the microstructure of cognition. 1986
2. Y. Bengio, R. Ducharme, P. Vincent. A neural probabilistic language model. 2003

NNLM - 2003

The word embeddings are learned together with a language model.

This is done by passing the word embeddings through a hidden layer resulting in a computationally expensive model



1. Y. Bengio, R. Ducharme, P. Vincent. A neural probabilistic language model. 2003

Word2vec - 2013

Model made with the sole objective of learning high quality word embeddings

Two models are presented:

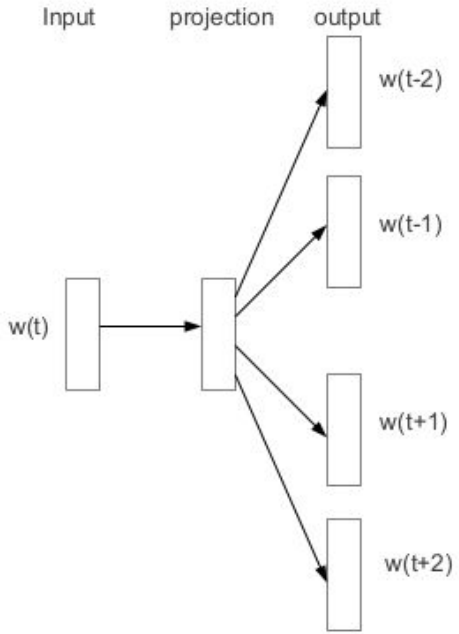
- Continuous bag of words

- Skip-gram

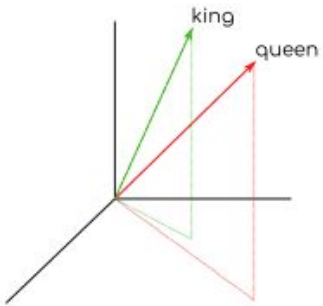
Tomas Mikolov et al. Distributed Representations of Words and Phrases and their Compositionality - 2013

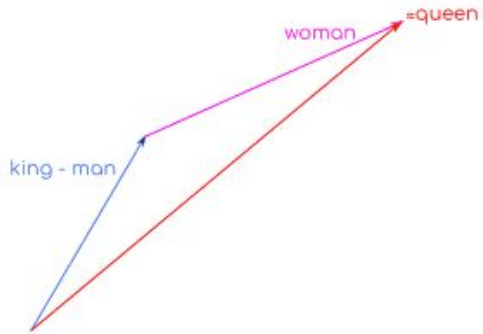
Skip Gram model

Predict context from input word



$$p(w_O|w_I) = \frac{\exp\left({v'_{w_O}}^{\top} v_{w_I}\right)}{\sum_{w=1}^{W} \exp\left({v'_w}^{\top} v_{w_I}\right)}$$





Tomas Mikolov et al. Distributed Representations of Words and Phrases and their Compositionality - 2013

Capital cities

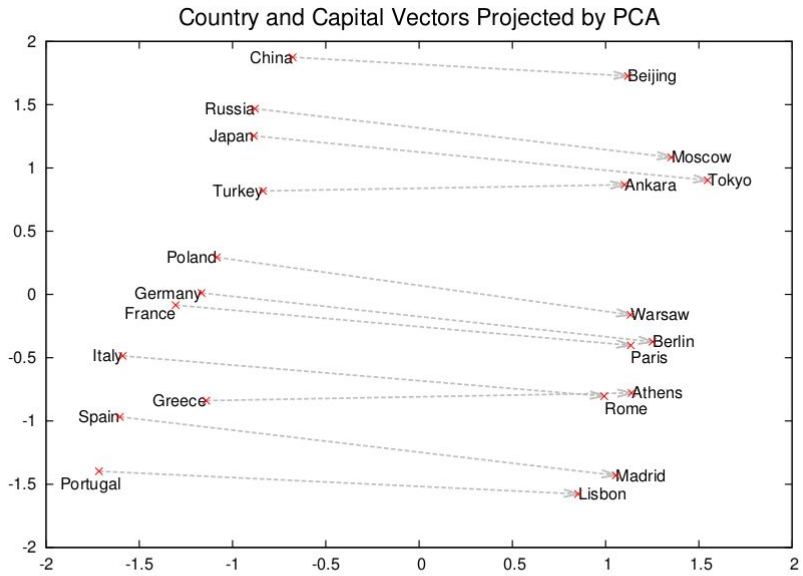


Figure 2: Two-dimensional PCA projection of the 1000-dimensional Skip-gram vectors of countries and their capital cities. The figure illustrates ability of the model to automatically organize concepts and learn implicitly the relationships between them, as during the training we did not provide any supervised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

Tomas Mikolov et al. Distributed Representations of Words and Phrases and their Compositionality - 2013

Phrases

Newspapers			
New York	New York Times	Baltimore	Baltimore Sun
San Jose	San Jose Mercury News	Cincinnati	Cincinnati Enquirer
NHL Teams			
Boston	Boston Bruins	Montreal	Montreal Canadiens
Phoenix	Phoenix Coyotes	Nashville	Nashville Predators
NBA Teams			
Detroit	Detroit Pistons	Toronto	Toronto Raptors
Oakland	Golden State Warriors	Memphis	Memphis Grizzlies
Airlines			
Austria	Austrian Airlines	Spain	Spainair
Belgium	Brussels Airlines	Greece	Aegean Airlines
Company executives			
Steve Ballmer	Microsoft	Larry Page	Google
Samuel J. Palmisano	IBM	Werner Vogels	Amazon

Table 2: Examples of the analogical reasoning task for phrases (the full test set has 3218 examples). The goal is to compute the fourth phrase using the first three. Our best model achieved an accuracy of 72% on this dataset.

Tomas Mikolov et al. Distributed Representations of Words and Phrases and their Compositionality - 2013

Vector compositionality

Czech + currency	Vietnam + capital	German + airlines	Russian + river	French + actress
koruna	Hanoi	airline Lufthansa	Moscow	Juliette Binoche
Check crown	Ho Chi Minh City	carrier Lufthansa	Volga River	Vanessa Paradis
Polish zolty	Viet Nam	flag carrier Lufthansa	upriver	Charlotte Gainsbourg
CTK	Vietnamese	Lufthansa	Russia	Cecile De

Table 5: Vector compositionality using element-wise addition. Four closest tokens to the sum of two vectors are shown, using the best Skip-gram model.

Tomas Mikolov et al. Distributed Representations of Words and Phrases and their Compositionality - 2013

Word2vec demo

A few examples of what can be achieved using word vectors

embeddings colaboratory

Presentation structure

Word embeddings

- ✓ Before word2vec
- ✓ Word2vec
- Contextual embeddings ELMo

[embeddings colaboratory]

Language models

- Before neural networks
- RNNs
- LSTMs
- Transformers

[Language models colaboratory]

Language model

The professor read the _____

- Book
- Paper
- Chapter
- Newspaper
- …

$$P(x^{(t+1)}|x^{(t)}, ..., x^{(1)})$$

$$x^{(t)} \in V = \{w_1, ..., w_{|V|}\}$$

$$P(x^{(1)}, ..., x^{(T)}) = P(x^{(1)}) \times P(x^{(2)}|x^{(1)}) \times, ..., \times P(x^{(T)}|x^{(T-1)}, ..., x^{(1)}) =$$

$$= \prod_{t=1}^{T} P(x^{(t)}|x^{(t-1)}, ..., x^{(1)})$$

Language models

Generally speaking language models try to predict a word given a certain context

Statistical models:

- N-gram based models

Neural network based models:

- **RNNs, LSTMs, Transformers**

Statistical models - Backoff model

Based on the idea that sometimes less context is better:

If an n-gram (n>1) appears with sufficient frequency in the corpus we can trust it in our probability distribution for the prediction of the next word

Otherwise we fall back (backoff) to the (n-1)-gram.

Ex:

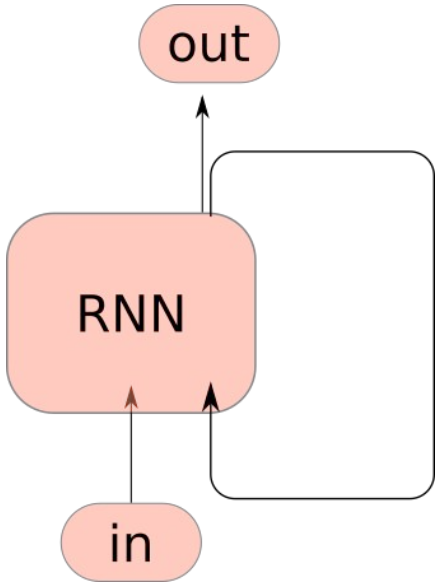
Next word after “New York City”: unless “New York City Rangers” has a certain frequency (an article about hockey) the model will use the trigram “New York City” or the bigram “New York” for its predictions

RNNs

Statistical models - very limited context (5-grams/7-grams)
NNLM - limited context due to fixed input size

“The goal of statistical language modeling is to predict the next word in textual data given context; thus we are dealing with sequential data prediction problem when constructing language models.” Mikolov et al. in 2010

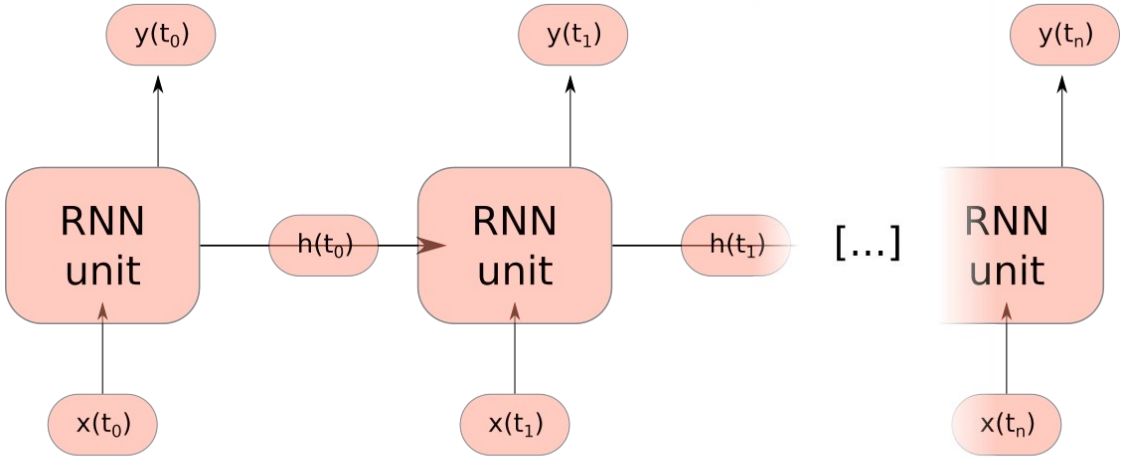
Enters the RNN.



Tomáš Mikolov et al. Recurrent neural network based language model 2010

Intuition

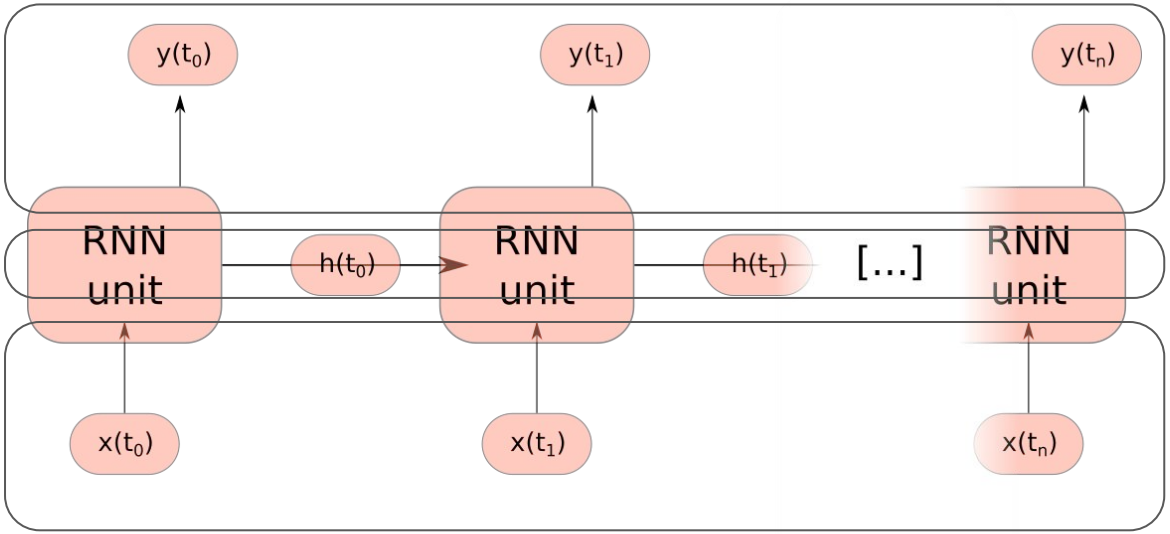
Each time step is processed by a unit of the RNN which uses as input the input at the time t and the hidden state, output by the previous steps.



Implementation

There are three layers to the network:

- input layer x
- hidden layer s
- output layer y



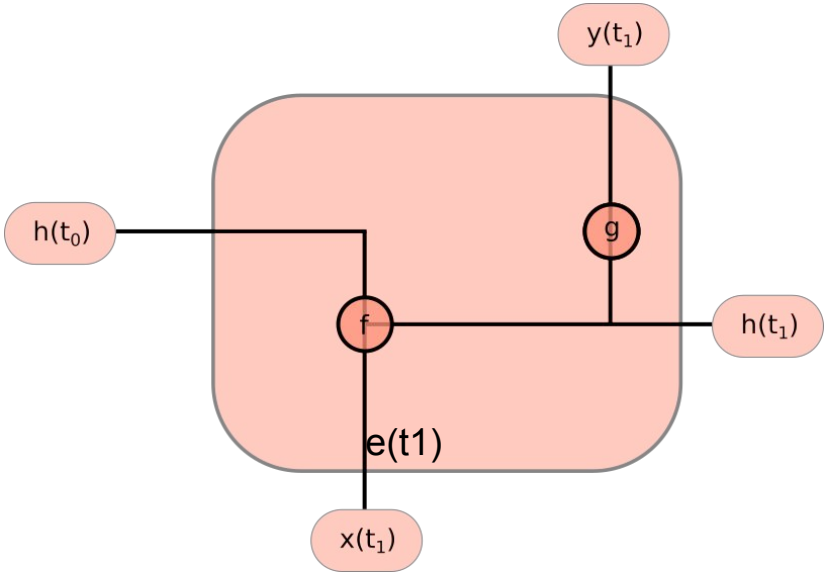
Implementation - 2

$$x^{(t)} \in \mathbb{R}^{|V|}$$

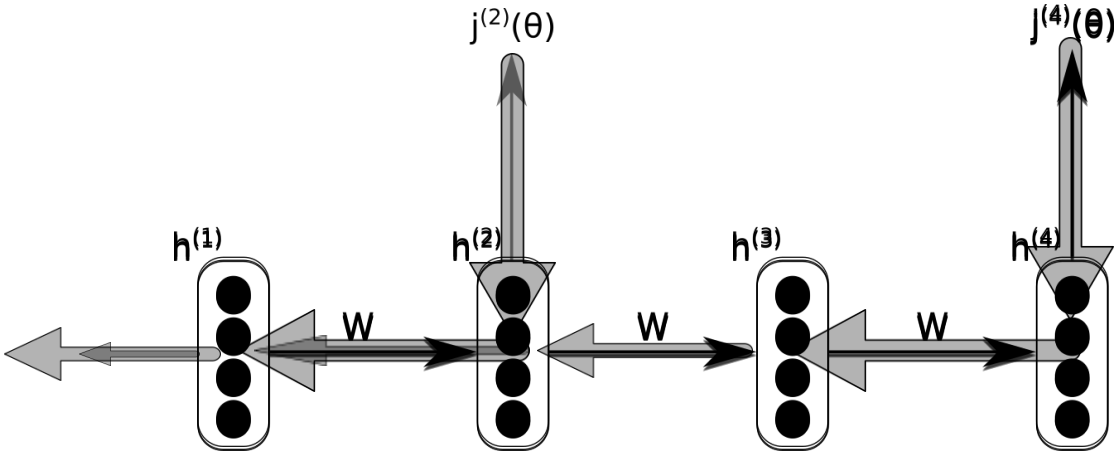
$$e^{(t)} = Ex^{(t)}$$

$$h^{(t)} = \sigma\left(W_h h^{(t-1)} + W_e e^{(t)} + b_1\right)$$

$$\hat{y^{(t)}} = softmax\left(Uh^{(t)} + b_2\right) \in \mathbb{R}^{|V|}$$



Limitations - vanishing gradient



$$\frac{\partial J^{(4)}}{\partial h^{(1)}} = \frac{\partial h^{(2)}}{\partial h^{(1)}} \times \frac{\partial h^{(3)}}{\partial h^{(2)}} \times \frac{\partial h^{(4)}}{\partial h^{(3)}} \times \frac{\partial J^{(4)}}{\partial h^{(4)}}$$

example:

The author of the books ___

Is or are?

RNN might be inclined to answer are as it related to books (the most recent neighbor with larger gradient)

Enters LSTMs

LSTM

Was first introduced in 1997 by Hochreiter and Schmidhuber

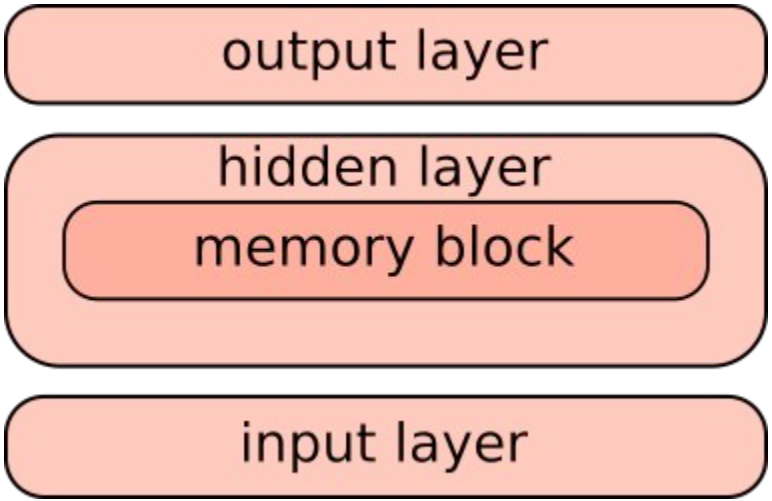
Was first made usable in 1999 by Gers, Schmidhuber and Cummins thanks to the introduction of the forget gate.

The LSTM is an RNN with a controlled memory. Making it much more adaptable for long time series.

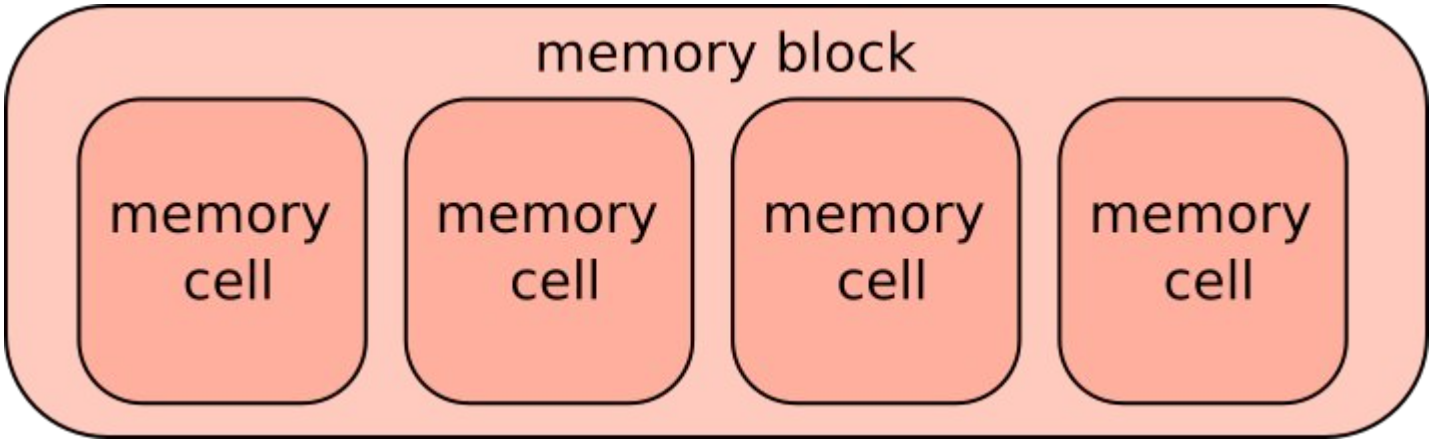
Sepp Hochreiter et al. LONG SHORT-TERM MEMORY - 1997
Felix A. Gers et al. Learning to Forget: Continual Prediction with LSTM - 1999

Intuition

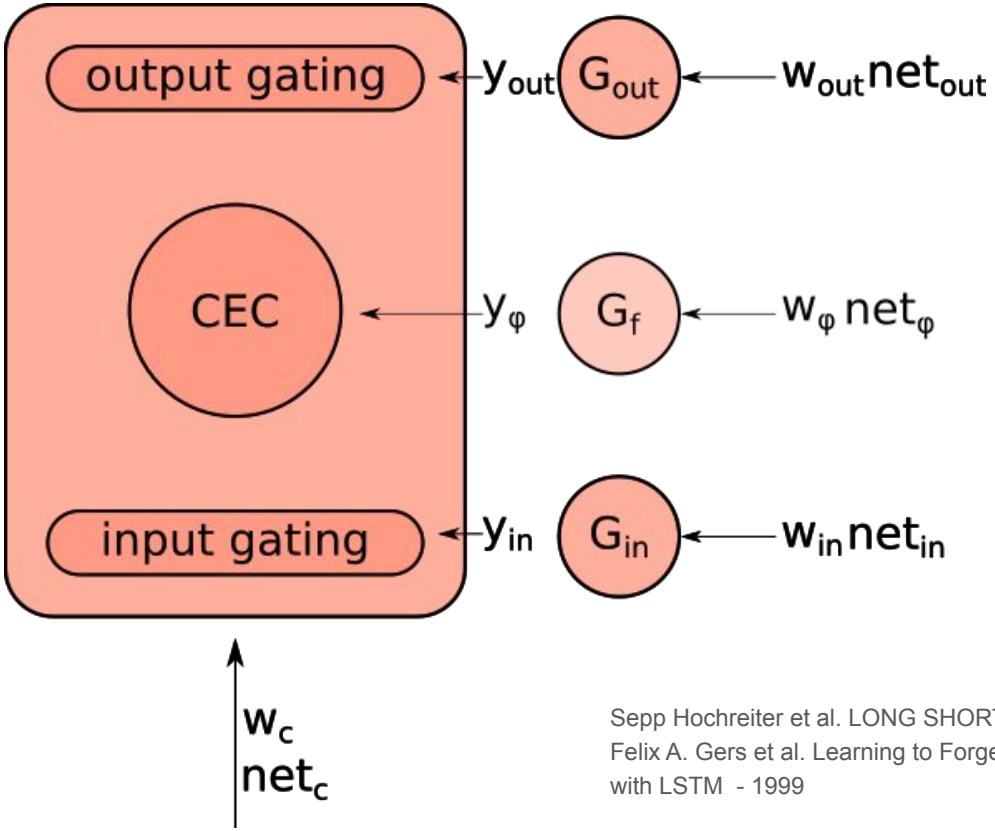
A memory system has to allow the user to decide when to write, read and delete the memory. The user in this case is the algorithm and the memory is the cell state at a specific time.



Memory block

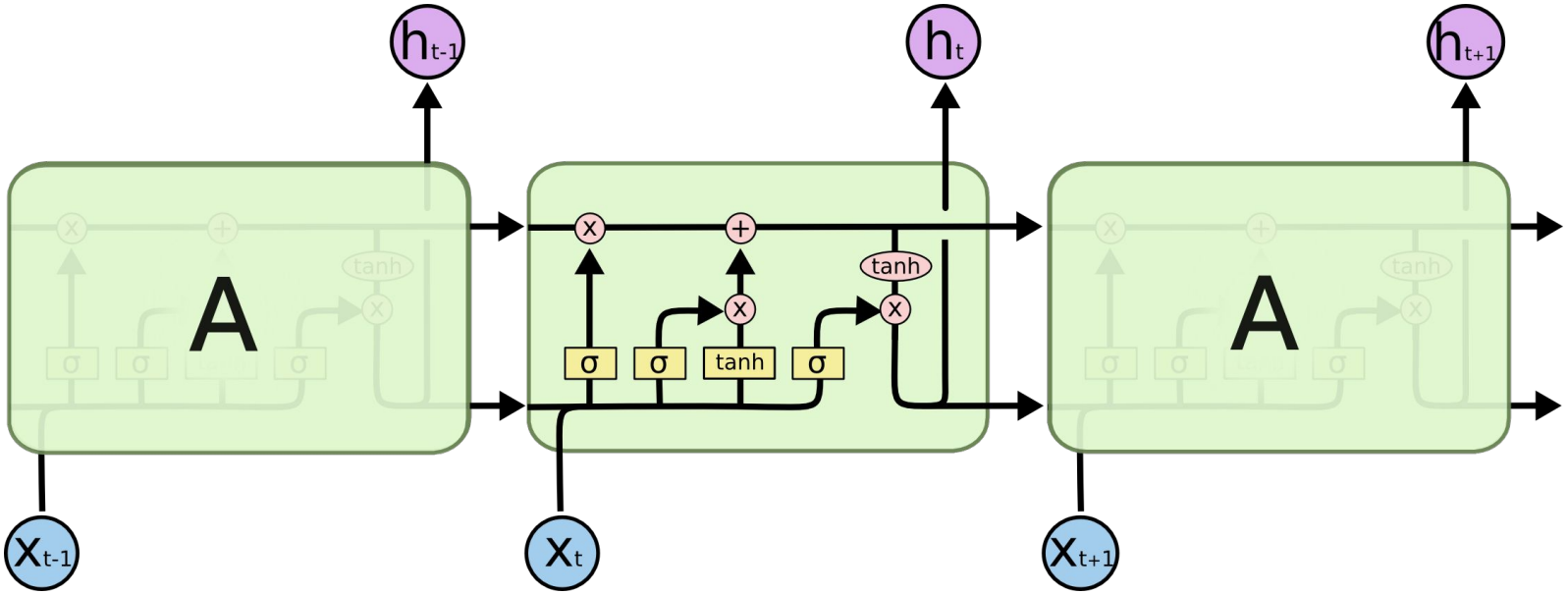


Memory cell



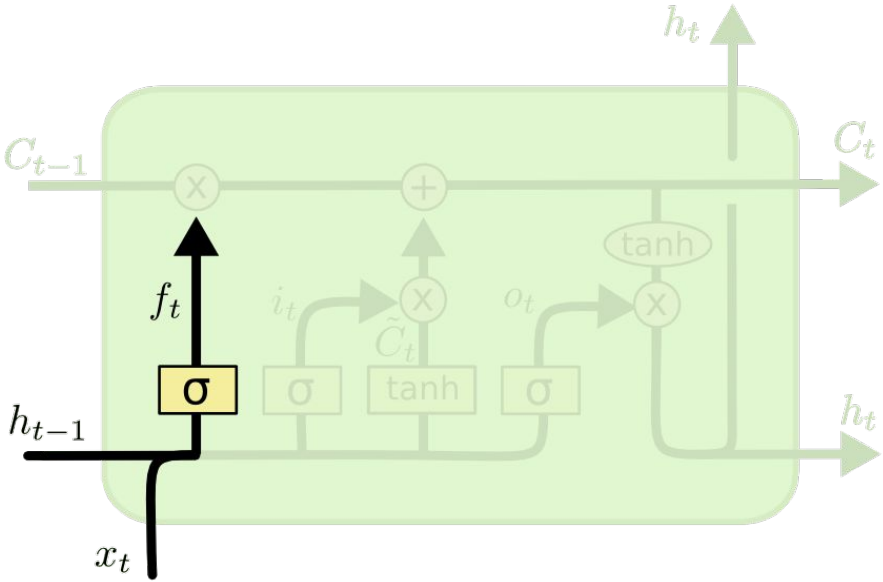
Sepp Hochreiter et al. LONG SHORT-TERM MEMORY - 1997
Felix A. Gers et al. Learning to Forget: Continual Prediction with LSTM - 1999

Another view

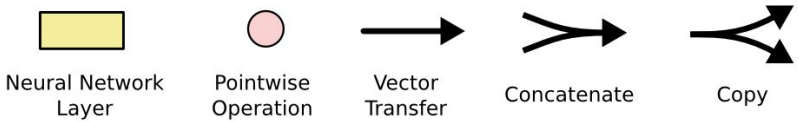


Chris Olah - Understanding LSTM Networks

Forget gate

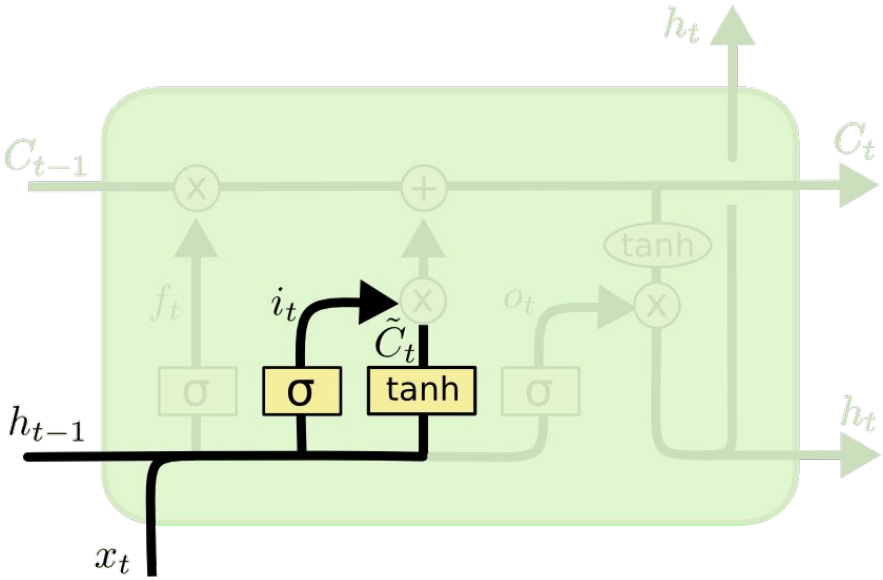


$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$



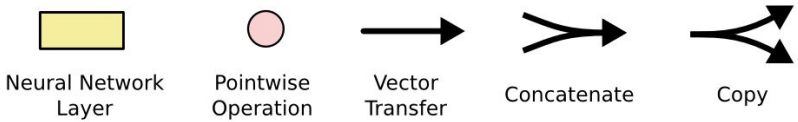
Chris Olah - Understanding LSTM Networks

Input gate



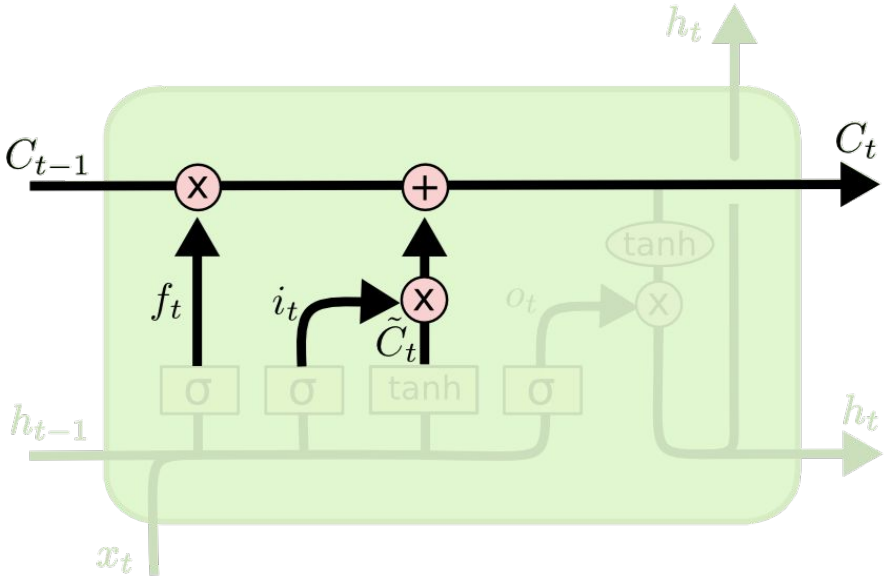
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

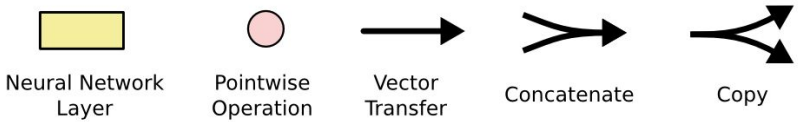


Chris Olah - Understanding LSTM Networks

Cell state update

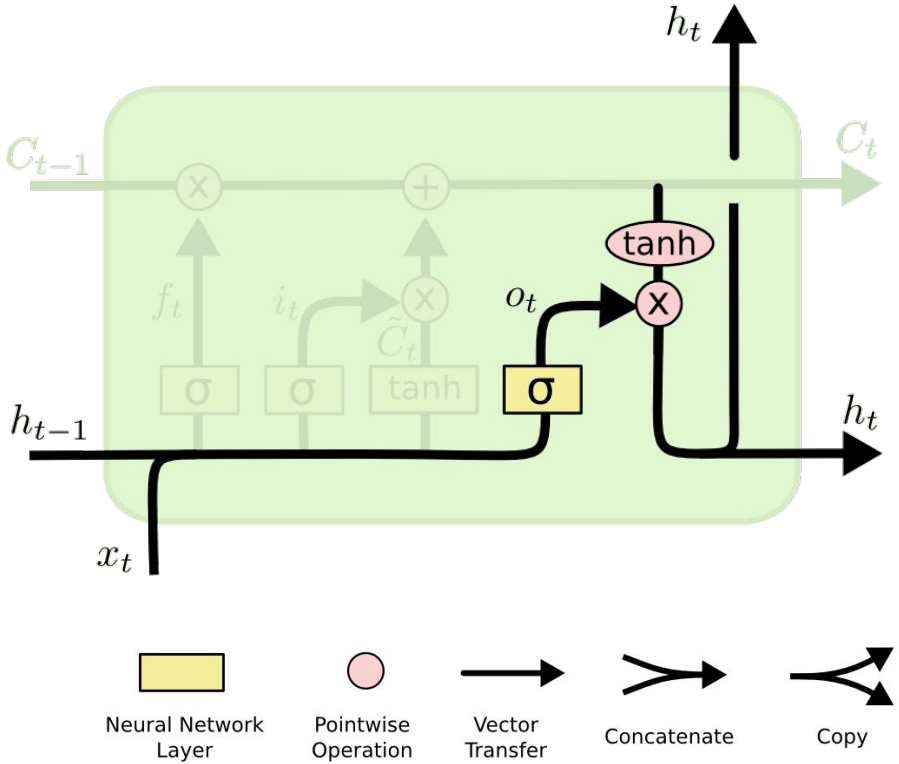


$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$



Chris Olah - Understanding LSTM Networks

Output gate



$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

Chris Olah - Understanding LSTM Networks

Applications

RNNs and in particular LSTMs have been the key to reach all the milestones of NLP in the last decade

The hidden states can be seen as contextualized embeddings!

Enters ELMo

Presentation structure

Word embeddings

- ✓ Before word2vec
- ✓ Word2vec
- ● Contextual embeddings ELMo

Language models

- ✓ Before neural networks
- ✓ RNNs
- ✓ LSTMs
- ● Transformers

embeddings colaboratory

Language models colaboratory

Why ELMo?

A problem with word2vec vectors:

"Wait a second" → $[v_1, v_2, v_3]$

"He finished second" → $[v_4, v_5, v_3]$

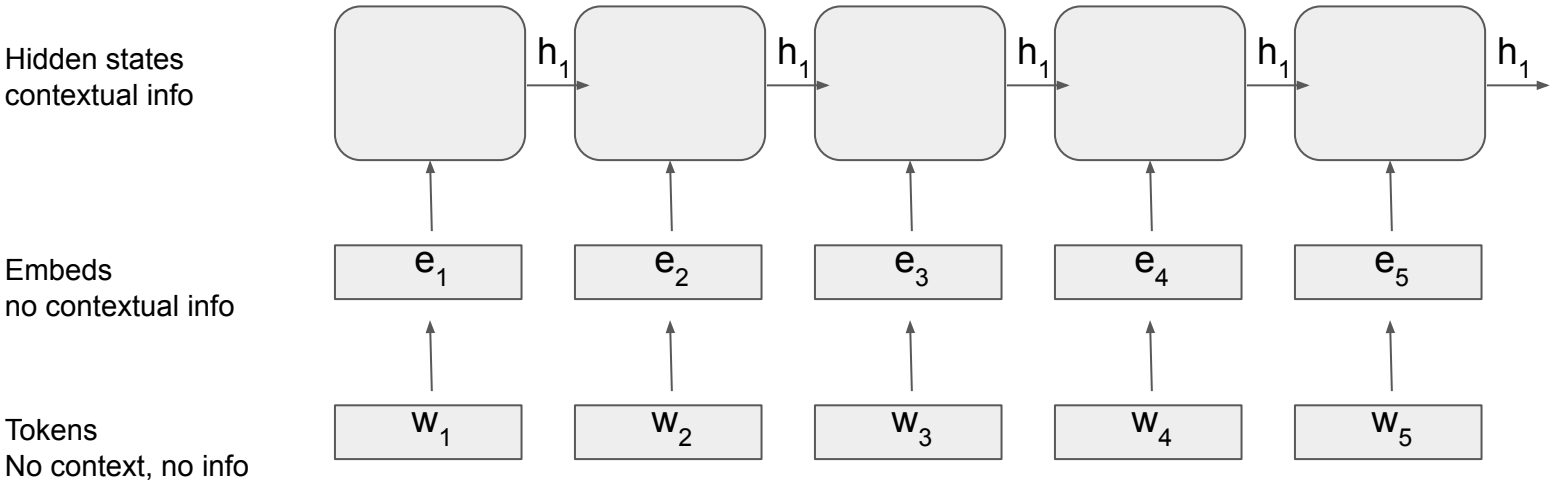
The vectors don't have any way to correct for polysemy.

The context in which a word appears doesn't have any influence on the vector

Why ELMo?

RNNs always produce contextual embeddings in their hidden states

Why not pretrain a model to find these **contextual word embeddings** from the beginning?



How it's made

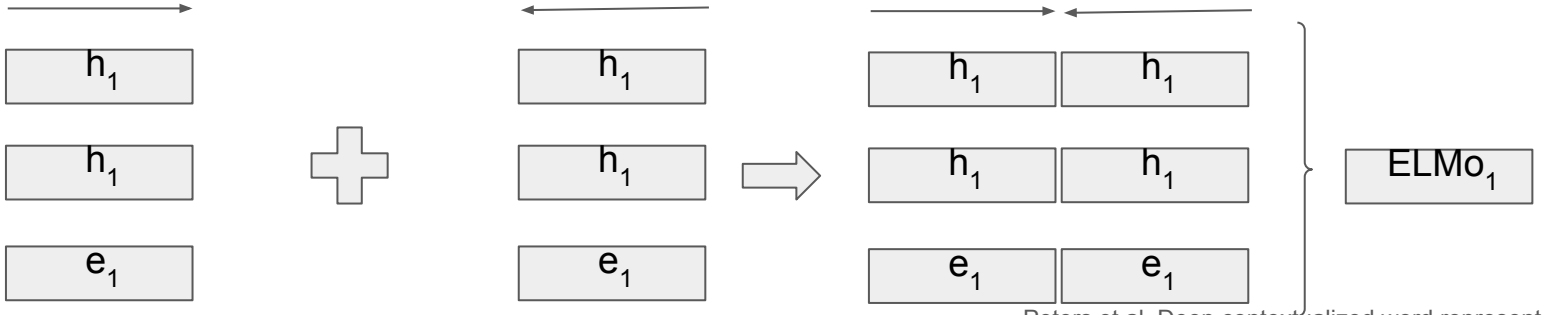
ELMo = **E**mbeddings from **L**anguage **Mo**dels

Two bidirectional LSTMs encode the words in a context, one from the left one from the right.

Contextual embeddings are derived by the weighted sum of concatenated hidden states and initial embeddings.

Higher layers capture context-dependent aspects of word meaning

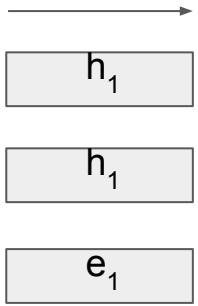
Lower layers model aspects of syntax



Into the biLSTMs layers

ELMo representations are deep = depend on all of the layers of the biLSTM through linear combination

- Higher layers capture context-dependent aspects of word meaning
- Lower layers model aspects of syntax



Implementation

Bidirectional Language Models biLM

$$p(t_1, t_2, ..., t_N) = \prod_{k=1} Np(t_k|t_1, t_2, ..., t_{k-1})$$

$$p(t_1, t_2, ..., t_N) = \prod_{k=1} Np(t_k|t_{k+1}, t_{k+2}..., t_N)$$

$$\Sigma_{k=1} N \left(\log p(t_k|t_1, ..., t_{k-1}; \Theta_x, \overrightarrow{\Theta}_{LSTM}, \Theta_s) + \log p(t_k|t_{k+1}, ..., t_N; \Theta_x, \overleftarrow{\Theta}_{LSTM}, \Theta_s) \right) \quad (1)$$

ELMo's multiple token representations

Bidirectional Language Models biLM

$$R_k = \{x_k^{LM}, \overrightarrow{x}_{k,j}^{LM}, \overleftarrow{x}_{k,j}^{LM} \mid j = 1, ..., L\}$$

$$R_k = \{h_{k,l}^{LM} \mid j = 0, ..., L\}$$

$$\mathbf{ELMo}_k = E(R_k; \mathbf{\Theta}_e)$$

$$\text{ELMo}_k^{task} = E(R_k; \Theta^{task}) = \gamma^{task} \sum_{j=0}^{L} s_j^{task} h_{k,l}^{LM}$$

Example

	Source	Nearest Neighbors
GloVe	play	playing, game, games, played, players, plays, player, Play, football, multiplayer
biLM	Chico Ruiz made a spectacular play on Alusik 's grounder {...}	Kieffer , the only junior in the group , was commended for his ability to hit in the clutch , as well as his all-round excellent play .
	Olivia De Havilland signed to do a Broadway play for Garson {...}	{...} they were actors who had been handed fat roles in a successful play , and had talent enough to fill the roles competently , with nice understatement .

Table 4: Nearest neighbors to "play" using GloVe and the context embeddings from a biLM.

Results

TASK	PREVIOUS SOTA		OUR BASELINE	ELMO + BASELINE	INCREASE (ABSOLUTE/ RELATIVE)
SQuAD	Liu et al. (2017)	84.4	81.1	85.8	4.7 / 24.9%
SNLI	Chen et al. (2017)	88.6	88.0	88.7 ± 0.17	0.7 / 5.8%
SRL	He et al. (2017)	81.7	81.4	84.6	3.2 / 17.2%
Coref	Lee et al. (2017)	67.2	67.2	70.4	3.2 / 9.8%
NER	Peters et al. (2017)	91.93 ± 0.19	90.15	92.22 ± 0.10	2.06 / 21%
SST-5	McCann et al. (2017)	53.7	51.4	54.7 ± 0.5	3.3 / 6.8%

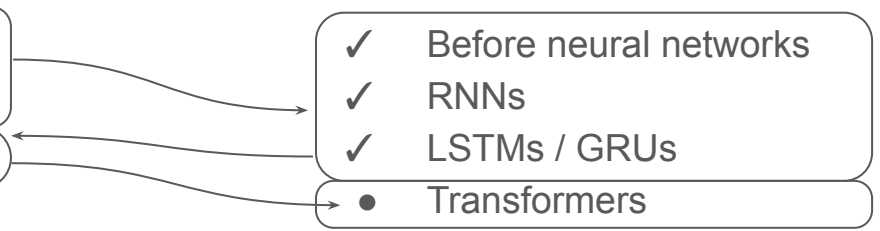
Presentation structure

Word embeddings

- ✓ Before word2vec
- ✓ Word2vec
- ✓ Contextual embeddings ELMo

Language models

- ✓ Before neural networks
- ✓ RNNs
- ✓ LSTMs / GRUs
- ● Transformers



embeddings colaboratory

Language models colaboratory

The situation in before transformers

LSTMs and variations of its standard form rule the field of NLP.

The sequential nature of LSTMs seems to be the biggest bottleneck for computational complexity.

The rise of bidirectional LSTMs shows the importance of truly bidirectional language models.

Attention is all you need

Vaswani et al. propose a new architecture built by keeping the three following objectives in mind:

1. the total computational complexity per layer
2. the amount of computation that can be parallelized, as measured by the minimum number of sequential operations required
3. the path length between long-range dependencies in the network

Attention is all you need

Table 1: Maximum path lengths, per-layer complexity and minimum number of sequential operations for different layer types. n is the sequence length, d is the representation dimension, k is the kernel size of convolutions and r the size of the neighborhood in restricted self-attention.

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)	$O(r \cdot n \cdot d)$	$O(1)$	$O(n/r)$

A. Vaswani et al. Attention Is All You Need - 2017

The transformer

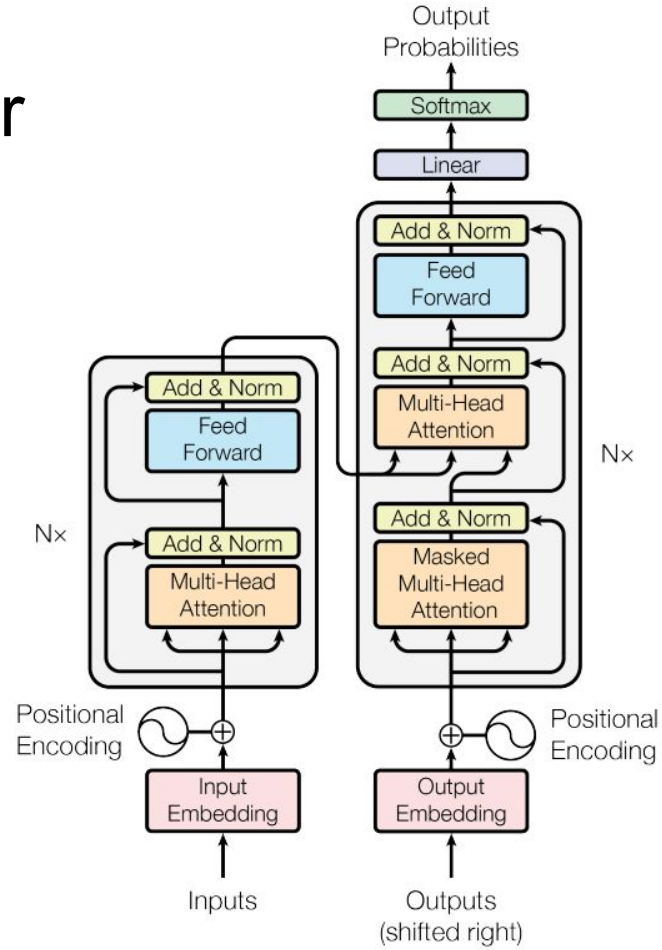
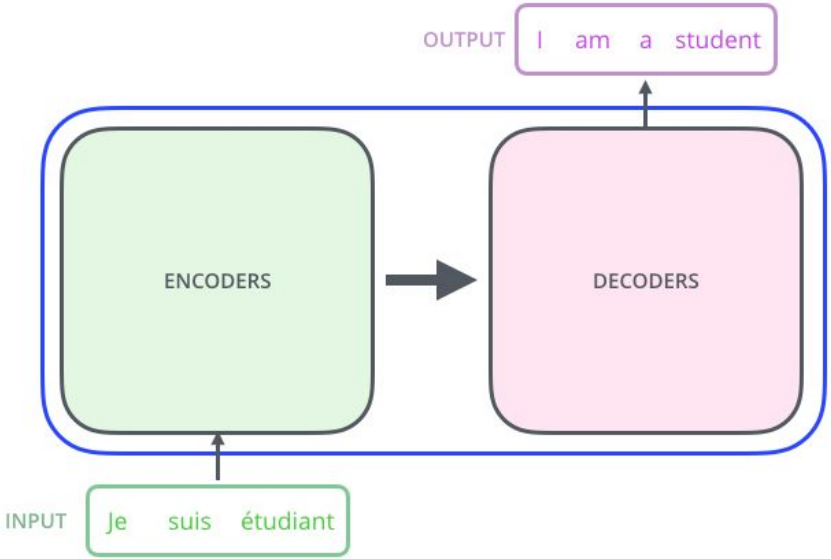


Figure 1: The Transformer - model architecture.

A. Vaswani et al. Attention Is All You Need - 2017

Overview

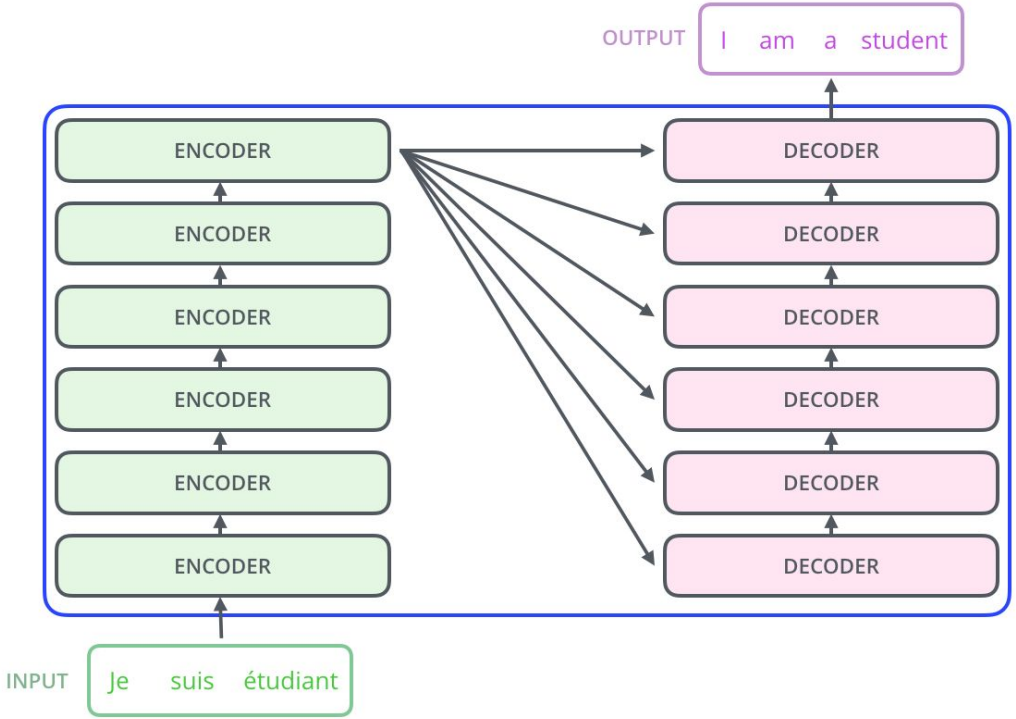
An encoder decoder (seq2seq) architecture:



A. Vaswani et al. Attention Is All You Need - 2017

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

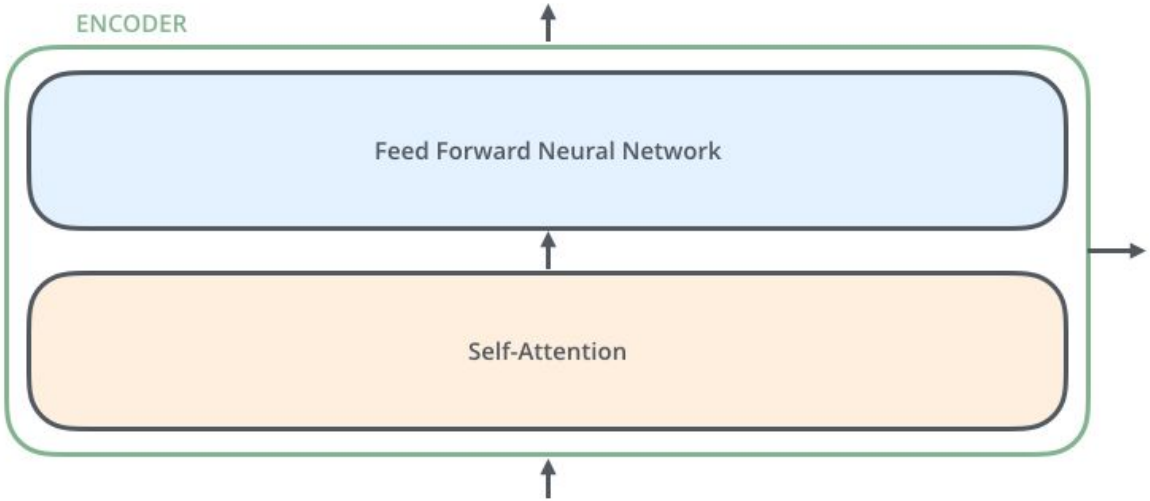
Stacked encoder-decoders



A. Vaswani et al. Attention Is All You Need - 2017

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

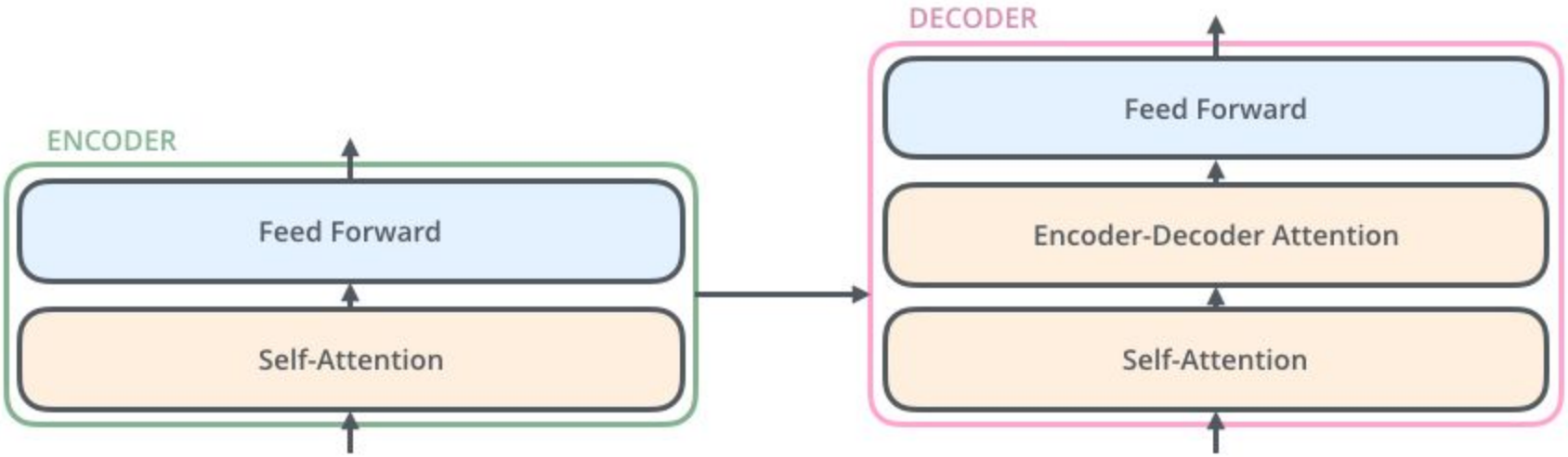
Inside an encoder



A. Vaswani et al. Attention Is All You Need - 2017

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

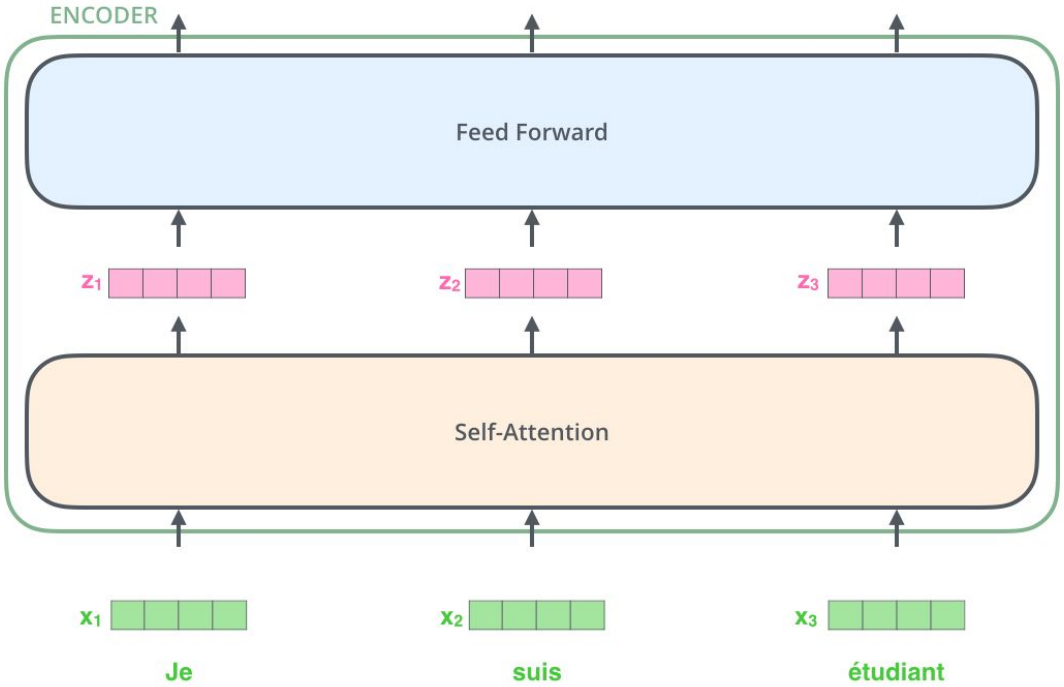
Inside an encoder-decoder pair



A. Vaswani et al. Attention Is All You Need - 2017

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

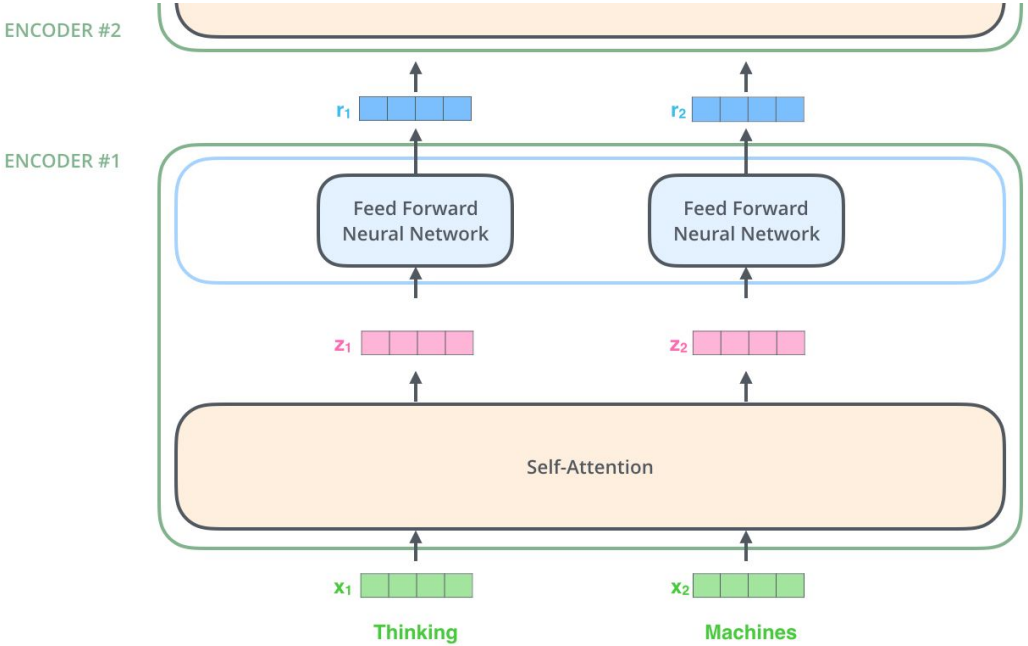
Stacked encoder-decoders



A. Vaswani et al. Attention Is All You Need - 2017

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

Stacked encoder-decoders



A. Vaswani et al. Attention Is All You Need - 2017

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

Self attention

A way to find which words are related to each other.

Example:

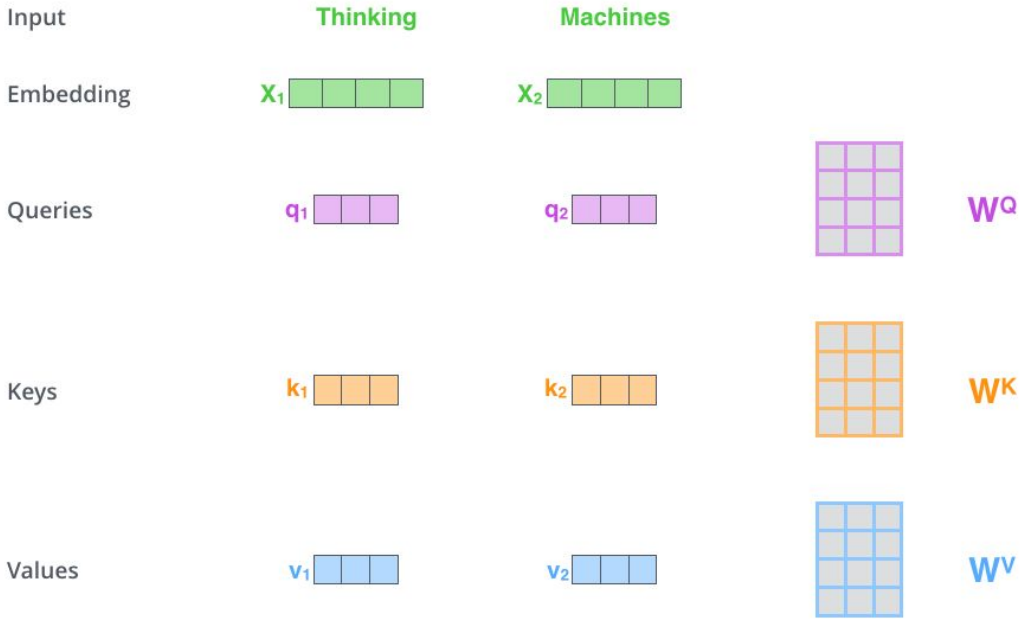
”`The animal didn't cross the street because it was too tired`”

What does “it” in this sentence refer to? Is it referring to the street or to the animal? It’s a simple question to a human, but not as simple to an algorithm.

When the model is processing the word “it”, self-attention allows it to associate “it” with “animal”.

A. Vaswani et al. Attention Is All You Need - 2017
http://jalammar.github.io/illustrated-transformer/

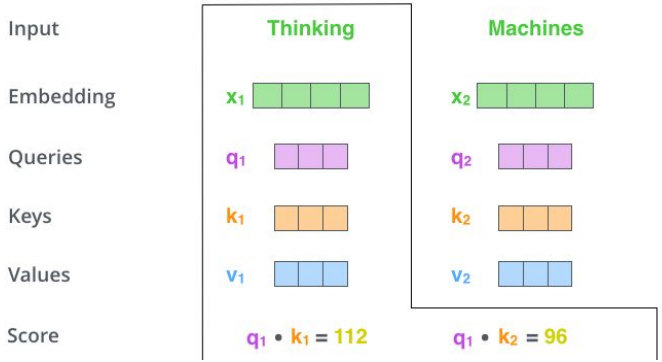
Self attention



A. Vaswani et al. Attention Is All You Need - 2017

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

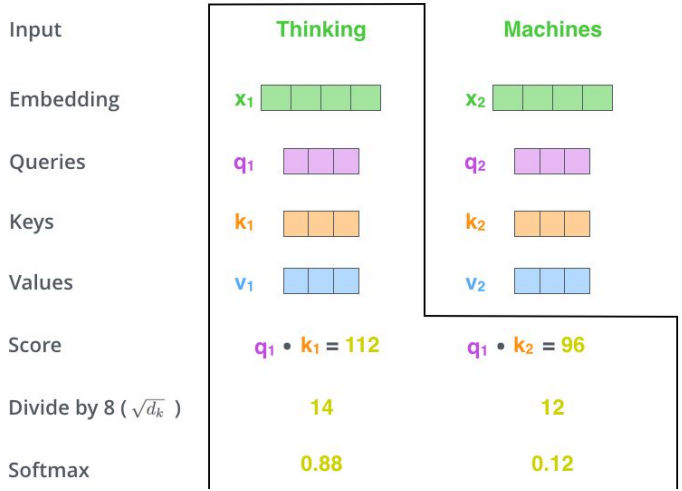
Self attention



A. Vaswani et al. Attention Is All You Need - 2017

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

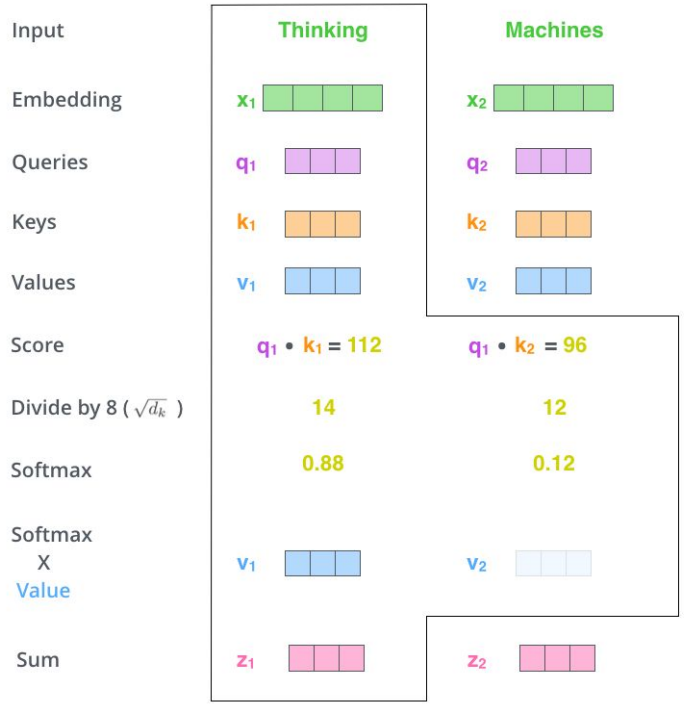
Self attention



A. Vaswani et al. Attention Is All You Need - 2017

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

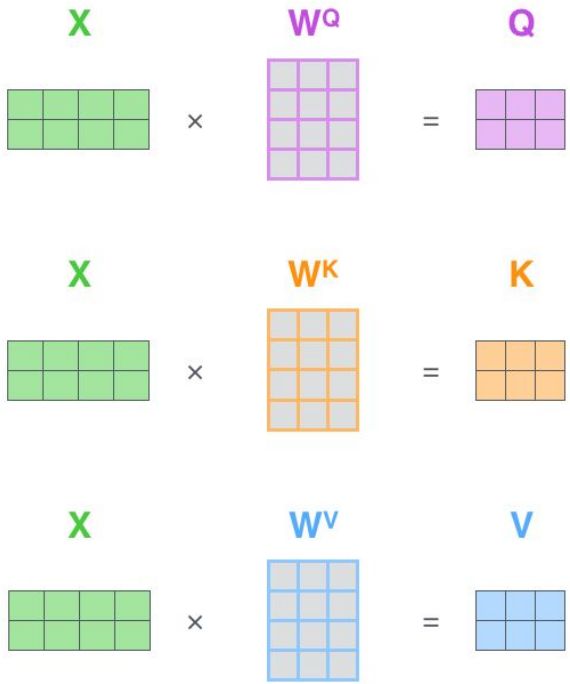
Stacked encoder-decoders



A. Vaswani et al. Attention Is All You Need - 2017

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

Attention matrixes



A. Vaswani et al. Attention Is All You Need - 2017

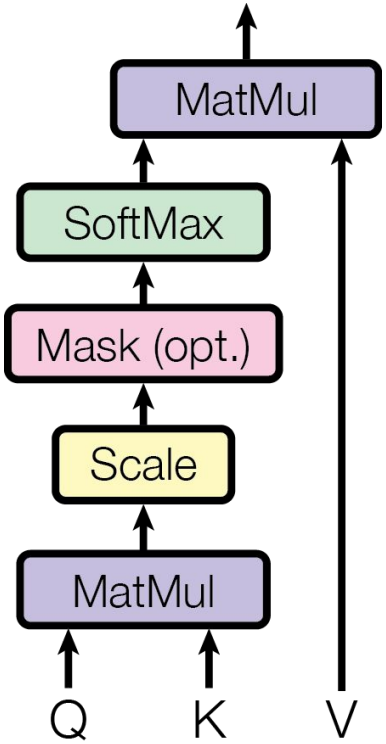
http://jalammar.github.io/illustrated-transformer/

Scaled dot product attention:

Input:
queries and keys of dimension d_k
values of dimension d_v

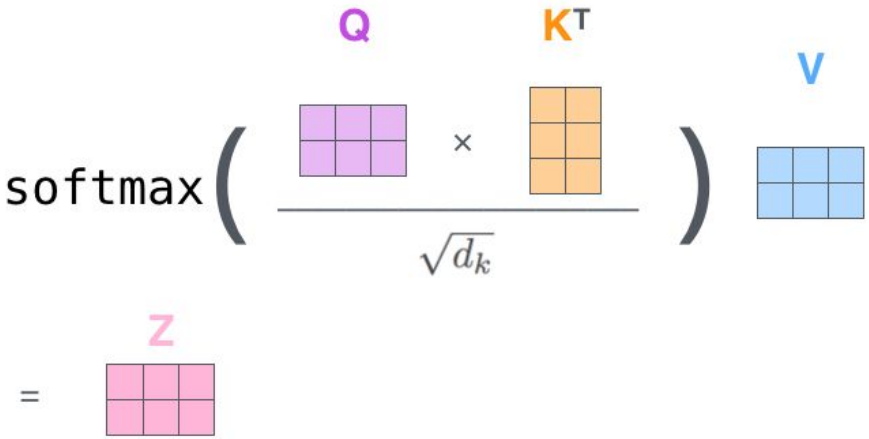
Compute the dot products of the query with all keys, divide each by $\sqrt{d_k}$
Apply a softmax function to obtain the weights on the values.

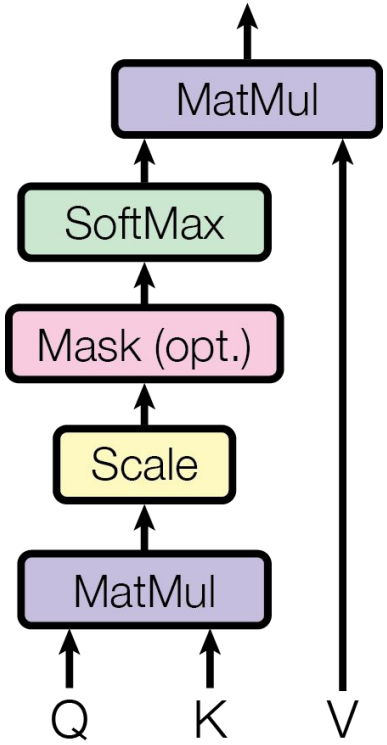
Compute the attention function on a set of queries simultaneously, packed together into a matrix Q. The keys and values are also packed together into matrices K and V.



A. Vaswani et al. Attention Is All You Need - 2017
http://jalammar.github.io/illustrated-transformer/

Scaled dot product attention:





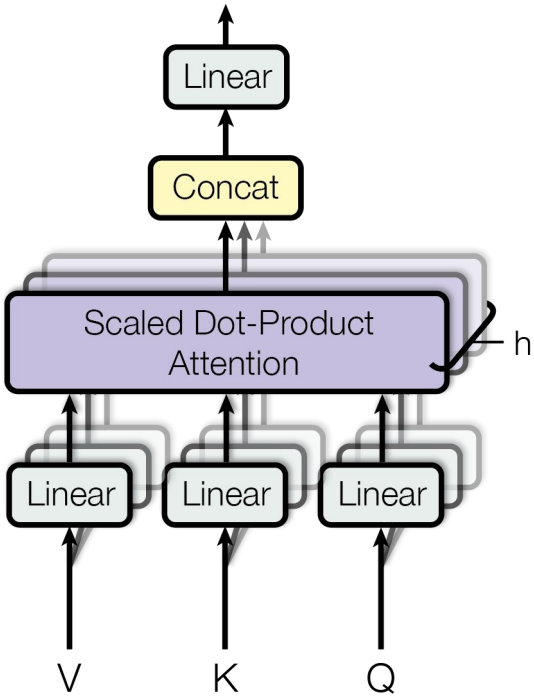
A. Vaswani et al. Attention Is All You Need - 2017

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

Multi headed attention:

The queries keys and values are projected h times with different learned linear projections.

This allows for the transformer to use information from different representation subspaces.



A. Vaswani et al. Attention Is All You Need - 2017

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

The decoder

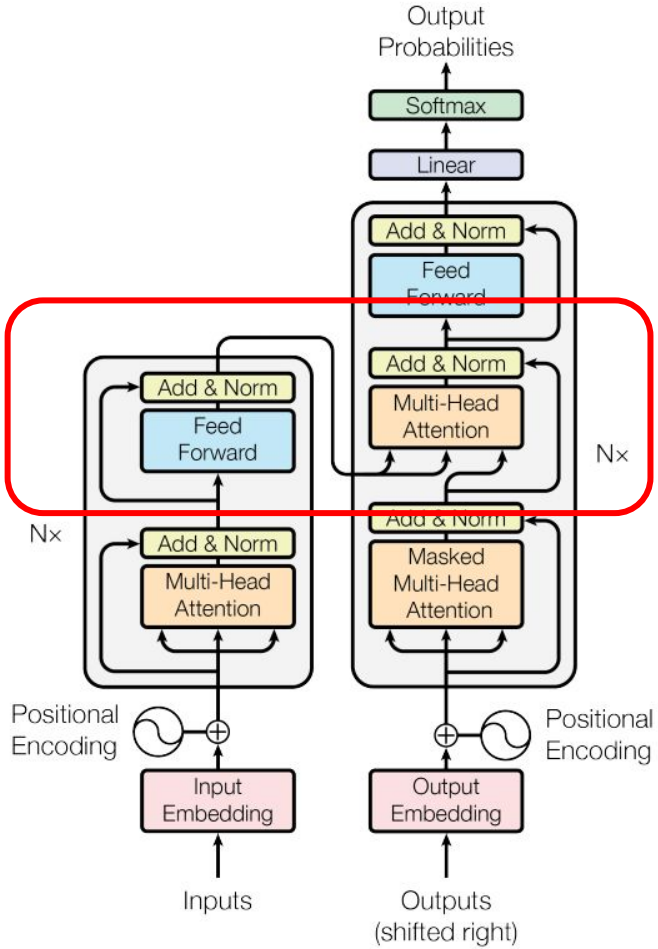
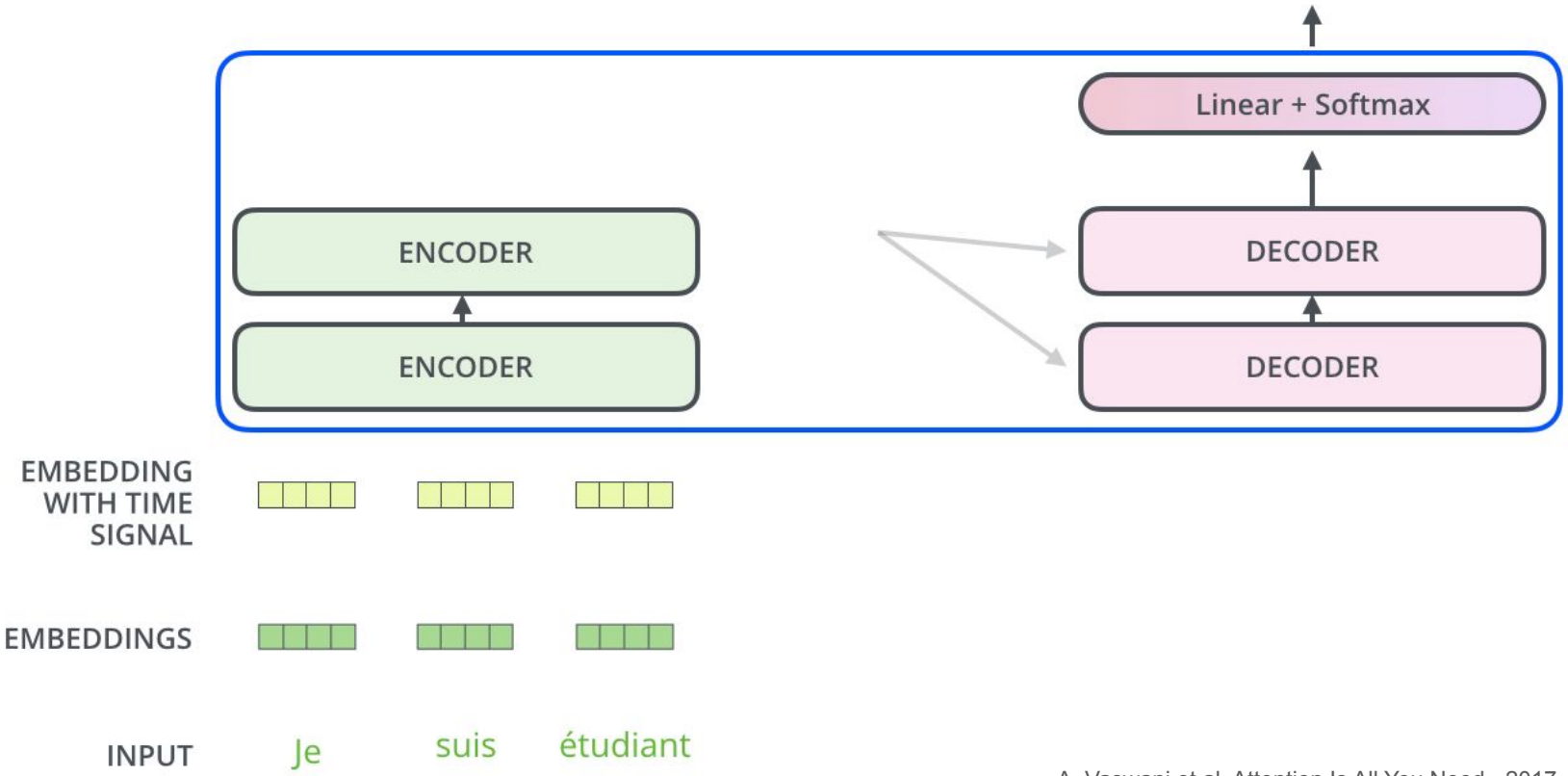


Figure 1: The Transformer - model architecture.

A. Vaswani et al. Attention Is All You Need - 2017

The transformer in action



A. Vaswani et al. Attention Is All You Need - 2017

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

BERT

Bidirectional **E**ncoder **R**epresentations from **T**ransformers

Powerful word embeddings

Masked language models and next sentence prediction

BERT advances the state of the art for eleven NLP tasks

Opened up a new era of pretrained models all available through huggingface.co and their transformers python package

Jacob Devlin et al. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding - 2019

Word tokenization - wordpiece

“Gulliver’s Travels” J. Swift - October 28, 1726

The author mocks linguists for their seemingly futile efforts:

“by cutting polysyllables into one, and leaving out verbs and participles; because in reality all things imaginable are but nouns.”

Just shy of three hundred years later we do this all the time with great results, admittedly for applications that were unimaginable at the time.

Jonathan Swift “Gulliver's Travels” October 28, 1726
Schister, Nakajima Japanese and Korean voice search - 2012

Wordpiece

based on a balance between vocabulary size and out-of-vocab words.

Example:

tokenize(“flower”) = [6546]

decode([6546]) = “flower”

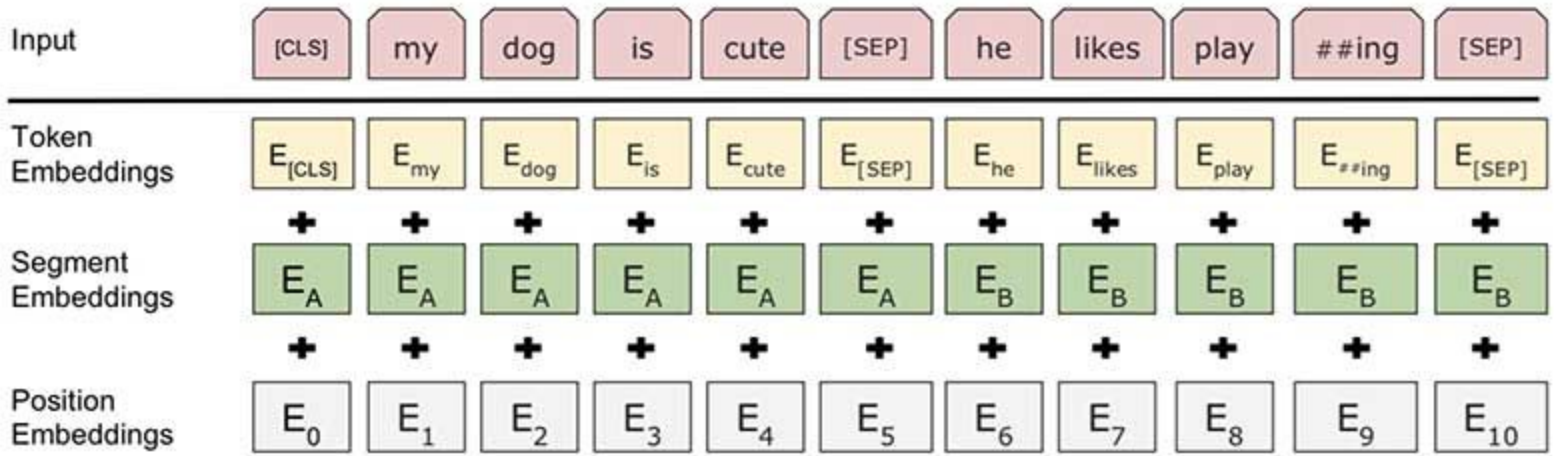
tokenize(“cauliflower”) = [6187, 15859, 14156]

decode([6187, 15859, 14156]) = [“ca”, “##uli”, “##flower”]

Jonathan Swift “Gulliver's Travels” October 28, 1726

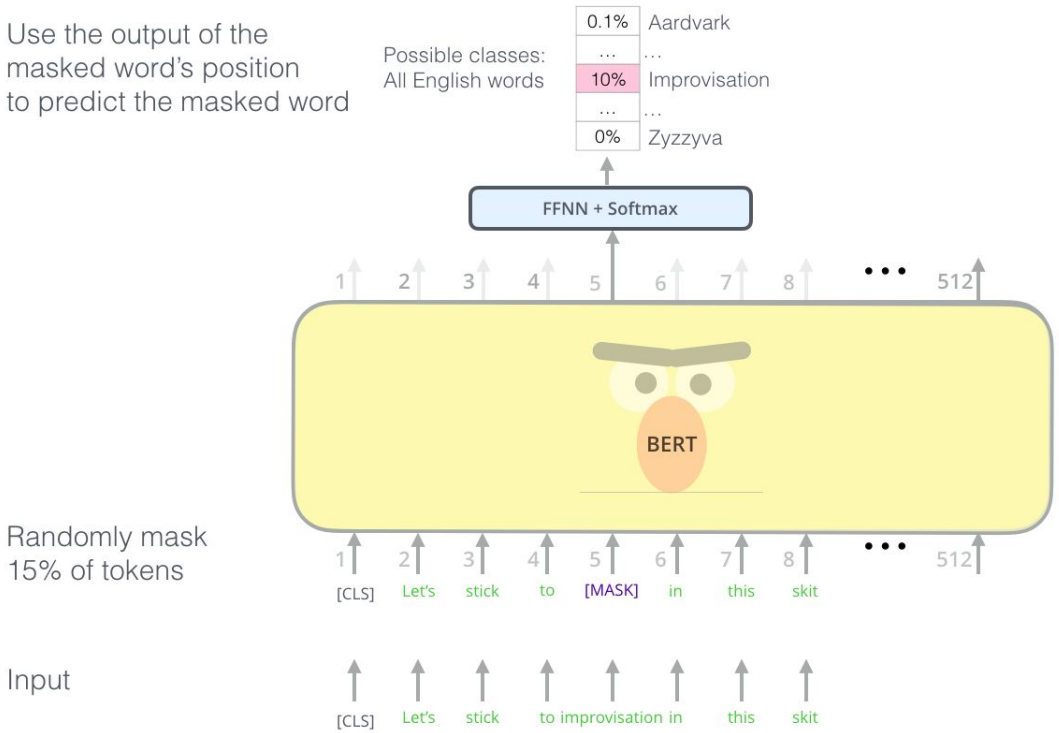
Schister, Nakajima Japanese and Korean voice search - 2012

Word embedding



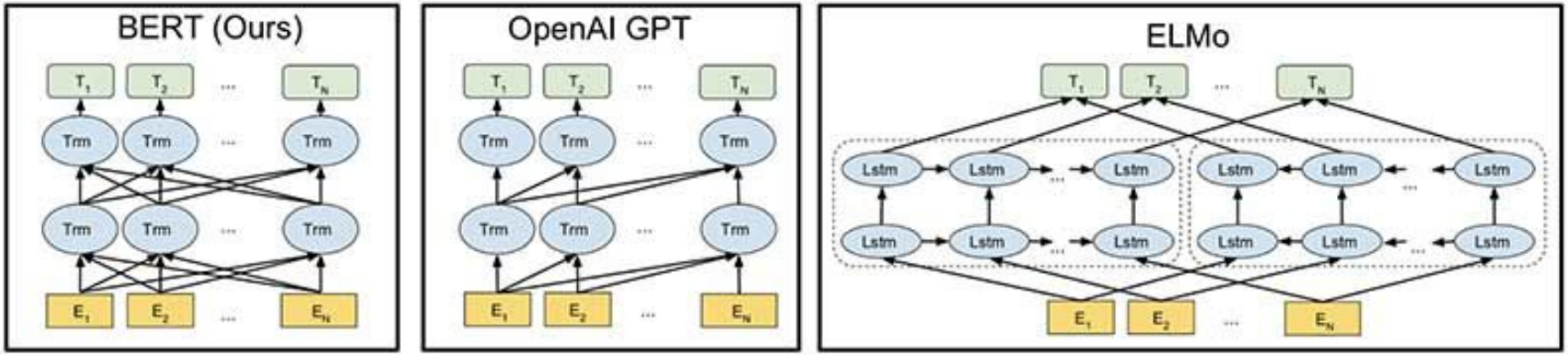
Jacob Devlin et al. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding - 2019

Masked language model



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

Comparison with GPT and ELMo



Jacob Devlin et al. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding - 2019

Fine tuning

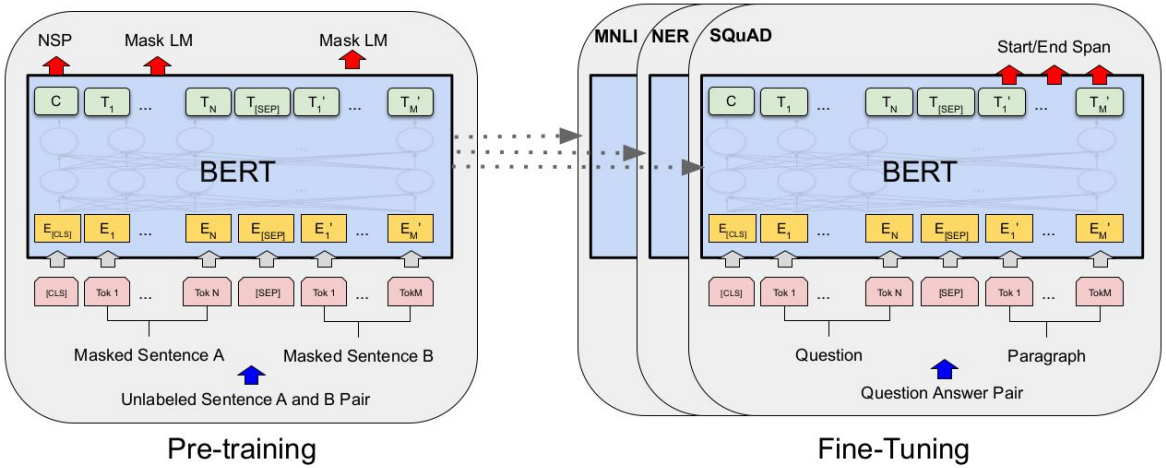


Figure 1: Overall pre-training and fine-tuning procedures for BERT. Apart from output layers, the same architectures are used in both pre-training and fine-tuning. The same pre-trained model parameters are used to initialize models for different down-stream tasks. During fine-tuning, all parameters are fine-tuned. `[CLS]` is a special symbol added in front of every input example, and `[SEP]` is a special separator token (e.g. separating questions/answers).

Next sentence prediction

Improving large scale language understanding, useful in question answering for instance.

Sentence A + sentence B

Sentence B 50% of the time follows A and the rest it doesn’t.

SOTAs

System	Dev		Test	
	EM	F1	EM	F1
Top Leaderboard Systems (Dec 10th, 2018)				
Human	-	-	82.3	91.2
#1 Ensemble - nlnet	-	-	86.0	91.7
#2 Ensemble - QANet	-	-	84.5	90.5
Published				
BiDAF+ELMo (Single)	-	85.6	-	85.8
R.M. Reader (Ensemble)	81.2	87.9	82.3	88.5
Ours				
$\text{BERT}_{\text{BASE}}$ (Single)	80.8	88.5	-	-
$\text{BERT}_{\text{LARGE}}$ (Single)	84.1	90.9	-	-
$\text{BERT}_{\text{LARGE}}$ (Ensemble)	85.8	91.8	-	-
$\text{BERT}_{\text{LARGE}}$ (Sgl.+TriviaQA)	**84.2**	**91.1**	**85.1**	**91.8**
$\text{BERT}_{\text{LARGE}}$ (Ens.+TriviaQA)	**86.2**	**92.2**	**87.4**	**93.2**

Table 2: SQuAD 1.1 results. The BERT ensemble is 7x systems which use different pre-training checkpoints and fine-tuning seeds.

Jacob Devlin et al. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding - 2019

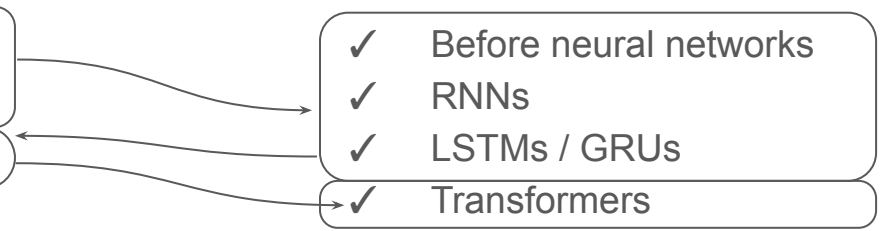
Presentation structure

Word embeddings

- ✓ Before word2vec
- ✓ Word2vec
- ✓ Contextual embeddings ELMo

Language models

- ✓ Before neural networks
- ✓ RNNs
- ✓ LSTMs / GRUs
- ✓ Transformers



[embeddings colaboratory]

[Language models colaboratory]